Distributed edge computing for enhanced IoT devices and new generation network efficiency

Mohammed Laroui

Université Paris Cité

LIPADE
Laboratoire d'Informatique PAris DEscartes

Ph.D. Thesis in
Computer Science

By
M^{r}. Mohammed Laroui

Distributed Edge Computing For Enhanced IoT Devices and New Generation Network Efficiency

Thesis Committee :

Pr.	Hassine Moungla	University of Paris Cite	Thesis Director
Dr.	Zohra Slama	UDL SBA	Supervisor
Pr.	Hind Castel	Telecom SudParis	Reviewer
Pr.	Ken Chen	Sorbonne University	Reviewer
Pr.	Hossam Afifi	Telecom SudParis	Examiner
Pr.	Badr Benmammar	University of Tlemcen	Examiner
Dr.	Nassim Dennouni	University of Chlef	Examiner

Year : 2022

Université Paris Cité
En cotutelle avec L'Université Djillali Liabes de Sidi Bel Abbes

École Doctorale Informatique, Télécommunication et Électronique «Edite de Paris *: ED130»*
*Laboratoire d'Informatique Paris Descartes (*LIPADE*)*

Distributed Edge Computing For Enhanced IoT Devices and New Generation Network Efficiency

Par Mohammed Laroui

Thèse de doctorat en Informatique (Réseaux)

Dirigée par Hassine Moungla
Et par Zohra Slama

Présentée et soutenue publiquement le 21 Juillet 2022

Devant un jury composé de :

Hind Castel, Professeur, Telecom SudParis, Rapporteur

Ken Chen, Professeur, Sorbonne Université, Rapporteur

Hossam Afifi, Professeur, Telecom SudParis, Examinateur

Badr Benmammar, Professeur, Université de Tlemcen, Examinateur

Nassim Dennouni, Docteur, Université de Chlef, Examinateur

Acknowledgement

I SINCERELY THANK MY PARENTS, MY BROTHER, MY SISTERS AND THE REST OF MY FAMILY FOR THEIR HELP AND SUPPORT, UNCONDITIONAL LOVE FOR YOU.

I WOULD LIKE TO THANK THE RESEARCHERS WHO HAVE CONTRIBUTED TO MY PH.D DURING ALL MY STUDIES YEARS.

FIRST, I WOULD LIKE TO THANK MY SUPERVISORS, THEIR EXPERTISE AND KNOWLEDGE HAVE GUIDED ME FOR MY RESEARCH.

IN ADDITION, I WOULD LIKE TO THANK THE JURY MEMBERS, REVIEWERS AND EXAMINERS FOR ACCEPTING THE INVITATION AND FOR THEIR HELPFUL FEEDBACK AND INSIGHTFUL COMMENTS.

FINALY, I WOULD LIKE TO THANK ALL MY FRIENDS AND PERSONS HELPED ME.

List of Abbreviations

Abbreviation	*Description*
IoT	Internet of Things.
CC	Cloud Computing.
MCC	Mobile Cloud Computing.
EC	Edge Computing.
MEC	Mobile Edge Computing.
VEC	Vehicular Edge Computing.
SDN	Software-Defined Networking.
NFV	Network Functions Virtualization.
VNF	Virtual Network Functions.
VNO	Virtual Network Operators.
SFC	Service Function Chaining.
UAV	Unmanned Aerial Vehicle.
QoS	Quality of service.
QoE	Quality of Experience.
AI	Artificial Intelligence.
ML	Machine Learning.
DL	Deep Learning.
QL	Q-Learning .
DRL	Deep Reinforcement Learning.
DQN	Deep Q-Network.
LSTM	Long Short-Term Memory.
GRU	Gated Recurrent Units.
SGD	Stochastic Gradient Descent.
RFID	Radio Frequency Identification.
WSN	Wireless Sensor Networks.
SaaS	Software as a Service.
PaaS	Platform as a Service.
IaaS	Infrastructure as a Service.
BTS	Base Transceiver Station.
VM	Virtual Machine.
gNb	GNodeB.
RSU	Roadside Unit.
AP	Access Point.
GPS	Global Positioning System.
RAN	Radio Access Network.
DNS	Domain Name System.
V2V	Vehicle-to-Vehicle.
V2I	Vehicle-to-Infrastructure.

V2X	Vehicle to Everything.
V2N	Vehicle to Network.
V2D	Vehicle to Device.
V2G	Vehicle to Grid.
V2P	Vehicle to Pedestrian(Person).
SCADA	Supervisory Control And Data Acquisition .
ETSI	European Telecommunications Standards Institute.
ISG	Industry Specification Group.
3GPP	3rd Generation Partnership Project .
MEPM	Mobile Edge Platform Manager.
MEO	Mobile Edge Orchestrator.
AMF	Application Rules Management Functions.
BCS	Backup Computing Server.
RPM	Remote Patient Monitoring.
HDMI	High Definition Multimedia Interface.
VR	Virtual Reality.
AR	Augmented Reality.
MR	Mixed Reality.
CPU	Central Processing Unit .
GPU	Graphics Processing Unit .
RAM	Random Access Memory.
KPIs	Key Performance Indicators.
ULLC	Ultra Low Latency Communication.
CDN	Content Delivery Network.
MVNO	Mobile Virtual Network Operator.
AMC	Adhoc Mobile Cloud.
AME	Adhoc Mobile Edge.
VME	Virtual Mobile Edge.
VMES	Virtual Mobile Edge Server.
SCP	Set Cover Problem.
WSC	Weighted Set Cover Problem.
MDP	Markov Decision Process.
CMDP	Constrained Markov Decision Process.
VEnPA	Optimal Virtual Edge nodes Placement Algorithm.
SO-VMEC	Service Offloading in Virtual Mobile Edge Computing.
OSPV	Optimal SFC Placement in IoT-VMEC.
MVEC	Mobile Vehicular Edge Computing.
OFMU	Optimal Follow Me UAV.
OVEAP	Optimal Virtual Edge-Autopilot Placement.
ESPV	Efficient SFC Placement Algorithm.
AFMU	Follow Me UAV Algorithm.

DVEAP	Deep Reinforcement Learning for Edge Autopilot Placement.
LQC	Low Quality Cloud.
LQE	Low Quality Edge.
MQC	Medium Quality Cloud.
MQE	Medium Quality Edge.
HQC	High Quality Cloud.
HQE	High Quality Edge.

Table 1 List of abbreviations.

Abstract

Traditional cloud infrastructure will face a series of challenges due to the centralization of computing, storage, and networking in a small number of data centers, and the long-distance between connected devices and remote data centers. To meet this challenge, edge computing seems to be a promising possibility that provides resources closer to IoT devices.

In the cloud computing model, compute resources and services are often centralized in large data centers that end-users access from the network. This model has an important economic value and more efficient resource-sharing capabilities. New forms of end-user experience such as the Internet of Things require computing resources near to the end-user devices at the network edge.

To meet this need, edge computing relies on a model in which computing resources are distributed to the edge of a network as needed, while decentralizing the data processing from the cloud to the edge as possible. Thus, it is possible to quickly have actionable information based on data that varies over time.

In this thesis, we propose novel optimization models to optimize the resource utilization at the network edge for two edge computing research directions, service offloading and vehicular edge computing. We study different use cases in each research direction. For the optimal solutions, First, for service offloading we propose optimal algorithms for services placement at the network edge (Tasks, Virtual Network Functions (VNF), Service Function Chain (SFC)) by taking into account the computing resources constraints. Moreover, for vehicular edge computing, we propose exact models related to maximizing the coverage of vehicles by both Taxis and Unmanned Aerial Vehicle (UAV) for online video streaming applications. In addition, we propose optimal edge-autopilot VNFs offloading at the network edge for autonomous driving. The evaluation results show the efficiency of the proposed algorithms in small-scale networks in terms of time, cost, and resource utilization.

To deal with dense networks with a high number of devices and scalability issues, we propose large-scale algorithms that support a huge amount of devices, data, and users requests. Heuristic algorithms are proposed for SFC orchestration, maximum coverage of mobile edge servers (vehicles). Moreover, The artificial intelligence algorithms (machine learning, deep learning, and deep reinforcement learning) are used for 5G VNF slices placement,

edge-autopilot VNF placement, and autonomous UAV navigation. The numerical results give good results compared with exact algorithms with high efficiency in terms of time.

Keywords: Internet of things (IoT), Cloud Computing, Edge/Fog Computing, Mobile Edge Computing, Artificial Intelligence, Optimization.

Résumé

Dans le cloud computing, les services et les ressources sont centralisés dans des centres de données auxquels l'utilisateur peut accéder à partir de ses appareils connectés. L'infrastructure cloud traditionnelle sera confrontée à une série de défis en raison de la centralisation de calcul, du stockage et de la mise en réseau dans un petit nombre de centres de données, et de la longue distance entre les appareils connectés et les centres de données distants.

Pour répondre à ce besoin, l'edge computing s'appuie sur un modèle dans lequel les ressources de calcul sont distribuées dans le edge de réseau selon les besoins, tout en décentralisant le traitement des données du cloud vers le edge autant que possible. Ainsi, il est possible d'avoir rapidement des informations exploitables basées sur des données qui varient dans le temps.

Dans cette thèse, nous proposons de nouveaux modèles d'optimisation pour optimiser l'utilisation des ressources dans le edge de réseau pour deux domaines de recherche de l'edge computing, le "service offloading" et "vehicular edge computing". Nous étudions différents cas d'utilisation dans chaque domaine de recherche. Pour les solutions optimales, Premièrement, pour le "service offloading", nous proposons des algorithmes optimaux pour le placement des services dans les serveurs edge (Tasks, Virtual Network Functions (VNF), Service Function Chain (SFC)) en tenant compte des contraintes de ressources de calcul. De plus, pour "vehicular edge computing", nous proposons des modèles exacts liés à la maximisation de la couverture des véhicules par les taxis et les Unmanned Aerial Vehicle (UAV) pour les applications de streaming vidéo en ligne. De plus, nous proposons un edge-autopilot VNFs offloading dans le edge de réseau pour la conduite autonome. Les résultats de l'évaluation montrent l'efficacité des algorithmes proposés dans les réseaux avec un nombre limité d'appareils en termes de temps, de coût et d'utilisation des ressources.

Pour faire face aux réseaux denses avec un nombre élevé d'appareils et des problèmes d'évolutivité, nous proposons des algorithmes à grande échelle qui prennent en charge une énorme quantité d'appareils, de données et de demandes d'utilisateurs. Des algorithmes heuristiques sont proposés pour l'orchestration SFC, couverture maximale des serveurs edge mobiles (véhicules). De plus, les algorithmes d'intelligence artificielle (apprentissage automatique, apprentissage en profondeur et apprentissage par renforcement en profondeur) sont utilisés pour le placement des "5G VNF slices", le placement des "VNF-autopilot" et la navigation autonome des drones. Les résultats numériques donnent de bons résultats par rapport aux algorithmes exacts avec haute efficacité en temps.

Mots clés: Internet des objets (IoT), Cloud Computing, Edge/Fog Computing, Mobile Edge Computing, Intelligence Artificielle, Optimisation.

Thesis Publications

Journals

1. Mohammed Laroui, Hatem Ibn-Khedher, Moussa Ali Cherif, Hassine Moungla, Hossam Afifi, and Ahmed E Kamel. SO-VMEC: Service offloading in virtual mobile edge computing using deep reinforcement learning. *Transactions on Emerging Telecommunications Technologies (ETT), e4211, 2021. [1]*

2. Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa Ali Cherif, Hossam Afifi, and Mohsen Guizani. Edge and Fog Computing for IoT: A Survey on Current Research Activities & Future Directions. *Computer Communications (ComCom), 2021. [2]*

Conferences

1. Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Autonomous UAV Aided Vehicular Edge Computing for Service Offering. In *IEEE Global Communications Conference (Globecom), 7-11 December 2021, Madrid, Spain. [3]*

2. Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Artificial Intelligence Approach for Service Function Chains Orchestration at The Network Edge. In *IEEE International Conference on Communications (ICC), (pp.1-6), 14-23 June 2021, Montreal, QC, Canada. [4]*

3. Hatem Ibn-Khedher, Mohammed Laroui, Mouna Ben Mabrouk, Hassine Moungla, Hossam Afifi, Alberto Nai Oleari, and Ahmed E Kamal. Edge Computing Assisted Autonomous Driving Using Artificial Intelligence. In *IEEE International Wireless Communications and Mobile Computing (IWCMC), (pp.254-259), 28 June-2 July 2021, Harbin City, China. [5]*

4. Mohammed Laroui, Hatem Ibn Khedher, Hassine Moungla, Hossam Afifi, and Ahmed E Kamal. Virtual Mobile Edge Computing Based on IoT Devices Resources in Smart

Cities. In *IEEE International Conference on Communications (ICC), (pp.1-6), 7-11 June 2020, Dublin, Ireland. [6]*

5. Mohammed Laroui, Moussa Ali Cherif, Hatem Ibn Khedher, Hassine Moungla, and Hossam Afifi. Scalable and Cost Efficient Resource Allocation Algorithms Using Deep Reinforcement Learning. In *IEEE International Wireless Communications and Mobile Computing (IWCMC), (pp.946-951), 15-19 June 2020, Limassol, Cyprus. [7]*

6. Mohammed Laroui, Boubakr Nour, Hassine Moungla, Hossam Afifi, and Moussa Ali Cherif. Mobile Vehicular Edge Computing Architecture using Rideshare Taxis as a Mobile Edge Server. In *IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), (pp.1-2), 10-13 January 2020, Las Vegas, NV, USA. [8]*

7. Mohammed Laroui, Aicha Dridi, Hossam Afifi, Hassine Moungla, Michel Marot, and Moussa Ali Cherif. Energy Management For Electric Vehicles in Smart Cities: A Deep Learning Approach. In *IEEE International Wireless Communications and Mobile Computing (IWCMC), (pp.2080-2085), 24-28 June 2019, Tangier, Morocco. [9]*

8. Mohammed Laroui, Akrem Sellami, Boubakr Nour, Hassine Moungla, Hossam Afifi, Sofiane Boukli Hacene. Driving Path Stability in VANETs. In *IEEE Global Communications Conference (GLOBECOM), (pp.1-6), 9-13 December 2018, Abu Dhabi, United Arab Emirates. [10]*

Table of contents

List of figures

List of tables

Chapter 1

General Introduction

1.1 Motivation

The Internet of Things (IoT) has been a huge factor in the development of the high-tech industry. With the large number of devices and profits to be made over the coming years, the IoT will have a profound and dominant impact on the high-tech industry in general and the semiconductor industry in particular. IoT devices can actually be any type of sensors and chips with different capabilities made by different manufacturers, and there are many applications that can be built to enable smart cities, smart transportation, smart homes, and smart healthcare.

In the IoT, connected devices can generate an enormous amount of data at very high speeds and some applications can require very low latency. The data is directly sent to the cloud for storage and processing in the data centers. Traditional cloud infrastructure will face a series of challenges due to the centralization of storage, computing, and networking in a small number of data centers, and the long-distance between connected devices and remote data centers.

To meet this challenge, edge computing seems to be a promising technology that provides computing resources closer to IoT devices.

In this thesis, we are motivated to identify and troubleshoot problems of edge computing architecture and the integration of this architecture with IoT applications. Further, the

literature lacks general edge-IoT architectures, this leads us to design recent architectures, especially for service offloading and vehicular edge computing.

Furthermore, despite the optimization tasks importance, such efficient functions are missing in the global edge-IoT architectures.
For this, we propose novel different optimization models (exact and large-scale algorithms) at the edge layer that solve both service offloading problems related to placement at the network edge, and vehicular edge computing problems that uses vehicles as mobile edge servers to provide services to end-users such as caching, computing, and streaming.

1.2 Contribution

Our contribution presents new optimization algorithms to improve the quality of service in the Internet of things based on edge computing networks to achieve maximum reliability and efficiency for both service offloading and vehicular edge computing.
First, for service offloading, we propose optimal placement models at the network edge for end-users tasks (ie. Optimal Virtual Edge nodes Placement Algorithm (VEnPA)), VNFs slices (ie. Service Offloading in Virtual Mobile Edge Computing (SO-VMEC)), and SFCs (ie. Optimal SFC Placement in IoT-VMEC (OSPV)). Then, for vehicular edge computing, we propose optimal model for mobile edge servers (Taxis and UAV) coverage for video streaming application (ie. Mobile Vehicular Edge Computing (MVEC), Optimal Follow Me UAV (OFMU)), and Edge autopilot for autonomous driving (ie. Optimal Virtual Edge-Autopilot Placement (OVEAP)).
Further, to cope with the high complexity problems of optimal algorithms in dense networks, we propose large scale algorithms for both service offloading and vehicular edge computing use cases. The artificial intelligence algorithms reinforcement learning and deep reinforcement learning are used for the placement of end-users tasks and VNFs slices (Q-learning and Deep Q-learning (DQN)), while an Efficient SFC Placement algorithm (ESPV) based on the Bin Packing model is implemented for SFCs placement in virtual edge servers.
Moreover, for vehicular edge computing, we propose different heuristic algorithms to cover the high requirement in dense networks. For the first use case, to maximize the client vehicles coverage using mobile edge server (Taxis) we propose an heuristic algorithm that select

only the taxis that share the same path segment(s) with the client vehicle to decrease the complexity of the algorithm. For the second use case, we propose heuristic algorithm for FMU (AFMU) based on the Weighted Set Cover (WSC) model to increase the UAV availability during the service. Finally, for the last use case, we propose Efficient Edge Autopilot Placement (DVEAP) based on deep reinforcement learning to place efficiently the autopilot chains in the edge server.

To assess the efficiency of the proposed algorithms for both small scale and large scale networks we use different tools including programming language, platforms for artificial intelligence models, and simulators for both networking and computing.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter (2) consists of two parts, first, we overview the Internet of things (IoT) and the use of cloud to provide computing services to IoT devices. Moreover, because of the latency issue of the cloud especially for real-time applications, we introduce the edge computing technology with its main applications and use cases. Then, we present the related works proposed by the research community for both service offloading and vehicular edge computing to show the main contributions and enhancements to place efficiently the services in the edge servers, and maintain the high availability of the services in a vehicular environment. Then, in the chapter (3), we present the optimal algorithms for both service offloading (VEnPA, SO-VMEC, OSPV), and vehicular edge computing (MVEC, OFMU, OVEAP) with its variables, constraints and objective function. Chapter (4) outlines the large-scale solutions for both service offloading (Q-learning, DQN, ESPV), and vehicular edge computing (Maximum coverage, AFMU, DVEAP). Finally, chapter (5) present the conclusion and the future topics and issues for edge computing networks.

Chapter 2

Edge Computing For IoT: Overview & Related Works

The rapid increase of smart devices in recent years and the change of applications' requirements led to the innovation of current network architecture to satisfy the needs of user applications to allow fast data processes and reliable services. Moreover, due to the perpetual emergence of greedy Internet applications and user demands that are challenging communication and computation, Edge Computing has been proposed to overcome cloud computing issues and to bring data computation closer to end-users. The first part of this chapter highlight the evolution of edge computing to cope with IoT applications by analyzing the need of edge servers with the cloud to overcome the existing issues. Then, the second part presents the state of the art for the two research directions that are studied in this thesis, service offloading and vehicular edge computing.

2.1 Edge Computing for IoT: Overview

2.1.1 Internet of Things (IoT)

The development of communication technology across generations and the competition of various international companies in order to provide fast and advanced communication services with the lowest cost led to the rapid evolution of connected vehicles, embedded systems, and mobile devices which resulted in the emergence of a smart world of interconnected devices

that collaborate, collect data, sense, and take decisions without the need to interact with the human. This smart world is called the Internet of things.

Definition

The Internet of things (IoT) [11, 12] allows connecting any things including vehicles, drones, devices, solar panels, etc that exchange data with each other. IoT is one of the important topics of future network technologies, in addition, it has an important interest from companies in different domains [13, 14] such as agriculture, medicine, airlines, energy, and telecommunication. The IoT aims to create a smart world based on different systems of interconnected objects. The different connected objects communicate with each other without the need to interact with a human using different models such as Bluetooth, Zigbee, and Wi-Fi. The main characteristic of IoT is the use of different communication technologies (e.g., identification, tracking, and actuator networks, wireless and wired sensors, etc.) to improve the interaction and the cooperation of various technologies.

An IoT system, for example, a smart home works by sending, analyzing, and receiving data continuously. Depending on the type of IoT system, analyzes can be performed by humans or by artificial intelligence aided by machine learning (AI / ML), in near real-time or over a long time. In the example of the smart home, to predict the optimal time to set the thermostat before return to home, the IoT system can connect to the Google Maps API to obtain real-time traffic modeling in the user area, or use data relating to the usual journeys that the car has collected over a long time. In addition, IoT data collected by each customer's thermostat can be analyzed by energy providers to optimize their services on a larger scale.

From an IT perspective, IoT solutions allow businesses to enhance their existing systems and create new points of connection with customers and partners. However, they also bring their share of new computer problems. A system of connected devices can generate massive amounts of data, which is referred to as "Big Data". However, integrating this Big Data into existing systems and setting up analyzes that allow it to be exploited is a challenge. In addition, security can become a critical issue depending on the degree of openness of the future IoT platform. Despite this, the benefits that the IoT brings to businesses more than outweigh the efforts required to implement it, and businesses in virtually every industry are already using it successfully.

In agriculture, the IoT is revolutionizing the world of agriculture on several levels, especially with the use of humidity sensors. By installing a network of humidity sensors in their fields, farmers are now able to receive more accurate data that allows them to determine the best time to irrigate their crops. The IoT can be taken advantage of further in this case, by connecting humidity sensors to IoT applications that control irrigation systems; Then irrigation is triggered automatically from the data received from the sensors, without any human intervention.

As with all technological progress, users remain wary of the IoT, however, because even if this trend opens up new and very interesting possibilities, it raises questions about the confidentiality and security of its data. So, cannot ignore this aspect when planning to implement an enterprise-wide IoT project, especially when targeting the general public as the end-user.

Figure 2.1 displays the IoT application domains.

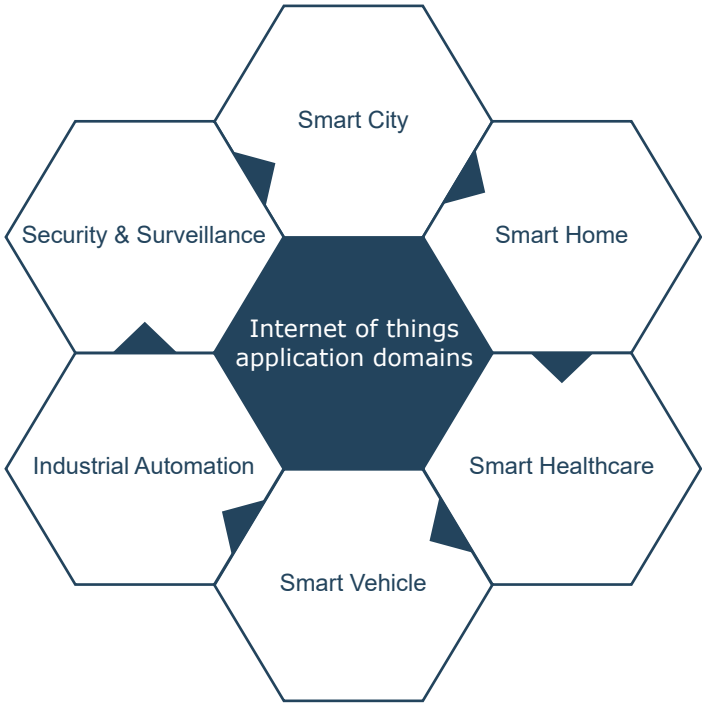


Fig. 2.1 IoT Application Domains

Essential Technologies

The IoT consists of three different components: (a) actuator: is the hardware of sense. (b) storage module and data analytics tools, and (c) visualization and interpretation tools. The following technologies represent the components defined above:

• *Radio Frequency Identification (RFID)* [15]: RFID enables the development and the design of microchips. It allows the numerical identification of anythings (e.g. card, vehicle, smart device, etc.) attached with a barcode. There are two main components of the RFID system, (a) RFID tags: are devices represented as microchips used to send data using wireless transmission. (b) RFID reader recovers the incoming signal from the microchips, is responsible for the object identification. The RFID system is one of the most used devices in IoT applications [16–18] including social applications, smart healthcare, and controlling privacy.

• *Wireless Sensor Networks (WSN)* [19]: is a set of sensors that collect data (e.g., temperature, movement, etc.), process, and transfer the collected information. WSN is composed of the following components: (a) the sensor (unit of capture) is responsible for data collecting from an environment as signals, and transfer it into digital data. (b) processing unit: allows the analysis of the captured data. (c) transmission unit: responsible for data transmission/reception, and (d) energy control unit: represent an essential part of the WSN system, responsible for optimal energy distribution to the other modules. WSN is used in different domains including medical application (e.g., remote medical monitoring), military, commercial, and environmental monitoring (e.g., forest fires, landslides, pollution, etc).

• *Data Storage and Analytics* [20]: the use of sensors and connect the different objects in the IoT network lead to generating a massive amount of data that require a huge storage capacity. Because of this reason, the data storage is an important issue in the IoT network. To overcome this issue, different solutions have been proposed to provide efficient data storage and analytic. Moreover, in the last years the use of the cloud is become more attractive and popular, where nowadays the use of cloud for data analytics and storage is mostly preferred because it provide huge storage capacity as well as high computing resources for data analytics.

Architecture

Different IoT architectures have been proposed for generic and specific use cases. In general, the basic IoT architectures consists of three parts [21] as follow: (a) perception layer: includes RFIDs, sensors, cameras, etc., (b) network layer: is responsible for transmitting the data collected from the perception layer, and (c) application layer that represents the application such as smart home and smart healthcare. Figure 2.2 display the 3-tier IoT architecture. The rapid development of IoT networks with the new generation of applications led to generate a huge amount of data that can not be processed in both the source devices and the access points because of it's computing requirement that exceeds the existing computing limit. For this, a 2-tier architecture called cloud computing has been deployed to process data in the cloud that offer a high capacity for storage and computing.

2.1.2 Cloud Computing (CC)

The rapid development of the new generation of applications the IoT led to new challenges for data processing and storage to satisfy the resources required for these applications. Cloud computing is an innovative technology that provides high computing resources for IoT applications including fast processing and huge storage capacity of data.

Overview

The use of computers and smartphones has dramatically increased in the last years. This growth led to the need for efficient architectures to provide the required processing, especially for data processing and storage. Moreover, support the increase of devices as well as the development of applications. Cloud computing [22, 23] is an innovative technology that allows providing a centralized environment for data processing including high computing resources and huge data storage capacity. Different types of cloud have been deployed [24] including public, private, community, and hybrid. The public cloud offers different services to connected users on the Internet. On the contrary, the private cloud provides services to specific organizations. Where the community cloud allows to offers services to a group of companies. The last type is the hybrid cloud that offers services where it's a good choice for companies because of the balance between both resource control and the utilization cost.

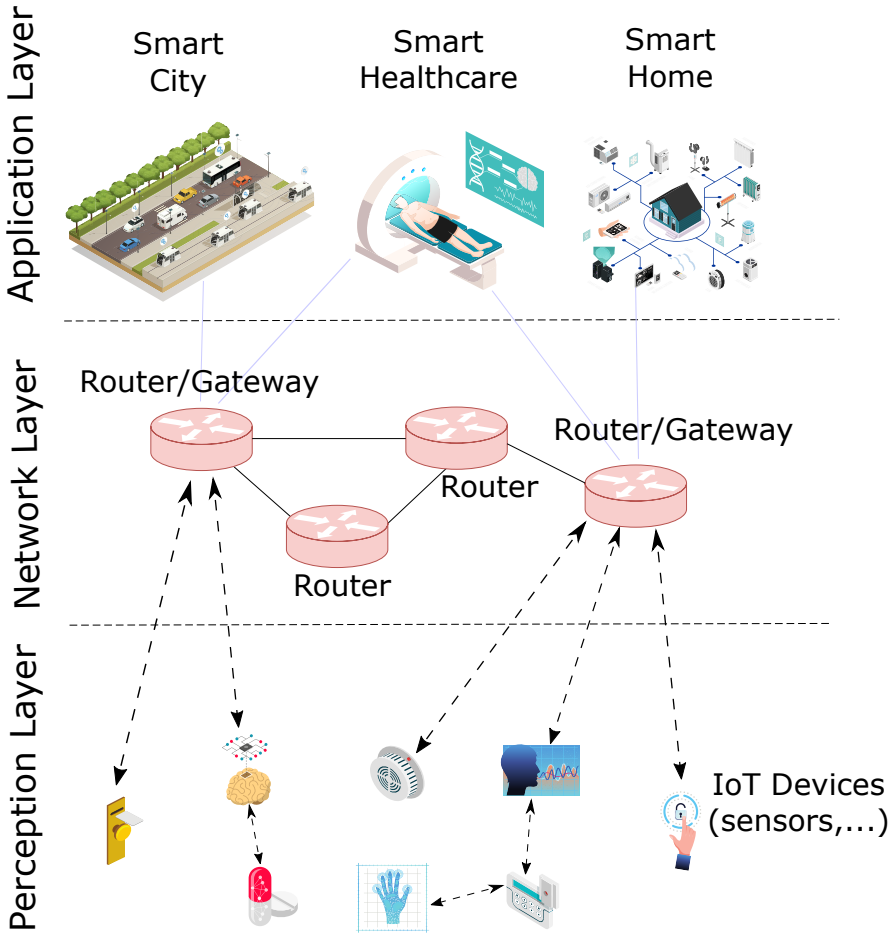


Fig. 2.2 3-Tiers IoT Architecture.

Cloud Computing Deployment Models

1. Public Cloud:

The public cloud is a fully connected network of high computing resources accessible to the general public users via the Internet. It allows the storage and processing of data, but also the development of applications and various technologies (Artificial Intelligence, Machine Learning, etc.). Public cloud services are provided by service providers, besides, these services can be free or billed peruse. The largest providers offer many services on their public cloud platforms, to guarantee their users an ever richer experience and an ever greater variety of products.

The public cloud has several advantages: ease of configuration and use, great flexibility, quick adaptation to business needs, and pay-as-you-go according to user needs.

2. Private Cloud:

The private cloud is a storage or data processing space only accessible by its owner, whether he is a single customer or a group of companies. Only users linked to this client can access the data and applications developed in this private area.

This configuration has several advantages: increased confidentiality, better control of the stored data, and a feeling of security.

3. Hybrid Cloud:

The hybrid cloud generally describes a cross form between a public and a private cloud. As a result, some of the applications and data processed on company servers and some on the servers of a dedicated provider. Ideally, a hybrid cloud combines both private and public clouds together in a transparent and symbiotic fashion. The company alone decides which type of data kept locally and the data sent to the cloud servers, for example, preserve the data protection files locally and store the remaining data on the cloud servers. while some other companies prefer to outsource only cloud computing and keep full storage locally. Or conversely, storing the data in the cloud to allow remote access, and use the computing power locally.

Cloud Computing Service Models

The cloud provides different services categorized by the type of service. The three categories of cloud services can be defined as follows:

1. Software as a service (SaaS): [25] is an online service in the cloud allowing developers to create their applications via the Internet without the need to install any software on their local computer. When users use SaaS, the data is permanently stored on the cloud. In addition, users do not have the possibility to control or manage the network, storage of data and the operating system used. SaaS has different advantages for both service providers and end users. Service providers enjoy maintenance, installation, and control of versions, while the end users enjoy access the services anywhere and anytime. Also, collaborate and share data becomes more easily.

2. Platform as a service (PaaS): [26] this type provides users the control of different resources in the cloud over the Internet including network capacity, operating systems, hardware and storage management, etc,. The PaaS offers a virtualized environment for users to design, deploy, and run their applications without the need to extend the capacity of the resources in their computers. This platform consists of software that includes development tools, middleware and databases. The main advantages of PaaS are: manage and update regularly the operating system features, provide an efficient scalability, offer quick development of softwares and applications, and provide efficient and better technology to businesses.

3. Infrastructure as a Service (IaaS): [27] offers to users the possibility to control and manage both hardware and software in the cloud. Moreover, IaaS provides online access to platforms and applications anywhere, using any connected device from any network, provides virtual infrastructure, load balancing, as well as a large capacity for computing. The storage services provide a large amount of data storage and retrieval services from anywhere using any device. In addition, provides security and compliance features that meet the most stringent data security requirements. Moreover, offers many advantages such as managing data with optimized costs. The computing services offers a resizable computing capacity in the cloud. It is designed to offers developers easier services such as a large capacity of computing and tools to build different applications, In addition, it guarantees flexibility, security, and a reliable environment. The use of IaaS has different benefits where users can access applications and platform (that are always available) from anywhere, using any device, and from any network, provide virtual infrastructure (i.e. network virtualization, storage, and servers), as well as provide services of load balancing and a large capacity for computing.

Figure 2.3 display the three categories of cloud computing services.

CC for Internet of Things

Both IoT and cloud computing are two innovative technologies that offer various services to our daily life. The cloud can serve the IoT technology by providing high computing resources for processing and a huge capacity for storage, that represent the main requirements for IoT to support the new generation of applications. For this, the CloudIoT paradigm is the merge of the two technologies into one architecture that connects the IoT devices directly with the cloud. Botta *et al.* [28] presented an integration of IoT with the cloud by providing services

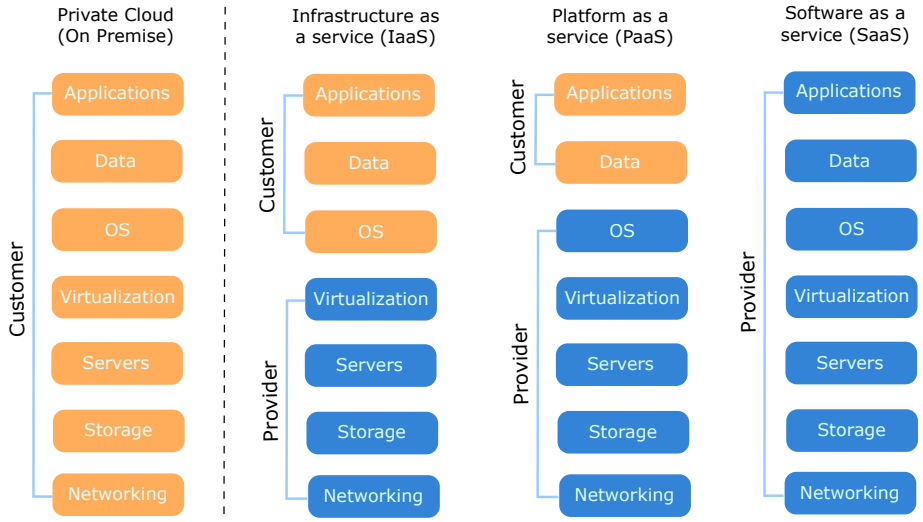


Fig. 2.3 Categories of Cloud Computing Services.

to IoT such as computing resources and data storage. Similarly, Neagu *et al.* [29] proposed a monitoring service for healthcare that offers different medical facilities by using cloud-IoT architecture. Ismail *et al.* [30] presented a study for task scheduling algorithm and virtual machine placement in Cloud-IoT architecture where they focused on the consumed energy in data centers. In the CloudIoT system, the devices connect directly with the cloud over the Internet and start exchanging data where the cloud provides processing resources and storage capacity to the end-user devices which form an interconnected system represent the IoT network. Figure 2.4 outline the CloudIoT architecture.

2.1.3 Mobile Cloud Computing (MCC)

Despite the rapid increase in mobile computing usage, it is difficult to exploit all its resources because of its inherent problems including frequent disconnections, mobility, and resource scarcity. Mobile cloud computing can resolve these issues by using providers' resources to execute mobile applications outside to the end-user devices (mobile devices).

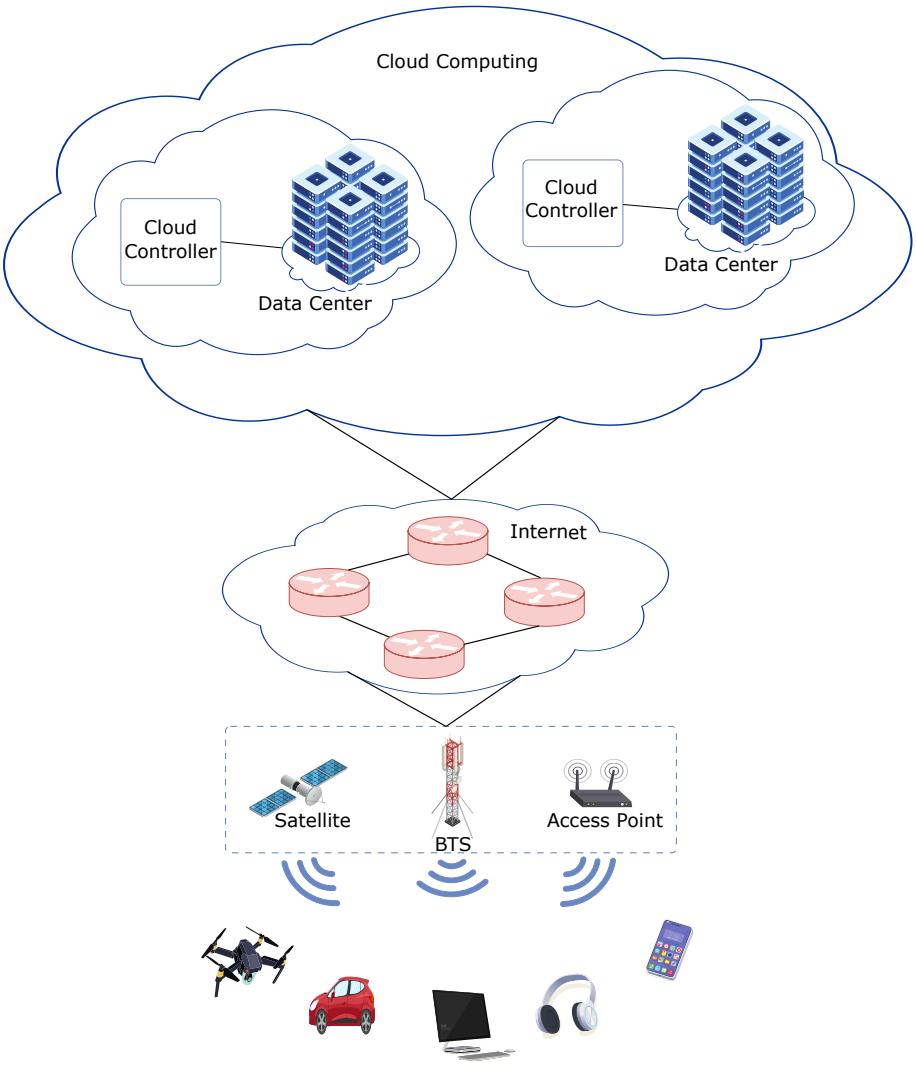


Fig. 2.4 Cloud Computing Architecture for IoT.

Overview

In a time of profound change in the computing field, the main developments have been on both mobile Internet and cloud computing. These two technologies overlap in mobile cloud computing.

The mobile cloud computing (MCC) [31–33] is a technology that combines both mobile computing and cloud computing to bring rich computational resources to network operators, cloud computing providers as well as mobile users. In MCC technology, data storage, and data processing happen outside of end-user devices (mobile devices) which offer many benefits such as extend battery life, improved scalability, and reliability, improve processing

power and data storage capacity, as well as access to data from anywhere. MCC is also widespread within companies by making cloud services available to mobile personnel who can access them in the field or on the move.

Architecture

Based on the general concept of MCC [34], figure 2.5 display the typical architecture of MCC. From the figure, the mobile devices (smartphones, laptops,...) are connected to the base station (satellite, access point, base transceiver station, etc) that allows to create the communication link, control the connection state, and functional interfaces between the mobile devices and the network. The requests of mobile users are transmitted to the mobile network that provides local services such as authorization, accounting, and authentication. For computing requests, the users demands are forwarded from the mobile network to the cloud over the Internet. In the cloud, the controllers process the users requests to provide services including computing, database servers, web applications, and virtualization, etc.

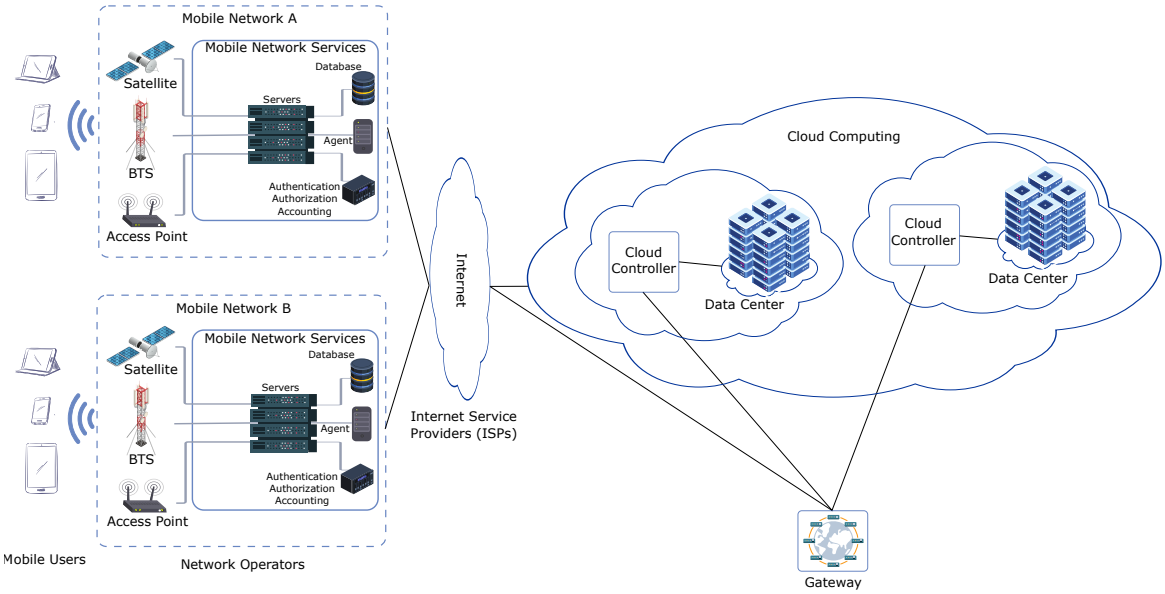


Fig. 2.5 Mobile Cloud Computing Architecture.

The centralized architecture of the cloud is not efficient to process large size of data generated by the IoT devices that require a short response time. To overcome such issues, an alternative paradigm namely edge computing has been used that processes the data in connected devices or the local gateways.

2.1.4 Edge Computing (EC)

In the incoming years, the number of connected devices in the IoT network will increase rapidly [35], this lead to generate a huge quantity of data [36]. Moreover, the new generation of applications such as online gaming, video streaming,.., etc. requires a very short response time to satisfy the quality of service requirements and provide efficient service to end-users. In addition, energy consumption in wireless communication is an important issue because of the limited capacity of the battery in IoT devices, which by consequence lead to the impossibility of processing in the local device. Thus, the centralized platform does not satisfy the IoT applications requirements by providing computation, storage, and networking resources in the cloud owned by companies that require payment for any service. Moreover, the huge amount of data sent to the cloud need to be processed in a short time which can not be provided in a centralized architecture.

Overview

The edge computing [37, 38] paradigm aims to be an innovative computing solution for future networks. Thus, it allows solving critical issues including computation-based and time-constrained applications. The main benefits of processing data at the edge of the network are: reduce the energy consumption of mobile devices, reduce communication latency and network load, give the new inventors chances to help future network innovations, provide security, privacy, and reliability, as well as eliminating the congestion within the network core.

Concepts & Relationships

To cope with the ambiguity related to the difference between fog computing and edge computing, we describe both fog and edge computing from different perspectives.

In the research community, some researchers considered edge computing and fog computing as the same technology with differences only in their names [39]. While other researchers defined the two as different concepts. Chiang *et al.* [40] said the edge refers to the edge network, with equipment such as edge routers, home gateways, and base stations. While fog computing is an end-to-end platform that distributes the computing, storage, control, and

networking function closer to end-user devices over the cloud-to-things. On the other hand, Goscinski *et al.* [41] claims that fog computing is located out between the edge cloud and the cloud, and thereby include the edge, while edge computing focuses on the processing of data at the edge. Moreover, Pan *et al.* [42] said that edge computing pushes applications, services, and data from the core to the network edge, based on the core-edge topology [36, 43]. The smart city, video analytics, cloud offloading and smart home are examples of edge computing applications. While fog computing is a background of the IoT, it extends cloud computing and different services to the devices such as multiplexers routers, switches, etc.

In a nutshell, both fog and edge computing have the same research topics and both of them aim at providing the computing services closed to end-users by the decentralization of data processing from the cloud to the edge of the network.

Architecture

The fog/edge layer is located between the end-users and the cloud, as is shown in the figure 2.6, is composed of the following components [44]:

(a) *Authentication and Authorization*: identify the access policies and the control roles, (b) *Offloading Management*: defines the manner to design an optimal offloading scheme, the partition for offloading, and the types of information in the offloading process, (c) *Location Services*: allows learning the mobility model by mapping the physical locations with the network, (d) *System Monitor*: offer information such as energy, usage, and workload to the other components, (e) *Resource Management*: is responsible for resource allocation and discovery, dynamic joining and leaving of the fog node, provisioning, and maintaining of resources, and (f) *VM Scheduling*: that aims to provide an optimal strategy and management for virtual machine scheduling.

EC for Internet of Things

The edge offers networking services, computation, and storage to end-users in IoT networks [45]. For example, an intelligent surveillance system doesn't need to send all recorded data to the cloud, the facial recognition and movement detection algorithms running on the fog/edge layer, hence saving bandwidth and storage space. So, the fog/edge will be the best option for applications that require temporary storage. In addition, the fog/edge can play an important

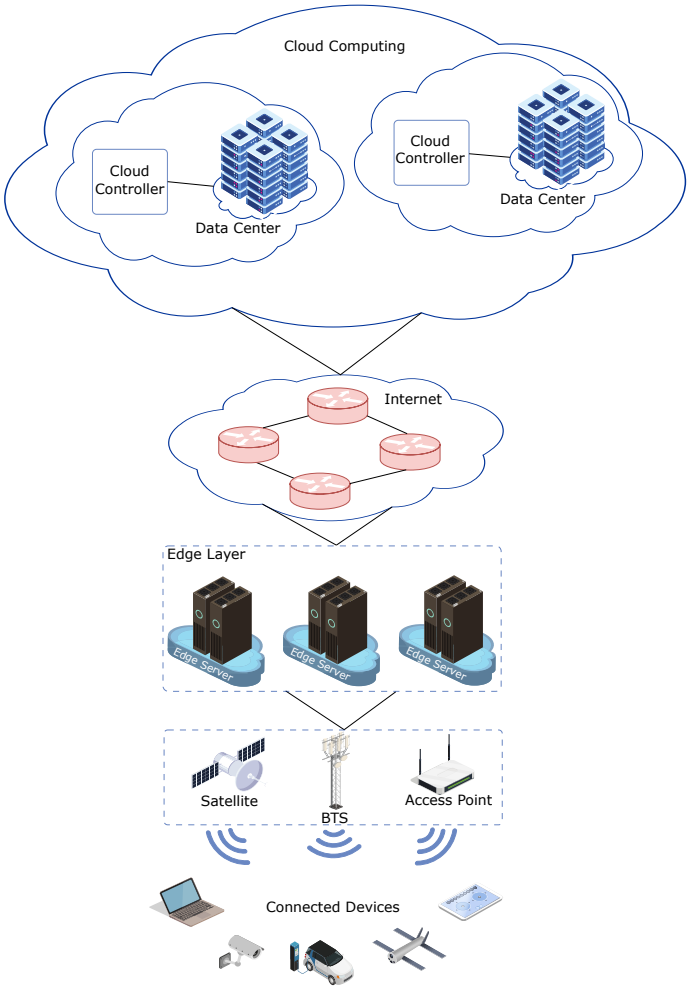


Fig. 2.6 Edge Computing Architecture for IoT.

role in the future, where the wireless sensors networks and IoT present with the integration of heterogeneous devices and protocols for enhancing services [46]. The edge computing is an appropriate architecture for the current and future Internet in general and the IoT networks in particular in which this distributed architecture may provide high-level communication and efficiency in the edge of the network rather than at the cloud-level. It supports a wide range of today's IoT applications, especially those that require a short response time.

Edge Computing for IoT: Use Cases & Applications

The edge computing offers an efficient platform for current and future IoT applications including smart cities, smart homes, smart grid, smart healthcare, etc. Different applications have been proposed for IoT based on edge computing as follow:

(a) Smart City Systems: A smart city [47] is an urban area composed of connected sectors that pull together to provide services to users through the data analysis that is collected from different sources. Smart cities aim to decrease traffic congestion and energy consumption which contribute to improving the quality of life.

Different technologies are used in smart cities that will create an important economic market which by consequence that reflected on the economic development of countries. Though, different issues and challenges can still face the deployment of smart cities such as large-scale sensing networks deployed in a large geographic area, big data analysis, cooperative communication in a wireless network, and energy and mobility management in intelligent traffic systems.

Smart cities require intelligent and efficient data monitoring and analysis in order to achieve an automated and fast decision without any human intervention [48]. Hereby, ensure people's safety and the reliability of components. Furthermore, smartness has some requirements such as efficient platforms with advanced algorithms including artificial intelligence models (machine learning [49], deep learning [50], and deep reinforcement learning [51]).

The edge computing technology is an efficient platform for smart cities [52], due to its architecture, the process of data will happen at the edge of the network instead of the cloud [36] and enhance both the network delay and the user experience. This is an important and essential element to build reliable and efficient smart cities that can manage the following challenges:

• *Huge Quantity of Data*: A huge amount of data will be generated by future applications in smart cities varies from smart traffic lights, utility, transport, etc. The process of all this data in the cloud is inefficient because of the long distance between the cloud and end-users which takes more time for data transfer. For this, the process of data at the edge of the network close to end-users will resolve the issues of data processing in the centralized cloud by offering an efficient processing service that is required by current and future applications in smart cities.

• *Low Latency*: The latency requirement is one of the most important issues in smart cities, where most applications such as health emergencies and public safety require a low latency to provide efficient services to end-users and enhance the quality of experience. The edge computing provide a promising platform for smart cities, it could decrease the time for data transmission and organize the network structure. Both pre-processing and decision could be made in the edge layer which lead to decrease the latency compared to the use of the centralized cloud only that is far from end-users which increase the latency [36].

• *Location Awareness*: Most smart cities applications such as intelligent transportation systems and utility management are geographic-based applications in nature. The edge computing provide efficient platform for the location awareness in smart cities by collecting the data based on geographic sites which increase the QoS and prevents the data access delay, as well as providing the transparency of the transferred information [36].

(b) Smart Home Systems: The smart home [53] allows controlling intelligent smart devices inside homes including fridge, cooker, air-conditioner, TV, etc. The smart services [54] inside a smart home can be classified into three categories:

• *Home Management Services*: such as smart control of connected devices such as cooker and TV.

• *Home Automation Services*: allows the automation of smart devices such as air-conditioners, cleaners, etc. And lastly,

• *Home Security Services*: such as detection of potential crimes, prevention from gas leakage, and explosions.

The connected devices inside a smart home generate a massive amount of data for decisions and smart control inside the home. The process and the analysis of the generated data require many resources and storage which need an efficient architecture to ensure the continuity and the quality of service. The edge computing provides a distributed architecture for smart home systems [55, 56] by processing data at the network edge in an efficient way without any degradation which provides: efficient data processing, low latency, and less energy consumption.

The edge computing platform aided smart home systems aims to provide an efficient architecture for future IoT applications, especially real-time applications that require short

response time such as online streaming [57] and intelligent surveillance inside the home [58], etc.

(c) Smart Transportation Systems & Vehicular Networks: A smart vehicle system [59, 60] consists of a connected vehicle embedded with devices for computing and storage that allow the vehicle to send and receive data from other vehicles which is known as a vehicle-to-vehicle network (V2V) also with roadside units known as vehicle-to-infrastructure (V2I), and communication with other connected devices known as vehicle-to-everything (V2X). The edge computing provides an efficient platform for smart vehicles system, it can support both the interactions among vehicles, the high mobility, collisions and obstacles detection, etc. Different application types can be developed and deployed for smart transportation systems aided edge computing, including:

• *Safety Applications*: This type of application can be used to send warnings and notifications about crashes, traffic violations, pre-crash sensing, as well as they could include sensing of approaching emergency vehicles.

• *Convenience Applications*: include advice on congestion situations [61], personal routing, in addition to some incidents such as power failure and network breakdown. In some scenarios, the smart vehicle can provide SOS and emergency calls, as well as can play important role in the monitoring of weather and road by sharing information from vehicle sensors.

• *Smart Traffic Lights*: smart lights synchronize the smart vehicles by sending warning messages using wireless networks to share traffic data which can help vehicles in different situations such as warning messages when approaching the pedestrian corridor, as well as the different problems related to the traffic light [62].

• *Smart Parking Systems*: Most big cities face traffic congestion, consequently, finding an empty place in parking is expensive and difficult [63]. To overcome this issue, it is important to build smart parking connected directly with smart vehicles to automatically check available places inside the parking. The parking system sends notifications messages using a wireless network to inform smart vehicles if there are empty places inside the parking. This aims peoples to gain both time and effortlessly find a place in the parking.

• *Commercial Applications*: represents paid location-based services [64] such as advertisements and entertainment, as well as services of diagnostics for vehicle problems and

updates of social networks. These services are provided by companies by offering special sensors or applications installed in the vehicles which can connect directly to the edge server of the associated company.

The edge computing provides an efficient platform that offers different benefits to smart vehicles and intelligent transportation system. The edge servers are localized nearby to connected vehicles, which provide efficient and rapid service to enhance the quality of service in smart cities.

(d) Smart Grid Systems: [65] composed of smart meters in different locations used to measure the status of electricity distribution. This information is analyzed by a server called SCADA [66], that manage the system by sending information about emergency and responds to different change requests to protect the power grid and stabilize the system.

The edge computing can be integrated with smart grid systems [67] which may offer great services to enhance the efficiency of this smart system. With the edge computing paradigm, the power generators (e.g., wind farms, solar panels, etc.) can be integrated with the main power grid which offers efficient control of the power networks. Moreover, the SCADA can be linked with the edge which can improve the security [68] and the network cost [69].

The edge computing aided smart grid will turn into a hierarchical system (multi-tier architecture) that aims to create interactions among different SCADA [70]. In this system, the edge layer manages the micro-grid, and exchange data and information with neighboring edge servers and the higher tiers.

(e) Smart Healthcare Systems: Remote Patient Monitoring (RPM) [71, 72] aims to create smart system allows remote patients monitoring. This system is composed of three components: data acquisition module, diagnostics module, and visualization module. To get data, the sensors are deployed on patient (e.g., blood glucose sensor). The first step is acquiring the data, then it will be sent to the diagnostic module (processing unit) for processing. And finally, the visualization module display the calculated analytic. In healthcare system [13], the data management is an important issue because the patient data contain private and important information that need to be processed in a secure and an efficient way.

The edge computing system provides efficient platform for smart medical systems which is one of the important technology in the Internet of things ecosystem. The patient data

analysis requires short response time and security, etc., which can be provided only by efficient platform and full distributed system offered by the edge computing architecture.

2.1.5 Mobile Edge Computing (MEC)

The path toward defining the next generation of cellular networks systems encompasses all the requirements of the communication network to cope with massive data generated that needs efficient processing so as to allow operators to offer services and provide content efficiently and profitably. Driven by the visions of cellular networks and the Internet of things, recent years have seen a rapid increase in computing requests due to the development of applications which led to a paradigm shift in mobile computing, from mobile cloud computing to mobile edge computing.

Overview

Mobile edge computing (MEC) [73, 74] also know as Multi-access Edge Computing introduced by the ETSI ISG (European Telecommunications Standards Institute, Industry Specification Group) standardization organization [75, 76]. It provides a combination of computing on a radio access network (RAN) on the network edge. The MEC aims to push storage, mobile computing, and network control to the edge of the network (e.g., access points and base stations) so as to provide resources especially for latency-critical and computation-intensive applications. MEC promises a spectacular reduction in mobile energy consumption and latency, tackling the main challenges for materializing the new generation of cellular networks vision. The promised benefits of MEC have motivated global efforts in both industry and academia on developing and deploying this technology.

MEC Framework & Architecture

The MEC framework (shown in figure 2.7), displays the involved high-level functional entities. These entities are grouped into the network-level, system-level, and host-level entities.

The network-level contains related entities are the external networks, local networks, and the 3rd Generation Partnership Project (3GPP) cellular network. This layer allows

connectivity to cellular networks, local area networks, and external networks such as the Internet.

The ME host-level consists of the management entity and the ME host that also includes the virtualization infrastructure, the ME platform, and ME applications.

The top-level is the ME system-level that has the overall visibility to all ME system. This layer consists of ME management to run applications on the operator network.

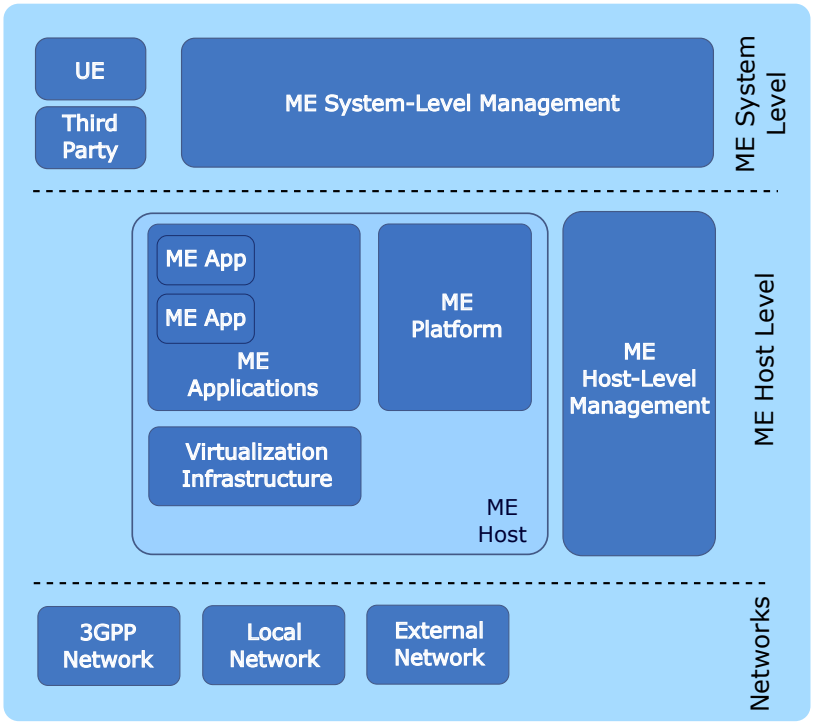


Fig. 2.7 Mobile Edge Computing Framework.

The reference MEC architecture defined in figure 2.8, represents a thorough understanding of the mobile edge (ME) system. It defines in detail the functional entities with the relations between them. The ME entity is a host consisting of the ME platform and the network resources, storage, and computing virtualization infrastructure. Besides, the ME host can offer permanent storage and real-time information related to applications. The data plane included in the virtualized infrastructure routes traffic between the services, networks, and applications, as well as forward the rules received by the ME.

The ME platform consists of a collection of functionalities that are important to run applications on ME and to enable ME applications to advertise, offer, and discover the ME

services. Both the applications and platform provides ME services, while the applications and the platform use ME services.

In the ME platform, a local domain name system (DNS) server/proxy can be configured to direct the user requests to selected ME applications. Many ME platforms can communicate with each other or grouped together to form a communication grid through an Mp3 interface.

On top of the virtualization infrastructure in the ME host, the ME applications are running as virtual machines (VMs), where the applications communicate with the platform through an Mp1 interface to use the services offered by the platform. The Mp1 interface is also used to prepare the relocation for the application state in the case of handover, in addition, indicating the application availability. Each ME application has its own resources or services needed, the system level validates these requirements, then, the selection of ME host(s) is performed based on the availability of application requirements in the host(s).

The ME platform manager (MEPM) consists of ME application lifecycle management, ME application rules management functions (AMF). and ME platform element management (PEM). The ALM provides the application events to the ME orchestrator (MEO). The AMF includes DNS configurations, traffic rules, authorizations, as well as resolving issues in the case of conflict between rules. The reference point between the MEPM and the ME platform is to manage the relocation of applications, configure the rules and the platform, and support the application lifecycle.

Strategic Relevance of MEC

MEC led to an evolution of telecommunications networking, and mobile base stations by shifting the load of the cloud to local servers in the network edge. Which enhancing the quality of experience (QoE) for end-users and helps reduce both latency and congestion on mobile networks. MEC allows applications to benefit from local computing resources, and real-time information related to the network. By deploying various caching content and services near mobile devices, the core of the networks can efficiently serve applications purpose which will act for interesting revenues for the industry including vendors, operators, and third-parties. Use cases include:

1. Connected Autonomous Vehicle Functions

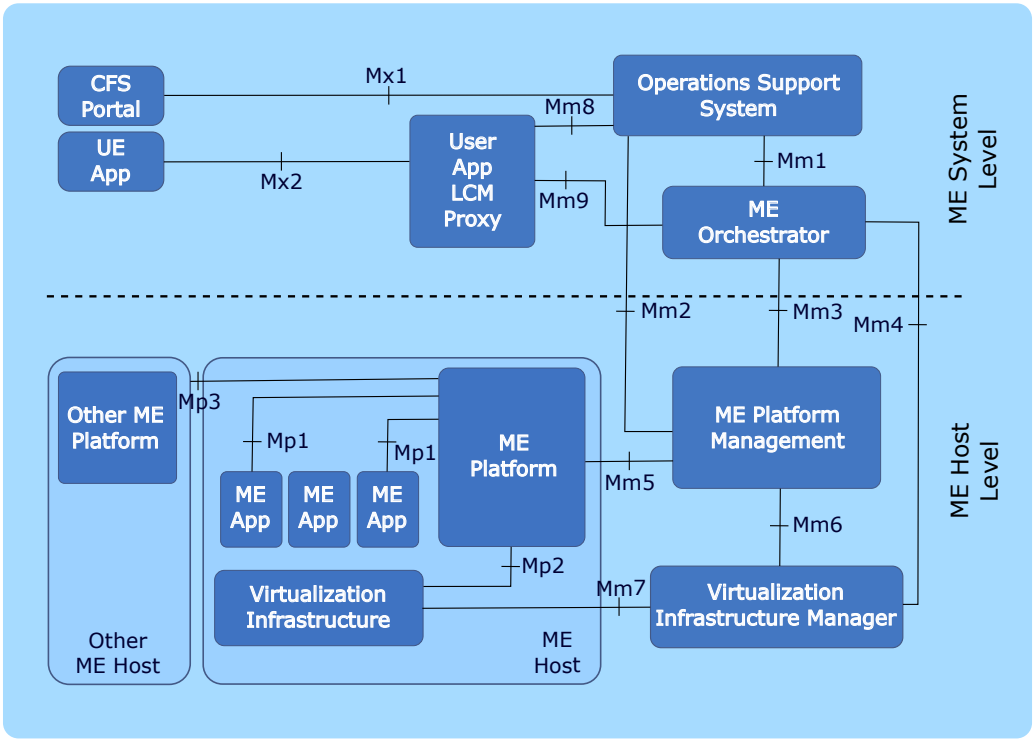


Fig. 2.8 Mobile Edge Computing Architecture.

Used to share information about positions of cars, animals, pedestrians, as well as the weather conditions and the road infrastructure directly to the vehicles. instead of interact with the central cloud, the combination of MEC with AI/ML will allow autonomous vehicles to be real-time synchronized with their surrounding situations and critical changes such as accidents, and incoming emergency vehicles, etc. The MEC provides low latency to autonomous vehicles system in order to operate safely which is important and critical as vehicles cannot wait for the data to be processed in the cloud.

2. Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR)

   The AR/VR can be used by the workers to repair and conduct maintenance tasks remotely. The MEC would provide an efficient platform that offers an overlay of important information related to a particular problem they are required to repair. The workers can fixe the problem on the screen field display on mobile devices. The 3D models require lot of resources because are too heavy so cannot be done in both the

remote cloud and the mobile device, as the latency is high. The MEC allows rendering the broken down device as 3D models of as well as process the data near to end-users (workers) devices, which allows a remote expert to comment the displayed model as well as increase the number of digital models displayed on the screen of the worker device.

Another use case for MR assisted MEC for enterprise application, is a collaboration of multi-user for construction, architecture, and engineering teams. It allows real-time cooperation by rendering 3D models for projects that will be done by the team on jointly in the network edge. This reduces the latency, especially for the 3D models that are often massive files. The MEC system allows easy collaboration between stakeholders, over a distributed network that offer efficient resources near to team devices.

3. Cloud Gaming and Multiplayer Gaming

   With MEC, the intensive graphics/compute processing would be moved from the gaming console to the edge of the network. This allows gamers to play different games with high graphics quality from a near edge server from anywhere and any time without the need to massive dedicated resources in their gaming console. Since the MEC offers a very low latency, cloud gaming will give to audience and developers a wider access to gaming studio and enjoy high gaming experiences, as well as create a potentially new income models for emerging companies in the field of electronic games.

4. Real-time Unmanned Aerial Vehicles (UAV) Detection

   The UAVs are getting a lot of attention from both researchers and developers because of its development in different field such as security, data transmission, packages delivery, etc. There is an important increasing necessity for innovative solutions that can detect when an UAV has entered a geo-fenced zone or a secure zone to trigger alarms to inform the teams that secure the site. As well as, this technology can be used in hospitals and prisons because the respond to threats must be immediately in these locations, which is provided by the MEC system. The use of MEC offers a lower latency for detecting a foreign UAV and get its path to show if its close to an exclusion

zone, where the MEC keeps the UAV data closer to its source, decreasing the taken time to react for any threat or a security breach.

5. Video Analytics

   The video surveillance is one of the most systems used in smart cities such as the surveillance of road, homes, markets, and enterprises, with the massive increase of data volume due to both video quality and the number of cameras. MEC enables data analysis in the edge of the network instead of send the videos traffic to remote central processing. This allows the collect of video streams from different cameras types, as well as, enable analytics of other video applications such as facial recognition. MEC decrease the volume, cost and time it takes to send the video chunks from the camera to the edge server, and allows real-time triggers during the analysis.

Table 2.1 outlines the main characteristics of the cloud computing and mobile cloud computing compared with edge computing and mobile edge computing.

	CC	**MCC**	**EC**	**MEC**
Ownership	Private entities	Private entities, companies, individuals		
Deployment	Network core	Network edge, devices		
Hardware	Servers	Servers, user devices	Heterogeneous servers	
Service	Virtualization			
Net. Architecture	Centralized	N-tiers, decentralized, distributed		
Mobility	Yes			
Latency	Low	Average		
Local awareness	Yes			
Availability	High			
Scalability	Average	High		

Table 2.1 CC & MCC vs EC & MEC

This part has presented an overview of cloud, edge, and fog computing for the Internet of things. First, we detailed the IoT system and architecture. Then, we detailed the cloud

computing technology by presenting the necessary components of the cloud system and mobile cloud computing mechanism. Further, we surveyed the edge computing technology by presenting a detailed overview of edge computing and mobile edge computing for IoT with its innovative strategies and solutions provided to next-generation networking with the different applications in smart cities, smart homes, smart vehicles, and healthcare.

The state of the art will be the following step in the next part to fully understand the problem and to have a detailed idea of the solutions proposed by different researchers around the world.

2.2 Edge Computing for IoT: Related Works

The integration of edge computing with the Internet of things (Edge-IoT) faces many issues and challenges related to QoS. For this, different research directions related to Edge-IoT have been studied to cover all communication models to satisfy the applications' requirements and provide efficient services to end-users. This chapter presents the state of the art for two Edge-IoT research directions, service offloading and vehicular edge computing.

2.2.1 Service Offloading In EC

Service offloading refers to uploading some computationally intensive services to the edge servers for processing. Resource allocation for these uploads refers to allocating some computing resources on edge servers depending on the requirements of the service. Moving services to the network edge facilitates storage, service delivery, content caching, and IoT management, which allows for improving response times and transfer rates, in this way ensuring that users have the fastest and the best service possible. Service offloading is an important network optimization task that still painstakingly tune heuristics to get sufficient solution. These algorithms use data as input and output near-optimal solutions. Service offloading in this novel concept is a complex task. It has to design efficient strategies to optimize the service offloading and allocation decisions. Despite the huge effort in ETSI research group, service offloading functionalities still suffer from the legacy heuristics used to design optimal resource management (orchestration, placement, scheduling, etc.) algorithms

and place concurrent slices within virtual network operators (VNO). In the following, we present a review of the existing solutions for service offloading in edge computing.

Miluzzo *et al.* [77] proposed mClouds, a mobile computing architecture where submitted tasks are executed on smart mobile devices that form a cooperative computing platform. Although he discussed the different steps of tasks execution and mClouds management, the authors ignored the mobility impact which is missed in their work. Moreover, both problem formulation and evaluation results are not presented in this work.

Fahim *et al.* [78] proposed a task offloading environment to execute tasks in mobile devices. Real mobile phones are used to validate the proposed system. The evaluation results prove the efficiency of the proposed system where the potential gain is high in both energy consumption and processing time. However, the proposed system and its formulation is limited to small network scale. Moreover, the impact of device mobility is not studied in this work.

Van *et al.* [79] proposed a workload offloading based on Markov decision process (MDP). The mobile cloudlets are used to process tasks by considering the impact of channel properties, distance and cloudlets mobility. The obtained results show that the efficiency of the proposed work. However, the formulation should be enhanced with energy and mobility parameters to model the transition behavior of the MDP algorithm.

Hasan *et al.* [80] proposed Aura; a mobile adhoc cloud (MAC) computing architecture for IoT task offloading. The framework consists of three main components: *i*) Mobile Agents (MA) that includes smartphones and other end-user smart devices. They are used to run the tasks that are needed to be offloaded and executed in the Aura. *ii*) IoT devices that form a part of the global Aura architecture. It forms the ad-hoc cloud, and *iii*) Controllers that form the link between IoT devices and MA. They ease the creation and the distribution of sub-tasks, the programmability of IoT devices, the MA action selection strategies, etc. It is worth mentioning that the obtained results demonstrate the efficiency of the Aura architecture in MAC cloud computing fields in terms of memory/CPU usage, cost of execution, and task completion time. However, the impact of IoT devices mobility (and energy) is missed in their work.

Van *et al.* [81] proposed a deep reinforcement learning (DRL) approach for tasks offloading in adhoc mobile cloud environment. The MDP theory is used to find tasks

processing actions. Then, the Deep-Q-Network (DQN) is used to find the efficient solution for the MDP model. The evaluation results show that the proposed model is efficient based on different metrics. However, the prediction of the dynamic parameters used in the MDP formulation is missed in this work.

In [82], the authors proposed an offloading technique in mobile adhoc cloud environment based on constrained Markov decision process (CMDP). An Q-Learning and LP schemes are proposed to find efficient solution of the CMDP model. The LP to find the optimal solution, while the QL is used to receive the offloading decision. The obtained results show the efficiency of the proposed schemes. However, the application of the proposed works in real environment is missed in this work.

Alam *et al.* [83] proposed a computation offloading solution in mobile fog/edge computing. They consider access points and controllers as on-demand fog nodes. A framework based on deep Q-learning is proposed to handle the resource demand for computation offloading. The authors prove the efficiency of the proposed solution in terms of computation offloading cost and service latency. However, the work missed the impact of the IoT devices mobility on the computation offloading decisions. Moreover, we believe that the IoT devices energy should be considered to deal with resource constrained devices.

Zhang *et al.* [84] proposed an optimal task offloading scheme based on deep Q-learning in heterogeneous vehicular edge computing networks. First, the authors used the intrinsic reward variable to minimize the utility of task offloading. Then, in order to guarantee the offloading reliability in the presence of tasks transmission failure, they proposed an adaptive redundant offloading algorithm. The overall approach guarantees the offloading process reliability and enhances the system utility. Further, they used a real traffic trace to evaluate their solution. The analytical results prove the efficiency of the proposed solution in terms of reliable offloading and latency as well as tasks offloading with optimal utility. However, the authors did not provide details about the implementation of the proposed solution, and its applicability in the real-world.

Wang *et al.* [85] proposed an optimal data offloading algorithm to ensure the resource provider privacy and achieve efficient resource usage. In addition, the proposed model takes into account the maximization of the resource buyer benefits and the minimization of resource provider cost. The obtained results prove the feasibility of the proposed solution and

its efficiency to identifying malicious resource providers. However, this work did not provide any details about the adaptation of this solution in a real mobile Adhoc cloud environment. In addition, this work needs to address important issues in the mobile Adhoc cloud such as the resources management of devices and the problem of offloading failure caused by obstacles and non-existing devices.

Lin *et al.* [86] proposed Circa, a framework for code offloading between a set of neighbor mobile devices. They used an indoor positioning system (iBeacons) for transmitting in short-range with lower energy and cost for sharing tasks among devices without the need for a centralized server. in addition, task allocation algorithms are proposed in order to select the set of reliable collaborators from the neighbors' mobile devices and offload efficiency the tasks among the selected devices in a fair fashion. The experiment results show that Circa reduces the execution time and preserving the mobile application performances. However, the impact of obstacles on offloading is not studied in this work. In addition, the communication range is very low which decreases the selected devices which by consequence leads to task failure caused by non-existing devices.

Saha *et al.* [87] developed a mobile device cloud framework where the resourceful and reputed worker devices execute the different application tasks. The user devices request service offloading for their application (set of tasks). After getting the willingness from the worker device, the cloudlet selects the optimal task mapping to minimize the cost and maximize the quality of experience (QoE). The results prove the efficiency of the proposed work. However, the evaluation in real-world large-scale networks is missing in this work.

Roy *et al.* [88] Proposed a task allocation framework for mobile device cloud. In addition, a workers auction mechanism is proposed to select buyer devices. Moreover, authors introduced the federated learning and the multi-weight subjective scheme to check the reliability and the trustworthiness of worker devices'. The simulation results prove the efficiency of this work in terms of buyer utility and QoE. However, the evaluation in dense networks is missing in this work.

Table 2.2 shows the summary of the existing works for service offloading in EC.

Paper	*Main Idea*	*Problem Formulation*	*Edge Position*	*Advantages*	*Limitations*
Miluzzo *et al.* [77]	Tasks execution in smart mobile devices.	Some equations for service payment, etc.	Mobile	Execute tasks for other users.	Long response time.
Fahim *et al.* [78]	Task offloading using mobile devices as computing nodes.	No mathematical formulation is used.	Mobile	Efficient energy consumption and processing time.	The mobility impact is not studied.
Van *et al.* [79]	Tasks processing in mobile cloudlets.	Markov decision process (MDP).	Mobile	Enhance time, and total energy consumption.	The implementation details is missing.
Hasan *et al.* [80]	Aura, a task offloading in mobile ad-hoc cloud architecture.	Equations for cost, time, and resources measurement.	Mobile	Enhance execution cost and time.	The mobility impact is not studied.

Van *et al.* [81]	Tasks offloading in an ad-hoc mobile cloud environment.	- Markov Decision Process (MDP). - Deep-Q-Network (DQN).	Mobile	Enhance energy consumption, payment, and delay.	The evaluation in large-scale networks is missing.
Van *et al.* [82]	Offloading technique in mobile ad-hoc cloud.	Markov decision process.	Mobile	Enhance task loss ratio, average delay, and energy consumption.	The evaluation in real environment is missing.
Alam *et al.* [83]	Computation offloading in mobile edge computing.	- Markov Decision Process (MDP). - Deep Q-learning.	Mobile	- Minimize the computing latency.	The mobility impact is not studied.
Zhang *et al.* [84]	Optimal task offloading in a mobile environment.	Deep Q-learning approach.	Fixed	Reliable and optimal task offloading.	The complexity is not discussed.
Wang *et al.* [85]	Optimal offloading model in mobile Adhoc cloud environment.	Distributed optimization algorithm.	Mobile	Identify malicious resource providers.	The Implementation detail is missing.

Lin *et al.* [86]	Circa, an offloading framework among mobile devices.	Tasks allocation algorithms.	Mobile	Enhance the execution time.	High risk of communication failure.
Saha *et al.* [87]	Mobile device cloud framework.	Optimal model and greedy algorithms for tasks provisioning.	Mixed	Low cost.	The evaluation in large-scale networks is missing.
Roy *et al.* [88]	Task allocation framework for mobile device.	Task allocation algorithms.	Mobile	Efficient both buyer utility and QoE.	The evaluation in large-scale networks is missing.

Table 2.2 Summary of the existing works for service offloading in EC.

Service offloading algorithms in the literature are based on formulating exact models to measure the optimal behavior of such a solution. However, in practice, network operators always design intelligent heuristics to deal with scalability problems in large-scale networks with huge amounts of data. Currently, these heuristics are near-optimal solutions and sometimes are very far from optimal. Further, heuristics are unused when system input (e.g. demand matrix, etc..) changes and gives wrong and fatal errors when unexpected input arrives in the system.

2.2.2 Vehicular Edge Computing

Smart vehicles are considered as mobile devices equipped with sensors, having the capability of data collection, computation, and communication. The information is collected from both

in-vehicle sensors and the external environment. Edge computing can provide an efficient and scalable architecture for vehicular networks by improving data processing and traffic in real-time. We present in the following parts a review of the existing edge and fog-based solutions for vehicular networks.

V2X and EC Convergence

The V2X model allows the passing of information from the connected vehicle to any other connected devices including other vehicles, UAV, end-user devices, edge servers, cloud, etc. Also, it incorporates other types of vehicular communication such as V2N (vehicle-to-network), V2P (vehicle-to-pedestrian), V2D (vehicle-to-device), V2V (vehicle-to-vehicle), V2I (vehicle-to-infrastructure), and V2G (vehicle-to-grid). The main motivations for the V2X model are traffic efficiency, energy savings, and road safety. In the following we present the existing works for V2X assisted Edge computing.

Kim *et al.* [89] presented parking management in a vehicular network based on the Fog-Cloud environment. The proposed model consists of many parking slots, the status of slots inside the parking lot (i.e. reserved or vacant) are sent to the fog server that is located at different areas (e.g., shopping malls, restaurants, banks, etc). The fog server provides information of their managed vacant spaces to the RSUs. At each RSU, the roadside cloud and fog computing communicate with the fog servers aiming to direct drivers to the optimal space. The performance evaluation shows that this approach is reliable for finding a vacant slot. However, this approach requires a scalable fog system.

Zhang *et al.* [90] proposed a task offloading mechanism in a vehicular edge computing framework. They formulated the proposed mechanism as a stackelberg game, and developed a distributed algorithm in order to obtain optimal strategies of the vehicular edge computing servers. In the proposed algorithm, each vehicular edge computing server (VEC) starts selecting the selling price and the amount of resources to purchase from the backup computing server (BCS) where both the price and the resources are limited in an interval. Numerical results show that the proposed mechanism guarantees an optimal delay of the computation tasks. However, this work did not take into account the risk of link failure between vehicles and RSUs.

Zhu *et al.* [91] proposed a solution for task allocation in vehicular fog computing, namely Fog Following Me (Folo). They formulated the task allocation as an optimization problem and solved it using mixed integer linear programming. The Folo composed of two types of fog nodes: stationary fog nodes located with WiFi access points and RSUs, and mobile fog nodes that are mobile edge nodes carried by vehicles. Moreover, there are other types of vehicles, named client vehicles, which generate tasks and send them to the fog nodes. Furthermore, a service zone is used in the Folo system as a service coverage zone inside the city. Although The obtained results show that the proposed solution reduces the latency, the proposed solution for task allocation requires computation resources.

Gillam *et al.* [92] Proposed a distributed computing architecture for off-vehicle and on-vehicle computation to support autonomous and connected vehicles. They suggested a local computation in the edge rather than the central cloud to reduce the latency. The evaluation is missing in this work to validate the efficiency of the proposed architecture.

Caminha *et al.* [93] proposed SensingBus, a smart city architecture using buses and a fog system. The SensingBus architecture contains three levels: (i) sensing level that is composed of sensing nodes installed in buses, allows the gathering of data about the city, registers the geographic coordinates from the GPS; (ii) fog level that consists of nodes installed in strategic locations around the city, e.g., bus stops, these nodes are responsible for pre-processing data that has been received from the sensing nodes and forwards it to the cloud; and (iii) cloud level that is considered as the final destination of collected data and consists of nodes, each node contains a set virtual machine. The performance analysis of the proposed prototype proves the enhancing of memory and CPU utilization. However, SensingBus is limited because the buses cover only some areas in the city, and can-not provide wide coverage.

In [94], authors studied connected vehicle-based congestion identification techniques. Computing strategies for vehicular networks are studied to develop feasible solutions to detect congestion that performs efficiently for multiple scenarios with large coverage. The authors expected that the designed system can detect congestion to achieve traffic management objectives that are important in transportation systems. However, the proof of work is missing in this study.

Ning *et al.* [95] proposed a three-layer traffic management model in a vehicular fog environment to minimize the response time of the events that are reported and collected

in the city by vehicles. Moreover, an offloading scheme using optimization techniques is formulated by leveraging parked and moving vehicles as fog nodes. The obtained results are evaluated based on real-world taxi trajectory data. However, the vehicle's mobility impact is not studied in this work.

Moubayed *et al.* [96] proposed a V2X service placement in a mobile edge computing environment. An optimal model is formulated for service placement. Hence, to deal with dense networks they proposed a heuristic algorithm that offers efficient service placement with less complexity compared with the optimal model. The numerical results validate the efficiency of the proposed approach in terms of delay/latency and resource utilization. However, the impact of vehicle's mobility is not studied in this work.

Similarly, Shaer *et al.* [97] targeted V2X service placement in an edge computing environment. An optimal model is formulated to minimize the delay and satisfying the requirements of V2X services. The obtained results prove the soundness of the proposed approach in terms of delay. However, important metrics are missing in this work such as resource utilization.

In [98], authors proposed a real-time autonomous driving system in the edge computing environment. an offloading strategy is proposed to decide where and when to offload the tasks of autonomous driving. The authors considered only the edge power consumption minimization. However, more important metrics are not studied in this work such as resource utilization and QoS.

Hameed *et al.* [99] proposed a dynamic clustering approach in vehicular fog environment by creating clusters based on the vehicles' network. Further, a load balancing technique is used for inter and intra-cluster for IoT jobs. The division of vehicle networks into clusters increases the efficiency of the resource management, as well as provides an efficient distribution of job load among vehicles'. The performance evaluation proves the efficiency of the proposed approach in terms of QoS and energy. However, the authors used only one mobility model in this work.

Table 2.3 shows the summary of the existing works for V2X-assisted EC.

Paper	*Main Idea*	*Problem Formulation*	*Edge Position*	*Advantages*	*Limitations*
Kim *et al.* [89]	Parking management in fog-cloud environment.	Many-to-one matching game model.	Fixed	Reliability to find available parking slot.	The scalability of the fog system is not studied.
Zhang *et al.* [90]	Task offloading in mobile environment.	Stackelberg game.	Fixed	Optimal delay for tasks computation.	The risk of link failure is not studied.
Zhu *et al.* [91]	Task allocation mechanism in mobile environment.	An exact model for task allocation problem.	Mixed	Enhance the end-to-end latency.	Requires high computation resource.
Gillam *et al.* [92]	Distributed computing architecture for vehicles	N/A	Fixed	Reduce latency.	No evaluation is presented.
Caminha *et al.* [93]	SensingBus, a data collecting system from sensors on urban buses.	N/A	Mobile	Enhance CPU and Memory utilization.	The buses can not provide wide coverage.

Thakur *et al.* [94]	Connected vehicle-based congestion identification techniques	N/A	Fixed	System can detect congestion to achieve traffic management objectives	The proof of work is missing
Ning *et al.* [95]	Traffic management model in a vehicular fog environment	Exact offloading model	Fixed	Reduce the response time	The vehicle's mobility impact is not studied
Moubayed *et al.* [96]	V2X service placement in a mobile edge computing environment	Optimal V2X service placement model	Fixed	Enhance delay/latency, and resource utilization	The impact of vehicle's mobility is not studied
Shaer *et al.* [97]	V2X service placement in an edge computing environment	Optimal model for service placement	Fixed	Minimize delay	Resource utilization metric is not studied
Tang *et al.* [98]	Real-time autonomous driving system in the edge computing environment	Task offloading model	Fixed	Efficient resource utilization and power consumption	Resource utilization and QoS metrics are not studied

Hameed *et al.* [99]	Dynamic clustering approach in vehicular fog environment	Optimal model for network's utilization	Mobile	Enhance QoS and energy consumption	One mobility model is used

Table 2.3 Summary of the existing works for V2X-assisted EC.

Current convergences between V2X and EC technology should take into consideration safety metrics and real-time reactions in critical situations. Therefore, despite the existing works in EC-assisted vehicular communications, next-generation applications such as road safety and autonomous driving require efficient deployment to achieve ultra-low latency in dense vehicular networks.

UAV and EC Convergence

Unmanned Aerial vehicles (UAVs) are introduced in almost wireless communication fields to aid the EC technology due to their deployment, mobility, and cost-effective. The UAV meets EC to increase the efficiency of the offered services to end-users to achieve ultra-low latency especially for new-generation of applications. Next, we overview the existing works for UAV-aided EC technology.

Zhou *et al.* [100] tackled the UAV-enabled mobile edge computing system where the UAVs are used as mobile edge servers to offers both offloading services and energy to mobile end-users. The results prove that the proposed solution achieves efficient energy consumption. However, the mobility impact for both UAVs and mobile end-user devices is not studied in this work.

Hu *et al.* [101] proposed a UAV-enabled mobile edge computing where mobile end-users are served by UAVs that have more computing resources. The user tasks can be offloaded to the UAV or served locally depending on the task type and its requirements. The results prove the efficiency of the proposed approach. However, the mobility of both UAVs and end-users is not studied in this work.

Similarly, Xiong *el al.* [102] proposed a UAV-assisted edge computing solution where UAV

equipped with edge server and employed as a mobile base station to serve mobile end-users by offering tasks processing service. The performance evaluation demonstrates the efficiency of the proposed approach. However, the mobility is not studied in this work.

Zhan *el al.* [103] proposed a UAV-assisted edge computing solution where UAVs are considered as edge servers to serve IoT by offering computing resources to process tasks. The obtained results show that the proposed solution is efficient by ensuring the trade-off between completion time and energy consumption. However, the impact of mobility is not studied in this work.

Wu *el al.* [104] proposed a UAV-assisted edge computing architecture where UAVs are used as edge servers to accomplish IoT tasks. The obtained results prove the feasibility and the efficiency of the proposed solution in terms of energy consumption. However, more evaluation metrics need to be considered in this work.

Yang *el al.* [105] proposed a UAV-aided mobile edge computing system where a group of UAVs cooperates to provide service as edge server to IoT by considering both tasks processing and load balancing. Simulation results prove the efficiency of the proposed solution. However, both UAVs' energy consumption and computing metrics are not studied in this work.

Deng *el al.* [106] proposed a tracking system based on edge computing where UAV track target vehicle inside the city and send the captured video to the edge server. Moreover, when the target vehicle is out of the UAV coverage, it stays tracked by the camera deployed inside the city. The results show that this approach is efficient in terms of tracking timeliness and reliability. The UAV energy consumption is not studied in this work.

In [107], authors studied the integration of UAVs with the edge computing environment. The video captured by UAVs is offloaded to the edge server for processing. The proposed framework is efficient since the UAV intensive computing is offloaded to the edge which decreases resource utilization in the UAV, consequently maintain energy for a long time. However, UAVs require real-time processing with ultra-low latency which is an open challenge.

Table 2.4 shows the summary of the existing works for UAV-assisted EC.

Paper	*Main Idea*	*Problem Formulation*	*Edge Position*	*Advantages*	*Limitations*
Zhou *et al.* [100]	End-user devices computation offloading in UAV-enabled MEC system.	Optimal model for energy, resource, and trajectory.	Mobile	Efficient energy consumption.	The mobility of both UAV and users is not studied.
Hu *et al.* [101]	UAV-enabled MEC system to serve mobile end-users.	Optimal model for tasks offloading.	Mobile	Minimize latency.	The mobility of both UAVs and users is not studied.
Xiong *el al.* [102]	UAV-aided edge computing to serve IoT devices.	Optimal model for energy and resources allocation.	Mobile	Efficient energy consumption.	The mobility impact is not studied.
Zhan *el al.* [103]	UAV-assisted edge computing to serve IoT devices.	Optimal model to minimize resource utilization.	Mobile	Efficient energy consumption and resource utilization.	The impact of mobility is not studied.
Wu *el al.* [104]	UAV-assisted edge computing to accomplish IoT tasks.	Optimal algorithm for UAV position.	Mobile	Efficient energy consumption.	The resource utilization is not studied.

Yang *el al.* [105]	UAV-aided mobile edge computing system to provide service to IoT.	Optimal and deep reinforcement learning algorithms.	Mobile	Efficient tasks offloading.	UAV energy consumption is not studied.
Deng *el al.* [106]	Tracking system based on edge computing and UAV.	Set coverage and shortest path models.	Fixed	Efficient tracking timeliness and reliability.	UAV energy consumption is not studied.
Chen *el al.* [107]	UAV-edge integration for video offloading.	Framework for autonomous UAV and video offloading.	Fixed	Decreases resource utilization.	Ultra-low latency is not studied.

Table 2.4 Summary of the existing works for UAV-assisted EC.

Future network generation architectures based on UAV-EC require intelligent optimization algorithms that decide the optimal point of operations where UAV should be deployed in dense networks without neglect the energy factor that represents the important issue in UAV networks.

Conclusion

This chapter presents an overview of IoT, Cloud Computing, Mobile Cloud Computing, Edge Computing, and Mobile Edge Computing with the need of these technologies for next generation applications to satisfy the users requirements and enhance the Quality of Service. Moreover, in the second part, I presented the state of the art for both studied research

directions, *service offloading* and *vehicular edge computing*. In the following chapter, we presents our optimal proposed models for service placement in edge servers, maximize the vehicles coverage, and service offering to resolve different issues at the network edge including latency, computing resource utilization, energy consumption, etc.

Chapter 3

Optimal Solutions For Edge Computing Networks

The network optimization procedure is still an important task that needs to be carefully studied due to rapid networks evolutions and user demands. In this chapter, we present our exact models for different use cases in service offloading and vehicular edge computing research directions. For each use case, we present a detailed optimal model with the used constraints, objective function, and variables.

3.1 Service Offloading In EC

With the rapid evolution of network generations and applications requirements, service offloading is considered as one of the most important network tasks that allow to offload resource-intensive services to optimal computation hardware in both edge servers and the central cloud for processing. This complex task requires efficient communication, optimization strategies, and virtualization technologies to deal with different services and provide efficient QoS for users. In the following, we present our optimal algorithms for service offloading in EC networks.

3.1.1 UseCase 1: 5G VNF Slices Placement in EC

ETSI recently expected the benefits of MEC architecture to run all of its applications and entities as VNFs in an NFV environment [108].

The cloud VNF placement issue has gained more attention. It is similar to the placement of virtual machines (VMs), where VNFs are made up of VMs or containers that can perform network functions. Indeed, the placement of VNF at the edge of the network offers many advantages, in particular the proximity of the end-users, which decreases the latency and improves the quality of the services.

5G network slicing allows multiplexing of virtualized and independent networks, where each slice corresponds to an isolated network to meet application requirements.

The Proposed 5G-edge Computing Architecture For VNF Slices Placement

The proposed architecture allows the placement of VNF slices at the edge of the network in the context of 5G network where the physical network is divided into isolated logical networks called slices, each slice corresponds to a type of application including augmented reality (AR), IoT, unmanned aerial vehicles (UAVs), and so on.

Figure 3.1 shows the proposed 5G-edge computing architecture for VNF slices placement.

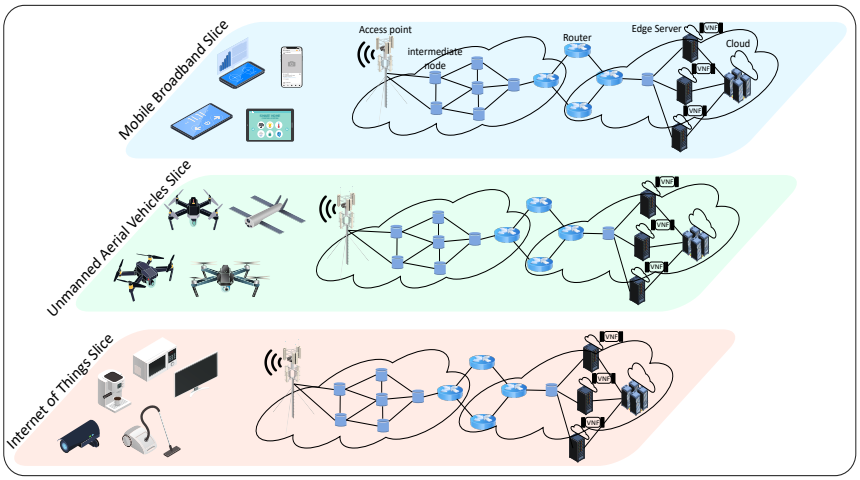


Fig. 3.1 VNF Slices Placement in Edge Computing Over 5G Network [7].

VNF Slices Placement Model Over 5G

In this part, we present the formulation of the exact model for VNF slices placement in a 5G-edge computing environment. Table 3.1 display the used notations in the proposed model. The model formulation is as follows:

Table 3.1 The notations used in the proposed VNF slices placement model.

Notations	**Definition**
N	The set of servers in the edge.
M	The set of vnf for the placement.
w_j	The number of slices in the vnf $j \in M$.
c_i	The number of cpu slots in the server $i \in N$.
α_i	Server characteristic fixed by network operators.
Decision variables	**Definition**
y_i	A binary variable indicate if the server $i \in N$ is used.
x_{ij}	A binary variable that assigns the vnf $j \in M$ to the server $i \in N$.

We present in the following the decision variables used to solve the proposed model:

- The binary variable y_i indicates if the server i is used or not. It is defined as follows:

$$y_i = \begin{cases} 1 & \textit{if the server i is used;} \\ 0 & \textit{Otherwise.} \end{cases} \tag{3.1}$$

- The binary variable x_{ij} indicates the assignment of the vnf j to the optimal server i. It is defined as follows:

$$x_{ij} = \begin{cases} 1 & \textit{if vnf j is assigned to the server i;} \\ 0 & \textit{Otherwise.} \end{cases} \tag{3.2}$$

$$\min \sum_{i=1}^{n} \alpha_i y_i \tag{3.3}$$

Subject to

$$\sum_{j=1}^{m} w_j x_{ij} \leq c_i y_i, \quad i \in N = \{0, ..., n\} \tag{3.4}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in M = \{0, ..., m\} \tag{3.5}$$

$$y_i \in \{0, 1\}, \quad i \in N \tag{3.6}$$

$$x_{ij} \in \{0, 1\}, \quad i \in N, \quad j \in M \tag{3.7}$$

The equation 3.3 formulates the proposed objective function. It represents the number of used servers. Note that each server is characterized by α_i which also depends on the server i and the operator's policy. The equation 3.4 guarantees that each server cannot exceed its computing capacity represented in the form of CPU slots. The equation 3.5 ensures that each VNF can be associated with at most one server.

The formulated problem is NP-hard because of our complex system, thus, the proposed exact algorithm is difficult to provide a decision for VNF slices placement in dense networks. Accordingly, efficient heuristic algorithms will be proposed in the next chapter.

3.1.2 Use Case 2: Task Offloading in Virtual Mobile Edge Computing (VME)

Several edge computing architectures have been proposed by using edge servers in fixed, mobile, or mixed positions each one offers benefits to the next network generations. We proposed to use the connected IoT devices in smart cities as a virtual edge server to process data at the network edge. Each selected device can execute sub-task(s) depending on its available resources for computing and the sub-tasks requirements.

VME Architecture

Figure 3.2 outlines the proposed VME architecture where IoT devices including drones, laptops, smartphones, etc. are connected in a smart city. All the devices are connected to an access point that is connected to a central cloud. When a request comes to the access point for task execution, it selects a set of IoT devices to accomplish the submitted task according to the task processing requirements and the resource availability in the connected devices that

form a virtual edge server. If the task processing requirements are higher than the available resources in the IoT devices, the task is forwarded directly to the cloud.

The aim of the network architecture design is to clarify the network communication scenarios, including the different entities that participate in the communication such as IoT devices. Thus, the proposed architecture allows to provide data processing in virtual edge servers made up of connected IoT devices, consequently, improves the efficiency of the offered services in the network by reducing the computation cost.

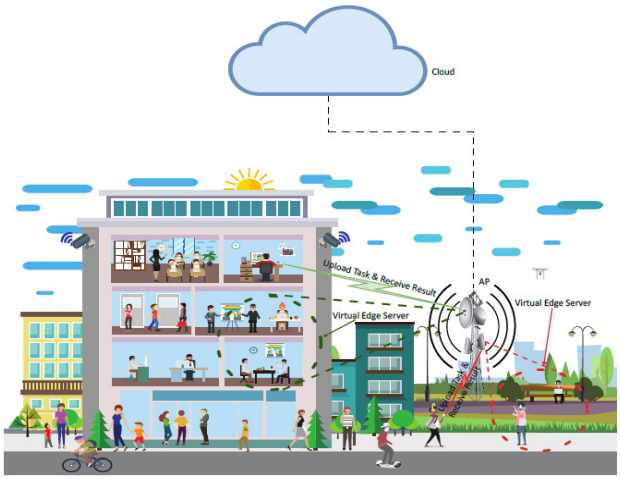


Fig. 3.2 Virtual Mobile Edge Computing Architecture [6].

Virtual Edge nodes Placement Algorithm (VEnPA)

We designed an optimal algorithm for task allocation, where the main objective is to maximize tasks allocation in virtual edge servers.

The proposed algorithm Optimal Virtual Edge nodes Placement Algorithm (VEnPA) give the optimal solution for tasks placement by allocate IoT devices to execute sub-tasks. The model input takes the network topology that consists of: IoT devices, network elements, and the cloud.

1. Problem Definition

   First, we overview the system hypotheses, after that, we present the tasks offloading model based on the exact optimization technique.

The submitted task is fragmented into sub-tasks dispatched to the IoT devices (virtual edge server) to be executed. After that, the received results of sub-tasks are combined together as the main result of the task.

Moreover, each selected IoT device has its available resources for processing (CPU, GPU, RAM, and Storage).

2. Mathematical formulation

   (a) VEnPA: System parameters

   In this part, we present the model parameters and constraints that are proposed to formulate the optimization model. This formulation defines the optimal location for task offloading and partitioning.

   We present in Table 3.2 the main VEnPA parameters and decision variables.

   (b) VEnPA: Decision variables

      i. The binary variable x represents the sub-tasks placement, and its offloading from the client device *ud* to the virtual edge device *ve* It is formulated as:

$$x_{ud,k}^{ve} = \begin{cases} 1, & \text{if service subtask } (ud,k) \text{ is placed on the virtual edge } ve \\ 0, & \text{Otherwise} \end{cases} \tag{3.8}$$

      ii. The binary variable y represents the virtual edge usage. It is defined as follows:

$$y^{ve} = \begin{cases} 1, & \text{if the virtual edge } ve \in V \text{ is used} \\ 0, & \text{Otherwise} \end{cases} \tag{3.9}$$

   (c) VEnPA: Algorithm constraints

   The formulation of the exact partitioned and offloading algorithm, **Optimal Partitioned and Offloading (OPO)** is as follows:

$$\max \sum_{ud \in U} \sum_{ve \in V} x_{ud,k}^{ve} \tag{3.10}$$

Table 3.2 The notations used in the proposed VEnPA model.

Notations	Definition
U	The set of User Applications.
V	The set of virtual edge servers.
K	The set of user application sub-tasks.
R_{ve}	Maximum RAM available in the edge device ve.
C_{ve}	Maximum CPU available in the edge device ve.
S_{ve}	Maximum Storage available in the edge device ve.
G_{ve}	Maximum GPU available in the edge device ve.
$r_{ud,k}$	Required RAM for the application sub-task (ud,k).
$c_{ud,k}$	Required CPU for the application sub-task (ud,k).
$s_{ud,k}$	Required Storage for the application sub-task (ud,k).
$g_{ud,k}$	Required GPU for the application sub-task (ud,k).
Decision variables	**Definition**
$x^{ve}_{ud,k}$	A binary variable that assigns the sub-task $k \in K$ of user application $ud \in N$ to the virtual edge $ve \in V$.

Subject to

$$\sum_{ve \in V} x^{ve}_{ud,k} \leq 1, \quad \forall ud \in U, k \in K \tag{3.11}$$

$$\sum_{ud,k} c_{ud,k} x^{ve}_{ud,k} \leq C_{ve}, \quad \forall ve \in V \tag{3.12}$$

$$\sum_{ud,k} g_{ud,k} x^{ve}_{ud,k} \leq G_{ve} \quad \forall ve \in V \tag{3.13}$$

$$\sum_{ud,k} r_{ud,k} x^{ve}_{ud,k} \leq R_{ve}, \quad \forall ve \in V \tag{3.14}$$

$$\sum_{ud,k} s_{ud,k} x^{ve}_{ud,k} \leq S_{ve}, \quad \forall ve \in V \tag{3.15}$$

$$x^{ve}_{ud,k} \in \{0,1\} \tag{3.16}$$

Algorithm constraints represent conditions that need to be verified. Equations (3.12), (3.13), (3.14), and (3.15) guarantee that the selected virtual edge device has enough amount of CPU, GPU, RAM, and storage respectively to execute the submitted user application sub-tasks. Besides, the constraint (3.11) ensure the offloading of each sub-task to only one virtual edge device.

(d) VEnPA: Algorithm complexity

The formulated OPO model is NP-hard because of the system constraints, the number of IoT devices, and the available resources that change over time in each IoT device. Accordingly, an efficient algorithm will be proposed in the next chapter.

VEnPA: evaluation of the OPO model

1. Simulation Setup

The performance of the proposed VME architecture is evaluated using three different tasks (applications) in a simulated environment. We used the simulator EdgeCloudSim [109] to simulate the VME communication and the offered services. For each application we define parameters including *CPU, GPU, RAM, storage*, and the number of *connected devices*. Table 3.3 outlines the VEnPA parameters.

2. Metrics

The performance of the VME system is based on resources utilization. Therefore, we focused on the measurement of *Processing Time, Service Time, Average Edge Devices Utilization, Failed Tasks*, and *Failed Tasks due to Edge Devices Capacity*.

Table 3.3 VEnPA parameters.

Parameter	Value
CPU slots (Min/Max)	(1/15)
GPU slots (Min/Max)	(1/10)
RAM (Min/Max) (Megabytes)	(20/2000)
Storage (Min/Max) (Megabytes)	(50/20000)
Number of connected devices (Min/Max)	(100/1000)

3. Simulation Results

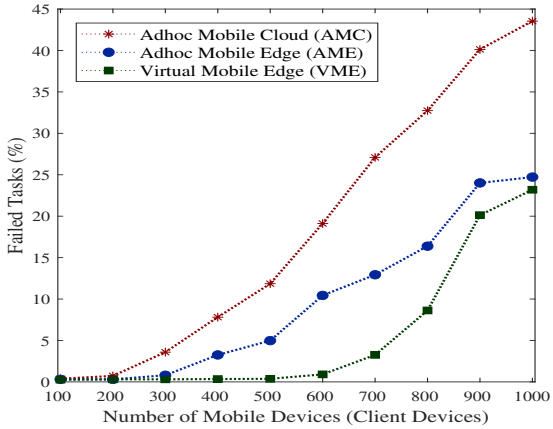


Fig. 3.3 Failed Tasks [6].

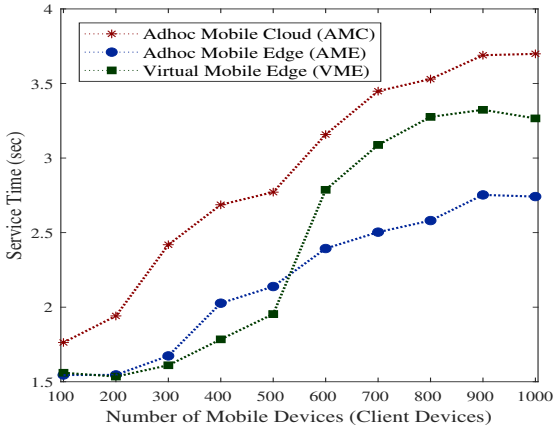


Fig. 3.4 Service Time [6].

Figures 3.3, 3.4, 3.5, 3.6, and 3.7 shows a comparison among the proposed architecture *VME*, with both *Adhoc Mobile Cloud (AMC)* that represent the architecture proposed

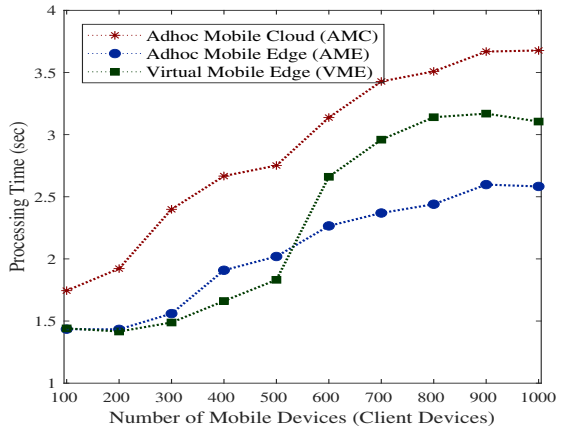


Fig. 3.5 Processing Time [6].

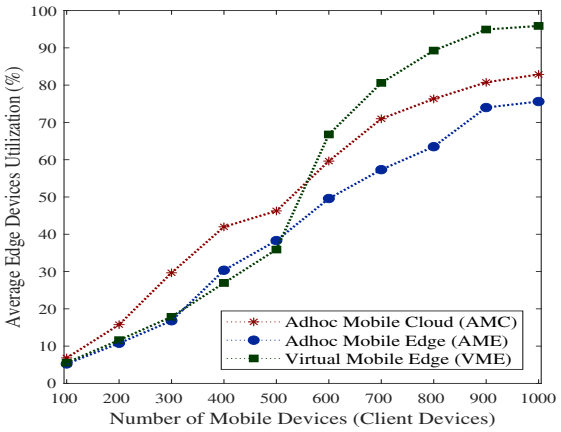


Fig. 3.6 Average Edge Devices Utilization [6].

in [77, 78, 80, 81]. And *Adhoc Mobile Edge (AME)* architecture that allow to create edge servers based on end-user devices in Adhoc mode.

Figure 3.3, show the percent of failed tasks. From the figure we show that the failed tasks increase at the same time when the number of mobile devices increase for the three architectures. For the *VME*, the increase level is low compared to the *AMC* and *AME* because of the proposed offloading algorithm can select virtual edge server for each task. Wherein the offloading take a long time in *AMC* and *AME* wich leads to task failure.

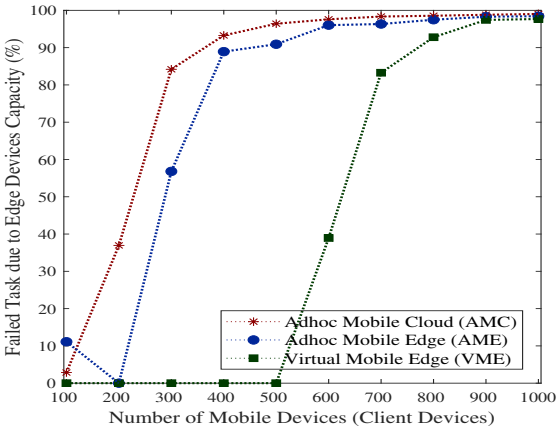


Fig. 3.7 Failed Tasks due to Edge Device Capacity [6].

Figures 3.4 and 3.5, display the service/processing time for task execution. From the figure we show that the *VME* service/processing time is low compared to the *AMC* and the *AME* when the number of mobile devices is less than 500. However, when the number of mobile devices exceed 500, the *VME* is greater than the *AME* because the *AME* failed tasks is high compared the *VME* (as shown in figure 3.3) which signify that the *VME* execute tasks more than *AME* this is because the *VME* take more time.

Beside, in the *AMC*, the end-user devices are used as cloud which leads to high level of tasks failure (as shown in figure 3.3) due to insufficient mobile cloud devices or high tasks requirements.

Figure 3.6 display the *Average Edge Devices Utilization*, we show that the edge devices utilization increase when the mobile device increase, because of the submitted tasks, in addition, the *VME* has the highest value of edge utilization because it has lowest failed tasks (as shown in figure 3.3).

Figure 3.7, show the *Failed Tasks due to Edge Devices Capacity*, we show that the *VME* has the lowest failed tasks due to edge capacity because of the proposed VEnPA model that ensure efficient offloading and placement.

3.1.3 Use Case 3: Service Offloading in Virtual Mobile Edge Computing (SO-VMEC)

As mentioned before, edge computing allows creating an intermediate layer between end-user devices and the cloud. This layer consists of access points, servers, routers, switches, etc. These devices can be replaced by IoT devices that can be used as virtual mobile edge servers (VMES).

The VMES is an edge server that offers different services such as caching and processing. It consists of connected IoT devices such as connected vehicles, UAVs, tablets, and smartphones, etc. When client devices require computation service, it offloads the VNF to the access point, after that, the VNF is split into slices. Then, the access point selects a set of IoT devices that satisfy the slices computing requirements including GPU, CPU, RAM, Energy, and Storage. These selected devices form a VMES. If the slice computing requirements are high compared with available resources in IoT devices, the slice is offloaded to the cloud. After computation, the results are sent back to the access point that combines the results to form the main result for the VNF. And finally, the result is forwarded to the source device. The VME can reduce latency, network delay, and the total cost. Further, it guarantees services with a short response time which is required in the new generation of applications.

SO-VMEC Architecture

The proposed SO-VMEC architecture consists of three layers as follows:

1. *Cloud Layer*: It is the top layer allows to process VNFs that require high processing resources.

2. *Edge Layer*: It is the medium layer composed of connected IoT devices. It provide processing services at the network edge according to the available resources in IoT devices.

3. *End-users Layer*: It is the bottom layer, represents the clients devices that require processing due to insufficient computing resources.

Figure 3.8 show the Virtual Mobile Edge Computing Architecture.

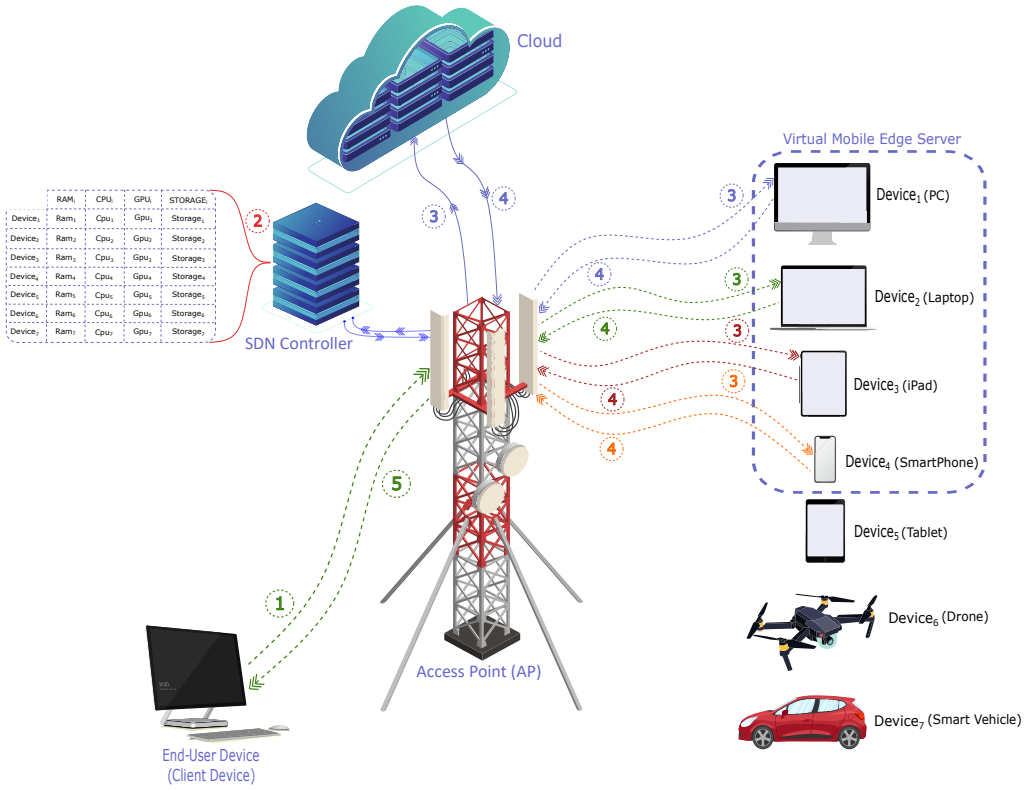


Fig. 3.8 Virtual Mobile Edge Computing Architecture [1].

SO-VMEC Communication Model

Figure 3.9 display the main steps of the SO-VMEC communication model that are detailed as follows:

1. VNF Submission and Slicing: client devices send their VNF for execution to the access point that allows resources discovery and selection of a VMES for slice allocation.

2. Resources Discovery in IoT Devices: when the access point receives the VNF from the client device, it divides the VNF into slices. After that, it starts the selection of IoT devices to form the VMES. The selection is based on the slices' computing requirements in terms of GPU, CPU, Storage, Energy, and RAM. This step taking into account the mobility of IoT devices by using the prediction technique for both mobility and energy consumption as follow:

   (a) *IoT devices mobility prediction:* The IoT device mobility prediction is an essential step to prevent the network from crashes during the processing such as

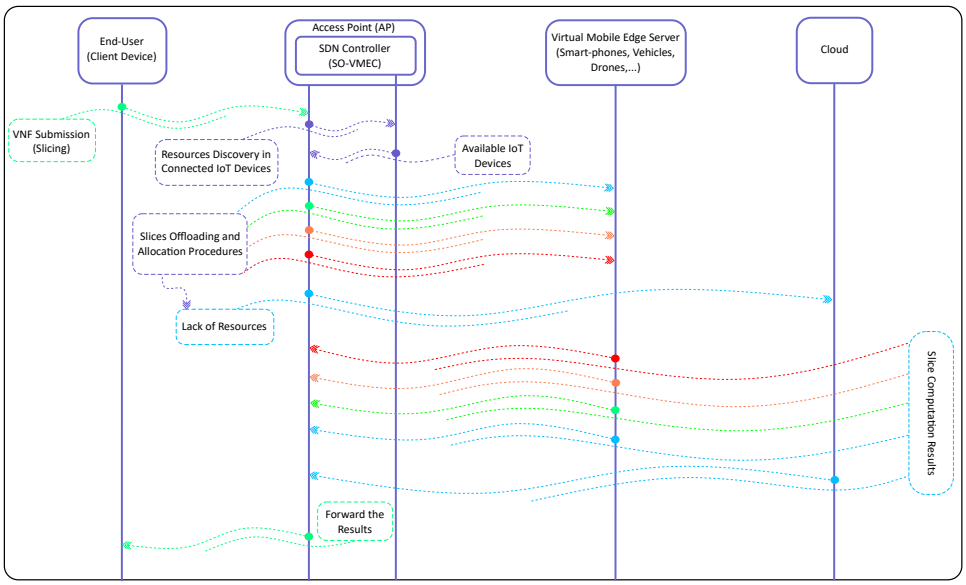


Fig. 3.9 SO-VMEC Communication Model [1].

disconnection, which decreases the efficiency of the proposed system, especially where the IoT devices are used as edge servers for processing. For this we used three different mobility models depend on the IoT device type as follow: *i*) *Low Mobility Model*: includes smart devices such as tablets, smartphones, laptops, etc. *ii*) *Medium Mobility Model*: This type of device includes drones, city buses, etc. *iii*) *High Mobility Model*: includes UAVs, vehicles, and all transportation systems in general, etc.

(b) *IoT devices energy prediction:* Most IoT devices are based on batteries as an energy source, for this, energy management is an important feature in the SO-VMEC when using the IoT devices as an edge server to process VNFs that require high processing resources include energy. The prediction technique is used to predict the IoT device energy consumption which can prevent the edge device from disconnection because of discharge during VNF slices processing.

3. *Slices Offloading and Resources Allocation*: When the VMES is selected and is ready for processing, the access point starts the slice offloading process by allocating devices from the selected VMES that satisfy the slices processing requirements. An optimization algorithm is used for devices allocation according to the network requirements.

Moreover, the cloud computing is used when no device can process slices due to resource miss.

4. *Slice Computation Results*: when the slices processing is finished in the *VMES* or/and the *cloud*, the results are sent to the access point that merges all results to form the main result of the submitted VNF.

5. *Results Forward*: in the last step, the access point sends the VNF processing result to the source device.

IoT Devices Mobility & Energy Prediction

1. Algorithms Overview:

   (a) *Long Short-Term Memory (LSTM)*: The LSTM proposed for language modeling [110], that came to resolve the vanishing gradient problem in Recurrent Neural Networks. The LSTM is composed of memory blocks called gates that contain multiplicative units. Each memory block is composed of three gates: the input gate, the output gate and the forget gate. LSTM based neural network techniques have been proposed to solve difficult problems in different fields, such as acoustic modeling, speech recognition, and traffic prediction.

   The LSTM model is important to understand what to forget, what to remember, and what to output. So when a new input arrives in the neural network module, the model first forgets any long-term information it decides it no longer needs. Then it learns which parts of the new input are worth using, and saves them into its long-term memory.

   (b) *Gated Recurrent Units (GRU)*: The GRU model is an improved version of the vanilla recurrent neural network, proposed to solve the vanishing gradient problem. Basically, it uses update and reset gates to store and filter the incoming information. It has gates responsible for modulating the information flow inside the central blocks. A GRU block consists of a couple of gates as follows: an update gate that is responsible for specifying the block updates content or activation, and a reset gate that makes the block read each input sequence without considering the previous sequence (i.e., forget the previous state).

We used both LSTM and GRU to predict the IoT devices mobility and energy consumption for the three mobility and energy consumption models based on real data-sets. The prediction step is required later to feed the proposed SO-VMEC optimization algorithms.

2. Data-sets Description:

   (a) *Real mobility traces*: Three datasets (Low mobility, medium mobility, and high mobility) are used for the mobility prediction.

      i. Low mobility (Smartphone): The data represents the walker mobility recorded every 0.6 seconds in the Ostermalm area inside Stockholm city, Sweden [111].

      ii. Medium mobility (City Bus): The dataset represents buses mobility in Rio de Janeiro, Brazil [112]. The data is represented as follows: the date and time, the bus id, the speed, the bus line, the longitude, and the latitude. The CSV file contains the positions of more than 12000 buses.

      iii. High mobility (Vehicle): The data represents taxis positions for one month in Rome city, Italy [113]. The data is classified in a text file represented as follows: taxi id, date and time (timestamp), and taxi position.

   (b) *Real IoT devices energy consumption traces*: We used three datasets for different battery types (low capacity (smartphone), medium-capacity (laptop), and high capacity (smart TV)).

      i. Low energy consumption (Smartphone): the dataset contains mobile phone usage of 342 persons for 3 months in Spain [114]. The dataset contains the events of 25 sensors such as location and received notifications. The sensors such as data consumption, phone calls, cell tower, airplane mode, Wifi details, SMS events, etc.

      ii. Medium energy consumption (Laptop): The dataset contains the electrical devices power consumption collected in Darmstadt city, Germany [115]. the data is grouped into directories and stored in CSV files where each directory corresponds to a device type. The data is represented as follows: date, time, and power consumption.

iii. High energy consumption (Smart TV): The dataset contains real power consumption data of electrical devices inside 22 houses in Korea [116]. The data is grouped in directories where each directory contains the data for a house. The data contains the timestamp and the power.

Optimization Model for SO-VMEC

In this section, we present the exact optimization model for the proposed SO-VMEC. The model takes as inputs the results of the mobility and the energy prediction, as well as the system capacity in terms of network, capacity, computing, and storage. It allows to optimally place the VNF slices in the selected VMES.

Table 3.4 The notations used in the proposed SO-VMEC model.

Edge notations	**Definition**
U	The set of User Devices, It represents user demands in term of VNF slices.
V	The set of virtual edge servers.
K	The set of (user) VNF slices.
E_{ve}	The maximum Energy available in the edge device ve (i.e., at a specific time slot according to the periodicity of the algorithm).
M_{ve}	The predicted (required) Mobility for the edge device ve during the VNF slice service time. It is represented as a Boolean matrix, detecting if the edge device is available or not during the service period.
R_{ve}	The maximum RAM available in the edge device ve.
C_{ve}	The maximum CPU available in the edge device ve.
S_{ve}	The maximum Storage available in the edge device ve.
G_{ve}	The maximum GPU available in the edge device ve.
VNF slices notations	**Definition**
$e_{ud,k}$	The predicted (required) Energy for the VNF slice (ud,k).
$r_{ud,k}$	The required RAM for the VNF slice (ud,k).
$c_{ud,k}$	The required CPU for the VNF slice (ud,k).
$s_{ud,k}$	The required Storage for the VNF slice (ud,k).
$g_{ud,k}$	The required GPU for the VNF slice (ud,k).
Decision variables	**Definition**
$x^{ve}_{ud,k}$	A binary variable that assigns the slice $k \in K$ of an end-user device $ud \in U$ to the virtual edge $ve \in V$.

1. Algorithm Constraints and Objective

   The studied problem considers VNF slices offloading to VMES served by the network operator. The problem objective is to determine where to offload the submitted VNF slices in an optimal way by taking into account the resource allocation and the mobility of the devices.

   We considered different constraints for the proposed SO-VMEC optimization model such as GPU, RAM, Storage, CPU, and IoT devices energy/mobility prediction. The main objective is to maximize the offload of VNF slices.

2. Model Formulation

   In this part, we present the defined parameters, constraints, and the proposed optimization model. This formulation represents the optimal offloading of VNF slices to the available devices that form the VMES.

   (a) Decision Variables: We quote in Table 3.4 the used parameters and variables. They are defined as follows:

      i. The binary variable x represents the VNF slices placement, and its offloading from the client device ud to the optimal device ve in the VMES. It is defined as:

$$x_{ud,k}^{ve} = \begin{cases} 1 & \text{if VNF slice } (ud,k) \text{ is placed} \\ & \text{on the virtual edge } ve \\ 0 & \text{Otherwise} \end{cases} \tag{3.17}$$

      ii. The binary variable y represents the VMES usage. It is formulated as follows:

$$y^{ve} = \begin{cases} 1 & \text{if the virtual edge } ve \in V \text{ is used} \\ 0 & \text{Otherwise} \end{cases} \tag{3.18}$$

   (b) Constraints: depending on the proposed communication model, the step of VNF slices placement is constrained by the computing resources (GPU, RAM, CPU, Storage, Energy), and the mobility of the devices. Hereafter, we present the constraints related to the proposed SO-VMEC.

i. We propose that only one optimal VMES $ve \in V$ should execute the offloaded VNF slice $(ud,k) \in (U,K)$. The constraint formulation is as follows:

$$\sum_{ve \in V} x_{ud,k}^{ve} \leq 1, \quad \forall ud \in U, k \in K \tag{3.19}$$

ii. According to the energy prediction, the selected device should have the necessary energy to execute the VNF slice(s), the formulation is as follow:

$$\sum_{ud,k} e_{ud,k} x_{ud,k}^{ve} \leq E_{ve}, \quad \forall ve \in V \tag{3.20}$$

iii. The available RAM in the server *ve* should be enough to execute the VNF slice(s), the constraint formulation is as follow:

$$\sum_{ud,k} r_{ud,k} x_{ud,k}^{ve} \leq R_{ve}, \quad \forall ve \in V \tag{3.21}$$

iv. The available storage in the server *ve* should be enough, the constraint formulation is as follow:

$$\sum_{ud,k} s_{ud,k} x_{ud,k}^{ve} \leq S_{ve}, \quad \forall ve \in V \tag{3.22}$$

v. The available CPU slots of the server *ve* should be enough to process the VNF slice(s), the constraint formulation is as follow:

$$\sum_{ud,k} c_{ud,k} x_{ud,k}^{ve} \leq C_{ve}, \quad \forall ve \in V \tag{3.23}$$

vi. The GPU capacity of the server *ve* should be enough to process the VNF slice(s), the constraint formulation is as follow:

$$\sum_{ud,k} g_{ud,k} x_{ud,k}^{ve} \leq G_{ve} \quad \forall ve \in V \tag{3.24}$$

vii. Guaranteeing the non-negativity of decision variables:

$$x^{ve}_{ud,k} \in \{0,1\} \tag{3.25}$$

(c) Objective Function Formulation: The general model formulation allows maximizing the VNF slices placement in the VMES. The formulation of the objective function is as follows:

$$\max \sum_{ud,k} \sum_{ve \in V} x^{ve}_{ud,k} \tag{3.26}$$

SO-VMEC Performance Evaluation

To evaluate the proposed SO-VMEC models, we used Keras [1] and TensorFlow [2] platforms for the devices mobility and energy prediction. Moreover, we used CPLEX for python tool [3] and EdgeCloudSim [109] to assess the efficiency of the SO-VMEC in both small and dense networks. Table 3.5 show the SO-VMEC parameters.

Table 3.5 SO-VMEC parameters.

Parameter	Value
CPU slots (Min/Max)	(1/30)
GPU slots (Min/Max)	(1/15)
RAM (Min/Max) (Megabytes)	(20/8000)
Storage (Min/Max) (Megabytes)	(50/50000)
Number of Devices (Min/Max)	(100/1000)

1. Key Performance Indicators (KPIs)

To evaluate the efficiency of the proposed approach, we used the following KPIs:

(a) Failed VNFs Execution: We proposed to measure the failed VNFs to evaluate the proposed model. Because the increase of failed VNFs decreases the efficiency of the offered service. We defined the successfully allocated VNF slices (NOS) as follow:

$$NOS = \sum_{ud \in U} \sum_{k \in K} x^{ve}_{ud,k} \tag{3.27}$$

[1] https://keras.io/
[2] https://www.tensorflow.org/
[3] https://pypi.org/project/cplex/

In the case of resource lack on VMES, the VNFs slices are offloaded to the cloud. The formulation of offloaded slices is as follows:

$$\sum_{ud \in U} \sum_{k \in K} \left(1 - x_{ud,k}^{ve}\right). \tag{3.28}$$

(b) Service Time: It allows measuring the total service duration that includes the VNF submission, slicing, VMES selection, slices offloading, and the results forwarding to the client device.

(c) Processing Time: represents the duration required to process the VNF slices in the selected VMES.

(d) Failed VNFs Execution Due To Edge Capacity: resource availability is important during processing to decrease the failed VNFs. When the GPU, RAM, CPU, and Storage satisfy the VNFs requirements, it increases the QoS. We defined the cost of the resources as follow:

$$CPU\ Cost = \frac{\sum_{ud,k} c_{ud,k} x_{ud,k}^{ve}}{\sum_{ve \in V} C_{ve}} \tag{3.29}$$

$$GPU\ Cost = \frac{\sum_{ud,k} g_{ud,k} x_{ud,k}^{ve}}{\sum_{ve \in V} G_{ve}} \tag{3.30}$$

$$RAM\ Cost = \frac{\sum_{ud,k} r_{ud,k} x_{ud,k}^{ve}}{\sum_{ve \in V} R_{ve}} \tag{3.31}$$

$$Storage\ Cost = \frac{\sum_{ud,k} s_{ud,k} x_{ud,k}^{ve}}{\sum_{ve \in V} S_{ve}} \tag{3.32}$$

(e) Failed VNFs Execution Due To Mobility: the IoT devices mobility impacts the SO-VMEC system which leads to decrease the QoS. For this, the prediction is used to estimate the future devices movement and prevent the system from disconnection because of devices mobility.

(f) Failed VNFs Execution Due To Energy: most IoT devices use battery as an energy source, for this we used the prediction to estimate the energy consumption to prevent the system from disconnection that leads to failed VNF slices processing.

The IoT device energy cost is formulated as follows:

$$Energy\ Cost = \frac{\sum_{ud,k} e_{ud,k} x^{ve}_{ud,k}}{\sum_{ve \in V} E_{ve}} \tag{3.33}$$

(g) Average Edge Devices Utilization: it represents the average number of used VMES for VNF slices processing. it depends on the available resource in the VMES, and the number of clients requests.

2. Comparative Analysis

To assess the behavior of the proposed SO-VMEC, we compared the efficiency of the proposed approach with other networking architectures from the state-of-the-art. We summarize these architectures as follows:

(a) Adhoc Mobile Cloud (AMC): proposed in [77, 78, 80, 81, 85, 86], this architecture allows using connected devices as adhoc cloud for processing. However, the risk of offloading failure is high in the case of unavailable devices, or when the VNF requirements are high compared with available resources.

(b) Adhoc Mobile Edge (AME): This architecture allows using connected devices as on-demand edge servers in adhoc mode, the submitted VNF is processed in the selected edge servers, wherein in the case of resource unavailability, the VNF is offloaded to the cloud. The risk of VNF failure is high in this architecture because of device disconnection.

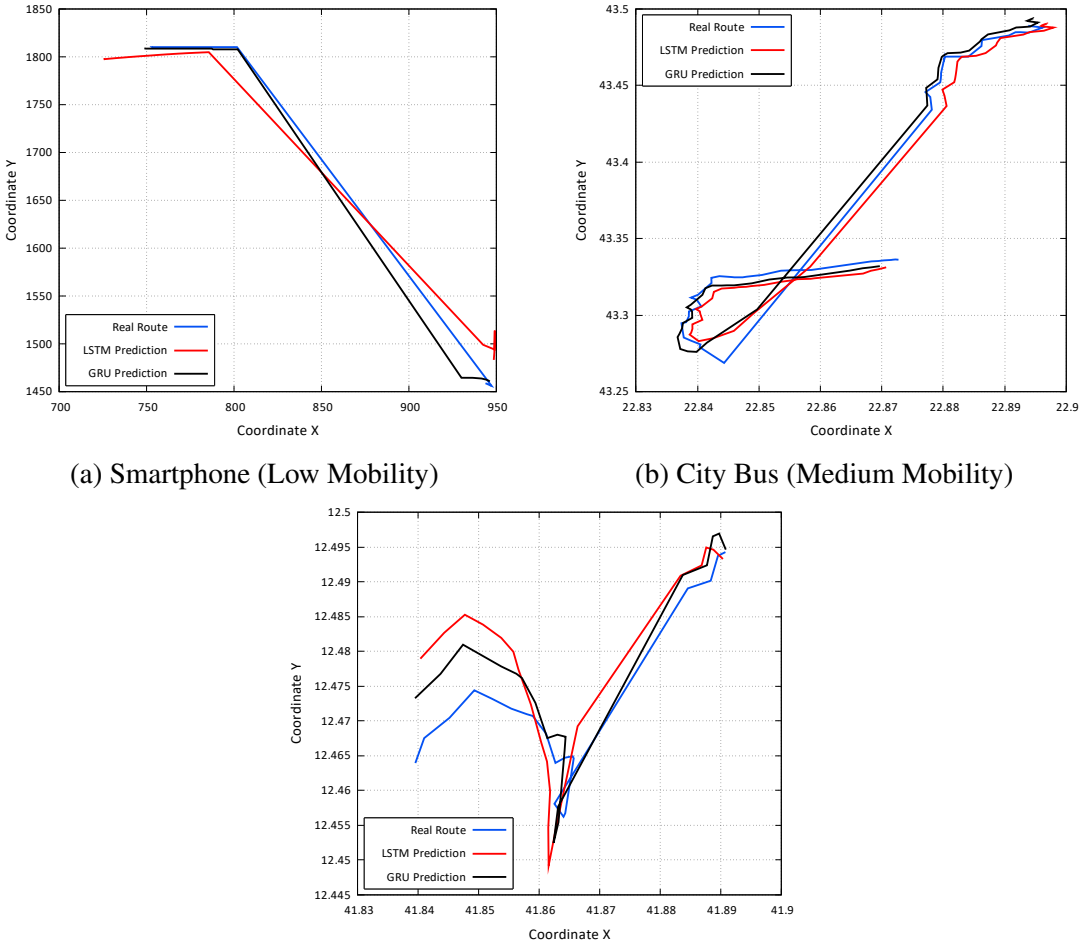
Table 3.6 show a summary of the main problems studied in this work compared with existing architectures.

Architecture	*Mobility impact*	*Energy impact*	*Failure offloading management*
AMC	✗	✗	✗
AME	✗	✗	✗
VME	✓	✓	✓

Table 3.6 The main strong points of VME against AME and AMC.

Hereafter, we evaluate the proposed algorithms under the above KPIs, and we present the comparison of the proposed architecture (VME) with both AMC and AME.

3. Prediction Results



(a) Smartphone (Low Mobility)

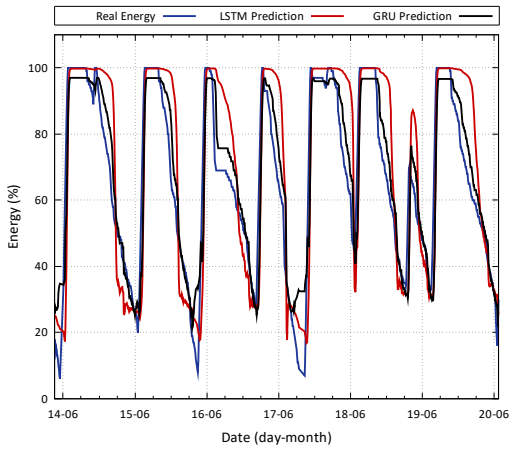
(b) City Bus (Medium Mobility)

(c) Vehicle (High Mobility)

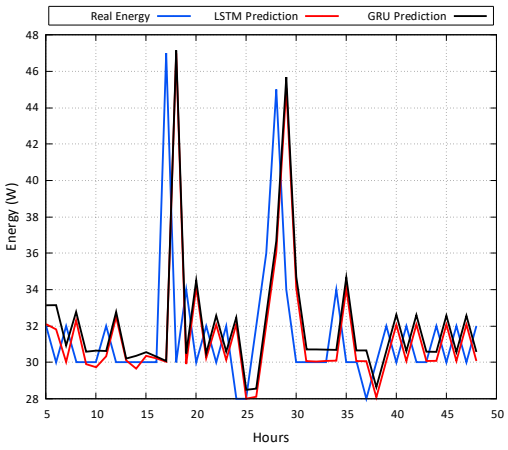
Fig. 3.10 Mobility Prediction for Three Mobility Models (Low, Medium, and High) [1].

Figure 3.10 shows the mobility prediction for three different mobility models (low, medium, and high). The obtained results show that both LSTM and GRU are efficient in terms of prediction accuracy.

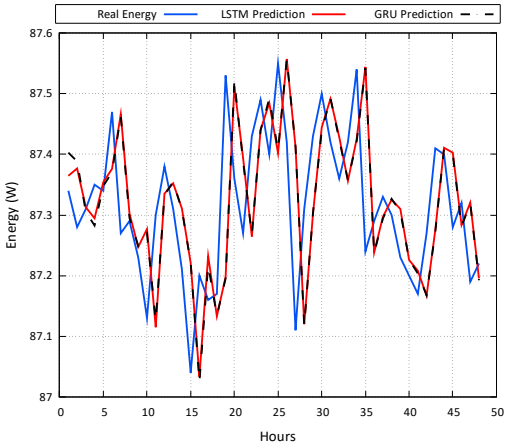
Figure 3.11 show the energy prediction results of LSTM and GRU algorithms. From the figure, we show that both algorithms converge to real energy values which help the system to prevent crashes during the processing step.



(a) Smartphone (Low)



(b) Laptop (Medium)



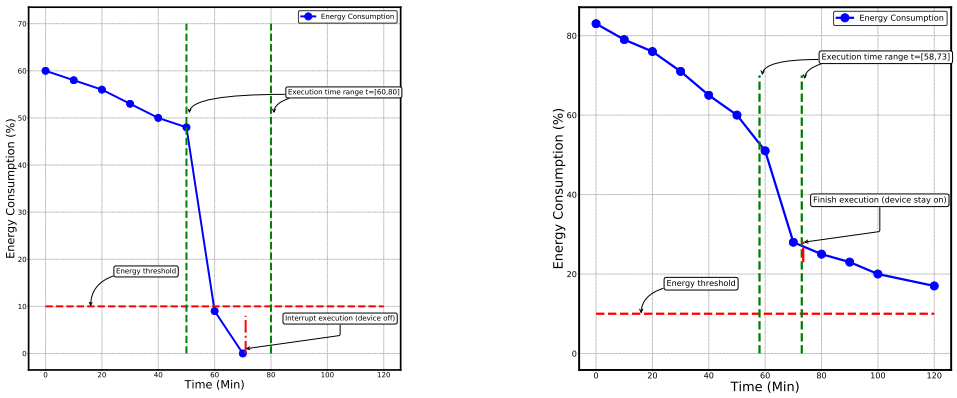
(c) Smart TV (High)

Fig. 3.11 Energy Prediction for Three Energy Models (Low, Medium, and High) [1].

Figure 3.12 shows the energy prediction impact on VNF slices execution. In figure (3.12a), the selection of VMES without prediction leads to processing interruption because the battery is running out of power during the VNF slices processing. Where in figure (3.12b), the selection of VMES using the prediction prevents the system from execution failure which improves the offloading efficiency on the VMES.

4. Optimization Results

Figure 3.13 shows the performance evaluation of the proposed VME architecure compared with both *AMC* et *AME* architectures.



(a) Selected IoT Without Energy Prediction.

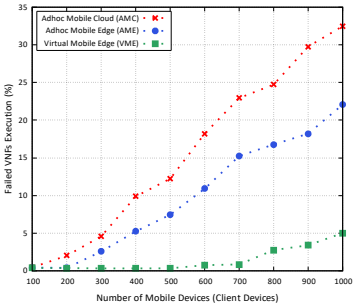
(b) Selected IoT With Energy Prediction.

Fig. 3.12 Energy Prediction Impact on VNF Slices Execution [1].

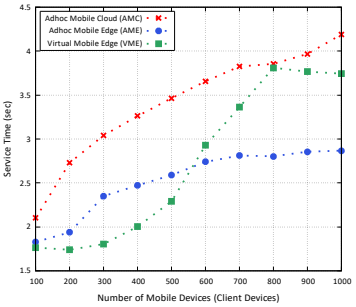
Figure (3.13a) displays the failed VNFs percentage when the number of client devices increases. We show that when the number of client devices increases the failed VNFs increase also for the three architectures. But, the percent of failed VNFs is very low in the *VME* compared with the other architectures because both *AMC* and *AME* take more time to select the edge devices which leads to a failure in VNF slices execution.

Figures 3.13b and 3.13c, shows the service/processing time for VNF slices execution. From the figure we show that the *VME* service/processing time is low compared to the *AMC* and the *AME* when the number of client devices is less than 550. However, when the number of client devices exceed 550, the *VME* is greater than the *AME* because the *AME* failed tasks is high compared the *VME* (as shown in figure 3.13a) which signify that the *VME* execute tasks more than *AME* this is because the *VME* take more time. While the service/processing time for the *AMC* is stay high because of the high failed VNF slices execution (as shown in figure 3.13a) due to high VNF slices requirements or insufficient edge devices.

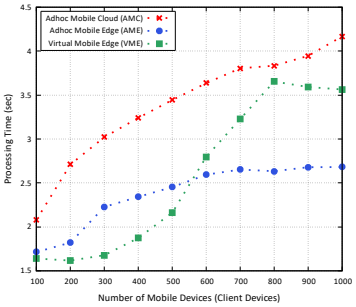
Figure (3.13d) shows the *failed VNFs processing due to edge capacity*. From the figure, we show that the *VME* outperforms the other architectures because it offloads VNFs correctly (fewer VNFs failed due to edge capacity). Moreover, the proposed model manages optimally the IoT devices resource.



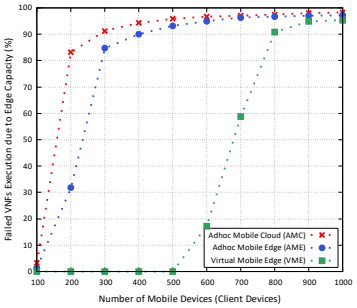
(a) Failed VNFs Execution



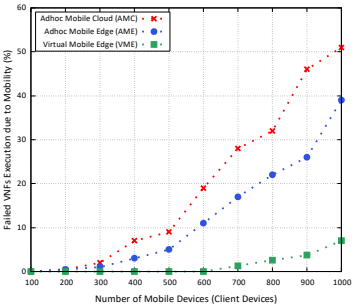
(b) Service Time



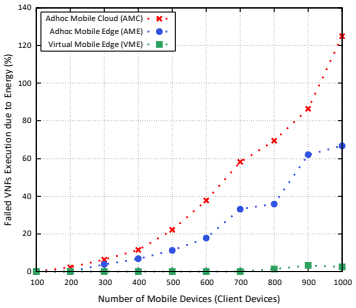
(c) Processing Time



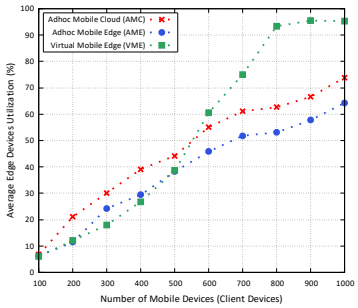
(d) Failed VNFs Execution Due To Edge Capacity



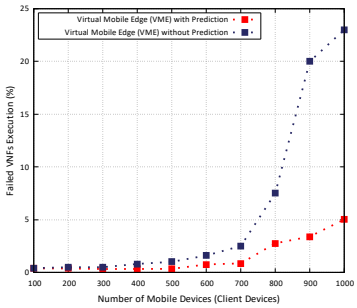
(e) Failed VNFs Execution Due To Mobility



(f) Failed VNFs Execution Due To Energy



(g) Average Edge Devices Utilization



(h) Failed VNFs Execution With(out) Prediction

Fig. 3.13 SO-VMEC Evaluation Results [1].

Figure (3.13e) displays the impact of device mobility in terms of failed VNFs. The proposed VME architecture outperforms both *AME* and *AMC* because of the used prediction technique that prevents the system from disconnection which by consequence decrease the failed VNFs.

Figure (3.13f) shows the impact of the energy consumption of IoT devices on the failed VNFs processing. The obtained results show that the failed VNFs is very low in the *VME* compared to both *AME* and *AMC* because of the used prediction technique

that helps the system to prevent crashes during the processing which by consequence decrease the number of failed VNFs.

Figure (3.13g) displays the VMES utilization. From the figure, we show that the VMES utilization increase because of the increase of incoming VNFs from the client devices. Indeed, the *VME* architecture needs many VMES since it offloads a high number of VNFs (less failed VNFs as presented in the figure 3.13a).

Figure (3.13h) shows the number of failed VNFs with and without prediction. From the figure, we show that the failed VNFs is decreased when the prediction technique is used for both energy and mobility as proved in the figures ((3.13e) and (3.13f)).

3.1.4 Use Case 4: Service Function Chains Orchestration in Virtual Mobile Edge Computing (VMEC)

IoT-VMEC Architecture

1. IoT-VMEC System

   The huge proliferation of IoT devices led to creates a complex network that generates a high amount of data that require a short time for processing. Edge computing offers a distributed architecture that provides efficient processing at the network edge. We proposed to use IoT devices as virtual mobile edge server (VMES) for service function chain (SFC) orchestration where artificial intelligence (AI) is used to help using both cloud and edge in an intelligent way. Figure 3.14 show the proposed IoT-VMEC architecture.

   The integration of AI at the network edge aims at:

   (a) **Data collection and analysis:** based on New Radio (NR) gNB network gateways.

   (b) **Mobility prediction of IoT devices:** at the gNB, we used deep learning to predict the IoT devices mobility to prevent the network from SFC VNFs execution failure due to mobility.

   (c) **Energy prediction of IoT devices:** based on deep learning, the gNB node predict the IoT devices energy consumption to prevent the network from SFC VNFs processing failure due to energy or battery crashes.

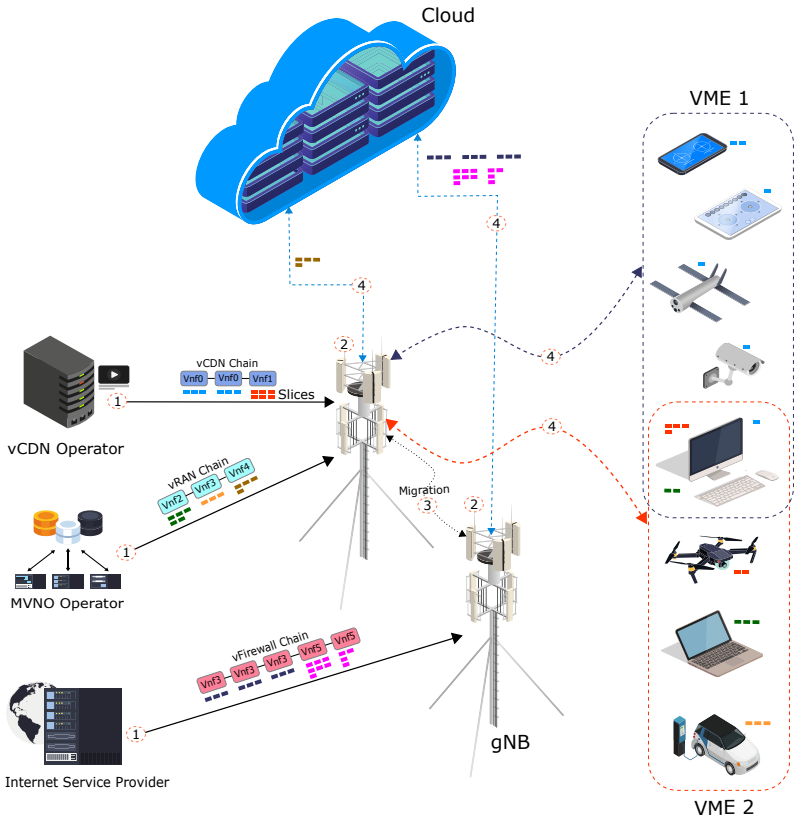


Fig. 3.14 Proposed IoT-VMEC Architecture.

(d) **Self-Organizing Network (SON) resources:** based on the AI outputs (the predicted mobility and energy), an optimization module is used to organize the SFC VNFs offloading (MVNO, vCDN,...) to available IoT devices (VMES).

For the purpose of efficient SON resources management, the different network entities cooperate together to ensure the processing of SFC parts (VNF slices) in an efficient way as follows:

(a) **Local processing of VNF slices** at the VMES where each IoT device process VNF slice(s) according to their available computing resources.

(b) **Migration of VNF slices** between VMESs in the case of lack of resources in the selected VMES.

(c) **Offloading of VNF slices** to the centralized cloud in the case of a lack of computing resources for both local processing and migration.

2. IoT-VMEC Communication Model

   Figure 3.14 shows the main system entities for SFC orchestration in VMEC. The communication steps (from 1 to 4) are presented as follows:

   (a) **Step (1)**: the distributed entities (vCDN, MVNO operators, etc) offload their SFC to the gNB for placement.

   (b) **Step (2)**: happens in the gNB allow to find a set of IoT devices that satisfy the requirements (vRAM, vGPU, vCPU, and vStorage) of the received SFC based on the energy and mobility prediction to create a VMES.

   (c) **Step (3)**: If the selected VMES does not satisfy all the SFC parts (VNF slices) requirements, the gNB tries to find IoT devices connected to the neighbor's gNB (using the migration) that can satisfy the rest of VNF slices requirements.

   (d) **Step (4)**: The last step is to send the VNF slices to the IoT devices in the selected VMES. Moreover, the rest of VNFs slices where their requirements are not satisfied in the selected VMES or the other VMES selected by migration, it offloaded to the cloud.

OSPV: Optimal SFC Placement in IoT-VMEC

We proposed an optimal SFC placement algorithm in IoT-VMEC using optimization technique. Table 3.7 outlines the main parameters and decision variables.

The Equations (3.34) and (3.35) represent the decision variables. The binary variable x indicates the placement of the VNF slices on the VMES. The binary variable y represent the VMES usage.

$$x_{sc,vs}^{gnb,ve} = \begin{cases} 1 & \text{if the VNF slices } vs \in V_s \text{ is placed} \\ & \text{on the server } ve \text{ of the } gnb \in G_{nb}. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.34}$$

Table 3.7 The parameters used in the proposed OSPV model.

Notations	***Definition***
S_c	The set of Service Chains.
V_s	The set of VNFs. Each VNF is composed of slices.
G_{nb}	The set of gNBs (gnb).
V	The set of IoT devices available at $gnb \in G_{nb}$.
K_{sc}	The number of VNFs in each service chain $sc \in S_c$.
C_{ve}	The maximum vCPU available in the server $ve \in V$.
G_{ve}	The maximum vGPU available in the server $ve \in V$.
R_{ve}	The maximum vRAM available in the server $ve \in V$.
S_{ve}	The maximum vStorage available in the server $ve \in V$.
E_{ve}	The maximum Energy available in the edge server $ve \in V$.
SFC slices notations	***Definition***
$e_{sc,vs}$	The predicted (required) Energy for the SFC slice (sc, vs).
$r_{sc,vs}$	The required RAM for the SFC slice (sc, vs).
$c_{sc,vs}$	The required CPU for the SFC slice (sc, vs).
$s_{sc,vs}$	The required Storage for the SFC slice (sc, vs).
$g_{sc,vs}$	The required GPU for the SFC slice (sc, vs).
Decision Variable	***Definition***
$x_{sc,vs}^{gnb,ve}$	A binary variable that allocates the SFC slice $vs \in V_s$ of the service chain $sc \in S_c$ to the server $ve \in V$ of the $gnb \in G_{nb}$.
$y^{gnb,ve}$	A binary variable that indicates if the server $ve \in V$ of the $gnb \in G_{nb}$ is used.

$$y^{gnb,ve} = \begin{cases} 1 & \text{if the server } ve \in V \text{ of the } gnb \in G_{nb} \\ & \text{is used.} \\ 0 & \text{Otherwise.} \end{cases} \tag{3.35}$$

The objective function **Optimal SFC Placement in VMEC (OSPV)** represented in the equation (3.36) allow to maximize the allocated SFC by using the minimum number of VMES.

$$\max \sum_{sc\in S_c, vs\in V_s} \sum_{gnb\in G_{nb}, ve\in V} x_{sc,vs}^{gnb,ve} - \sum_{gnb\in G_{nb}, ve\in V} y^{gnb,ve} \tag{3.36}$$

Subject to

$$\sum_{gnb\in G_{nb}, ve\in V} x_{sc,vs}^{gnb,ve} \leq 1, \quad \forall sc \in S_c, vs \in V_s \tag{3.37}$$

$$\sum_{sc\in S_c, vs\in V_s} e_{sc,vs} x_{sc,vs}^{gnb,ve} \leq E_{ve} y^{gnb,ve}, \forall gnb \in G_{nb}, ve \in V \tag{3.38}$$

$$\sum_{sc\in S_c, vs\in V_s} r_{sc,vs} x_{sc,vs}^{gnb,ve} \leq R_{ve} y^{gnb,ve}, \forall gnb \in G_{nb}, ve \in V \tag{3.39}$$

$$\sum_{sc\in S_c, vs\in V_s} s_{sc,vs} x_{sc,vs}^{gnb,ve} \leq S_{ve} y^{gnb,ve}, \forall gnb \in G_{nb}, ve \in V \tag{3.40}$$

$$\sum_{sc\in S_c, vs\in V_s} c_{sc,vs} x_{sc,vs}^{gnb,ve} \leq C_{ve} y^{gnb,ve}, \forall gnb \in G_{nb}, ve \in V \tag{3.41}$$

$$\sum_{sc\in S_c, vs\in V_s} g_{sc,vs} x_{sc,vs}^{gnb,ve} \leq G_{ve} y^{gnb,ve}, \forall gnb \in G_{nb}, ve \in V \tag{3.42}$$

$$\sum_{gnb\in G_{nb}, ve\in V} x_{sc,vs}^{gnb,ve} = \sum_{gnb\in G_{nb}, ve\in V} x_{sc,vs+1}^{gnb,ve}, \quad \forall sc \in S_c, vs \in V_s \setminus \{K_{sc}\} \tag{3.43}$$

$$0 \leq x_{sc,vs}^{gnb,ve} \leq 1, \forall gnb \in G_{nb}, ve \in V, sc \in S_c, vs \in V_s \tag{3.44}$$

$$0 \leq y^{gnb,ve} \leq 1, \forall gnb \in G_{nb}, ve \in V \tag{3.45}$$

The constraints of the proposed **OSPV** model are presented as follows:

Equation (3.37) ensures that the VNF slices $vs \in V_s$ of the service chain $sc \in S_c$ is placed in only one server $ve \in V$ of a $gnb \in G_{nb}$.

1. Equations (3.38), (3.39), (3.40), (3.41), and (3.42) guarantees that the selected server has enough energy, vRAM, vStorage, vCPU, and vGPU resources to execute the VNF slices $vs \in V_s$ of the service chain $sc \in S_c$.

2. Equation (3.43) guarantee that all the VNF slices $vs \in V_s$ of the service chain $sc \in S_c$ are placed.

3. Equations (3.44) and (3.45) ensures the non-negativity of the decision variables.

The proposed model is adapted to small-scale networks. Because of the high complexity in dense networks, we will propose a heuristic placement technique to solve the SFC placement in the IoT-VMEC architecture. The heuristic algorithm will be presented in the next chapter.

LSTM and GRU Models Evaluation

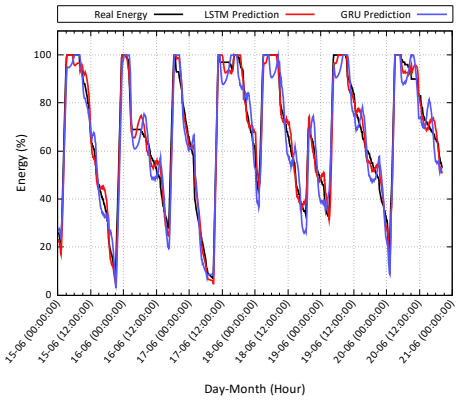
We used LSTM and GRU to predict both the energy and the mobility of IoT devices (Section 3.1.3), The obtained results are presented as follows:

1. Energy Prediction

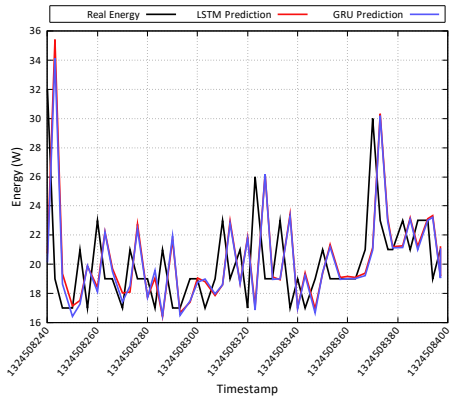
   Figure 3.15 show the energy consumption of IoT devices in three different consumption models (low, medium, and high). The obtained prediction results prove the efficiency of both LSTM and GRU models that is near to real energy data which is accurate the three consumption models 3.15a, 3.15b, and 3.15c respectively which prevent the system from problems such as crashes during the process step.

2. Mobility Prediction

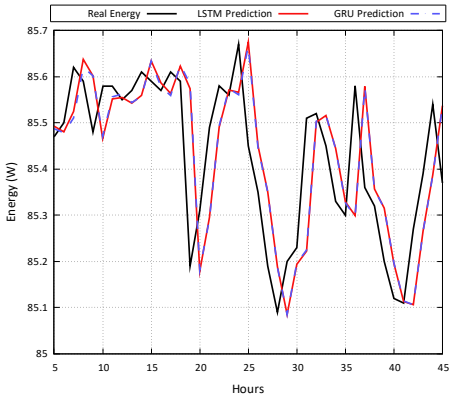
   Figure 3.16 display the IoT mobility prediction. We predict the coordinates (x, y) correspond to the positions for both pedestrian (smartphone) and city bus that represents two different mobility models. From the figures 3.16a, and 3.16b we show that the LSTM and GRU models are efficient in terms of prediction where the two are near to real positions, by consequence prevent the virtual mobile edge from crashes during processing caused by mobility.



(a) Smartphone (Low)

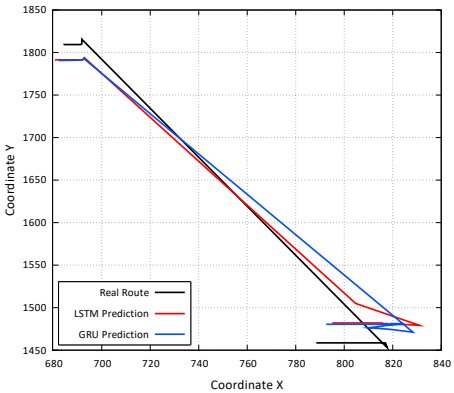


(b) Laptop (Medium)

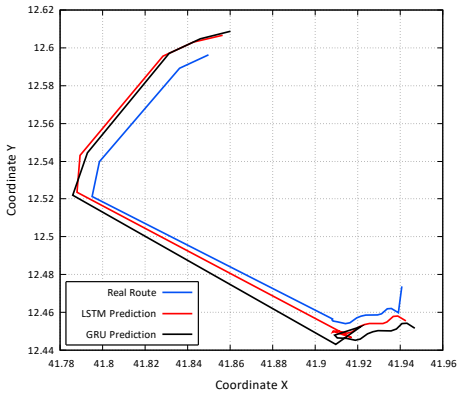


(c) Smart TV (High)

Fig. 3.15 Energy Prediction for Three Energy Models (Low, Medium, and High).



(a) Smartphone of Pedestrian



(b) Connected Vehicle

Fig. 3.16 Mobility Prediction for Two Different Mobility Models (Low and High).

3.2 Vehicular Edge Computing

With the recent development of urban networks, smart vehicles are considered as mobile devices equipped with different on-board sensors. Nowadays vehicles start to have the capability to communicate with each others (vehicle-to-vehicle) and with urban infrastructure (vehicle-to-infrastructure), collect data, and perform computation and in-vehicle processing. Today, typically, high computation applications are executed on the cloud. However, delay-sensitive applications can suffer from the long-time response and may suffer bad quality of experience.

The new vehicular networks generations allow using vehicles as an edge server in the context of vehicular edge computing systems (VEC) for data processing, video caching, etc. It allows to provide information to other connected devices included end-user devices, other vehicles, etc.

3.2.1 Use Case 1: Edge Computing Assisted Vehicular Networks for Service Offering

We proposed to use rideshare taxis as mobile edge server to offer video chunks to connected vehicles inside the city. We designed a distributed vehicular edge computing architecture that provide data caching and low latency communication.

MVEC: Mobile Vehicular Edge Computing Architecture

The main objective of designing any network architecture or protocol is to provide efficient communication with high scalability support, which is considered as a fundamental challenge. Adding the huge number of users, applications, and the dynamic mobility make the design more challenging. Thus, MVEC architecture (Figure 3.17) aims to distribute the communication and computation in a vehicular environments. This is achieved by using rideshare taxis as Mobile Edge Server, which covers a large city and provides ubiquitous content distribution and computation. That by consequence enhances the communication and data delivery in delay-sensitive applications.

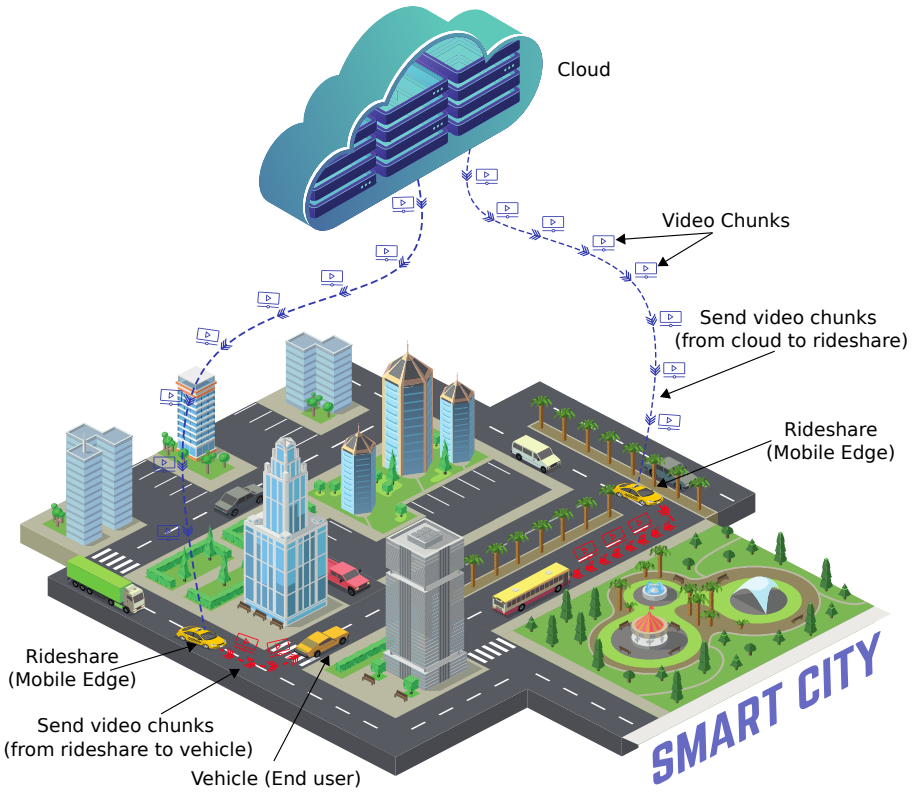


Fig. 3.17 MVEC architecture overview [8].

Problem Formulation & Proposed Solution

We used the prediction technique in Section 3.1.3 to estimate the future positions of vehicles inside Rome city. In addition, the formulation of the proposed model is based on the set cover problem (SCP) that is used to maximize the coverage (shared path distance between the taxis and the client vehicle) inside the city. The problem formulation is as follows:

Let $S = \{s_1, s_2, s_3, ..., s_n\}$ a set that represents the taxis routes, $E = \{e_1, e_2, e_3, ..., e_m\}$ the representation of the client vehicle route as a set of segments e where $\bigcup S = E$, and $H = \{h_1, h_2, h_3, ..., h_4\}$ a set of handovers correspond to taxis route. The objective is to find a subset X of S where $\bigcup X = E$ and $|X|$, $h(\bigcup X)$ are small.

$$\min \sum_{i=1}^{n} x_i h_i \tag{3.46}$$

Subject to:

$$\sum_{\{i|e_j \in s_i\}} x_i \geq 1 \quad \forall e_j \in E \tag{3.47}$$

$$x_i \in \{0,1\}, \forall i \in \{1...n\} \tag{3.48}$$

• **Complexity:**

The problem formulation is a variant of the SCP, that allows maximizing the coverage of elements. The entry of the problem is a set of elements, a list of subsets of this set, and an integer k, and one must find k subsets such as the number of elements belonging to at least the subset. one of these is maximized. An element is said to be covered if it belongs to one of the selected subsets. The SCP model is one of the Karp's 21 NP-complete problems [117], for this, an approximate algorithm is proposed in the next chapter to support the dense networks.

Performance Evaluation

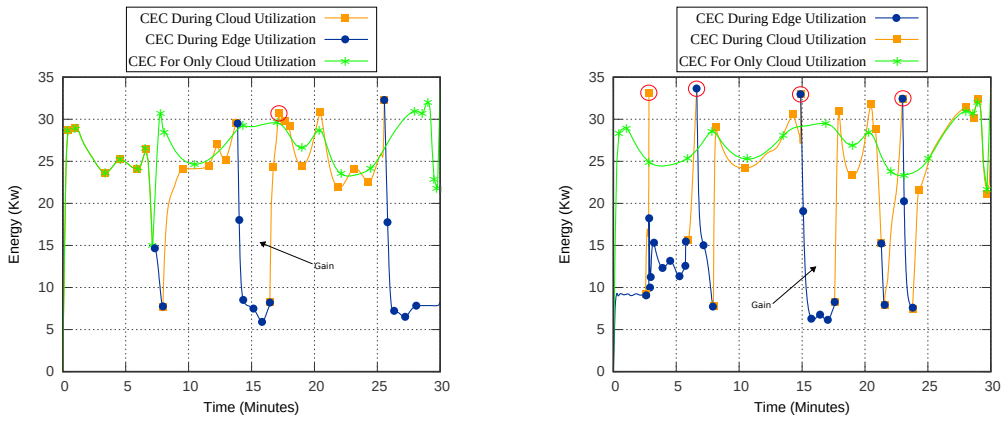
The performance evaluation of the proposed MVEC is evaluated in a simulated environment. We used Cloud Report[4] to simulate communications with the cloud, and SUMO[5] for urban mobility, and real tracking inside Rome city. Table 3.8 outlines the parameters used in the simulation.

Table 3.8 MVEC parameters.

Parameter	***Value***
Execution time	30 minutes
Number of taxis per zone	50
Number of client vehicles per zone	200

Figure 3.18 shows the cloud energy consumption (CEC) with/out edge servers for two different vehicle routes. The results are presented for three different cases. First, *CEC for only cloud utilization (OCU)*, represents the use of only the cloud as a source of video chunks. Second, *CEC during cloud utilization (CU)*, represents the use of cloud with the edge sometimes. Finally, the *CEC during edge utilization (EU)*, represents the use of edge as a source of video chunks.

[4]Cloud Report: www.github.com/thiagotts/CloudReports
[5]SUMO: www.sumo.dlr.de/docs/index.html



(a) Route 1 (b) Route 2

Fig. 3.18 CEC with/out Edge Servers [8].

In Figure 3.18a we show that the energy consumption for both CU and OCU is the same in the first 14 minutes because no taxis coverage exists. Then, the energy is decreased when the taxis (edge servers) are used which increases the energy gain.

Similarly, in Figure 3.18b we show that the taxis with high coverage leads to increase the gain of energy. Also we show that when no taxis coverage exists, the energy consumption of cloud increase (red circles in Figures 3.18a and 3.18b) caused by the handover during the communication which leads to send many video requests to the cloud.

3.2.2 Use Case 2: Edge Computing Aided Autonomous Vehicles

We propose to use edge computing for autonomous driving where the autopilot VNFs are offloaded to the edge for efficient self-driving. The proposed solution is summarized as follows:

1. We propose a reliable, end-to-end, and low-latency communication edge autopilot architecture that allows the offloading of autopilot VNFs services to edge servers to improve the driving efficiency for autonomous vehicles.

2. We propose a communication protocol for dense moving vehicles.

3. The optimization technique is used to model the edge-assisted autonomous driving for optimal autopilot VNFs allocation on edge servers.

Proposed Edge Autopilot Architecture

The proposed architecture for edge autopilot consists of three layers as follows:

1. **Centralized Cloud Layer**: represents the cloud autopilot, is responsible for processing a part of the autopilot VNFs.
2. **Distributed Edges Layer**: consists of edge servers responsible for autopilot VNFs analyzing and processing.
3. **Autonomous Vehicles Layer**: represents the autonomous vehicles that request autopilot VNFs services offloading because of the scarcity of local resources.

Proposed Edge Autopilot Protocol

1. **Autopilot VNFs**: the autonomous vehicles can offload some autopilot VNFs for processing. It requests the edge server in the near gNB or RSU to enable edge resource discovery and allocation.
2. **Resources Discovery in Edge Servers**: when the gNB receives the autopilot VNFs, it selects the edge servers that satisfy the VNFs requirement in terms of computing resources and network capabilities.
3. **Autopilot VNFs Offloading**: when the edge servers are selected, the gNB starts the allocation process by placing each VNF in the optimal edge server. Still, the cloud may represent the alternative solution in the case of a lack of resources at the network edge to ensure the autopilot VNFs processing.
4. **Vehicle-Edge/Cloud Connection**: in the last step, the gNB forwards the control commands to the autonomous vehicle by creating a communication link between the edge/cloud and the autonomous vehicle while satisfying its requirements.

Problem Formulation for Edge Autopilot

We proposed an optimal model for autopilot VNFs placement (OVEAP) based on the optimization technique. It takes as input the computing resources for each edge server. Then,

it allows to allocate optimally the autopilot VNFs upon the edge servers. The autopilot VNFs are offloaded to the edge server to reduce the processing time and enhance driving safety. The OVEAP model is formulated as follow:

1. OVEAP Parameters and Decision Variables

   Table 3.9 outlines the parameters and decision variables of the proposed OVEAP model.

Table 3.9 OVEAP Parameters and Decision Variables

Notations	***Definition***
E	The set of Edge Autopilots in terms of services chain.
V	The set of Edge Autopilot VNFs. Each Edge Autopilot VNF is composed of slices.
A	The set of Autonomous Vehicles.
M	The set of MEC servers at the network edge.
L_{ea}	The number of VNFs in each Edge Autopilot $ea \in E$. It represents the length of the Edge Autopilot.
G^{mec}	The maximum computing capacity vGPU available in the MEC server $mec \in M$.
$g_{ea,vnf}$	Required vGPU resources for the VNF slices vnf of the Edge Autopilot services chain ea.
Decision variables	***Definition***
$a^{mec}_{ea,vnf}$	A binary variable that allocates the Edge Autopilot VNF $vnf \in V$ of the Edge Autopilot $ea \in E$ to the MEC server $mec \in M$.
b^{mec}	A binary variable that indicates if the MEC server $mec \in M$ is used to process the edge autopilot VNFs.

The binary variable a indicates the allocation of the autopilot VNF (ea, vnf) on the MEC server mec.

$$a^{mec}_{ea,vnf} = \begin{cases} 1 & \text{if the VNF } vfn \text{ of the autopilot service chain } ea \\ & \text{is allocated on the MEC server } mec \\ 0 & \text{Otherwise} \end{cases} \tag{3.49}$$

Further, the binary variable y is used to track the MEC server usage. The variable formulation is as follows:

$$b^{mec} = \begin{cases} 1 & \text{if the MEC server } mec \in M \text{ is used} \\ 0 & \text{Otherwise} \end{cases} \tag{3.50}$$

2. OVEAP Model Formulation

The OVEAP objective function (3.51) allows to maximize the autopilot VNFs placement in the edge servers. The general model formulation is as follows:

$$\max \sum_{ea \in E} \sum_{mec \in M} a^{mec}_{ea,L_{ea}} - \sum_{mec \in M} b^{mec} \tag{3.51}$$

Subject to

$$\sum_{mec \in M} a^{mec}_{ea,vnf} \leq 1, \quad \forall ea \in E, vnf \in V \tag{3.52}$$

$$\sum_{ea \in E} \sum_{vnf \in V} g_{ea,vnf} \times a^{mec}_{ea,vnf} \leq G^{mec} b^{mec}, \quad \forall mec \in M \tag{3.53}$$

$$\sum_{mec \in M} a^{mec}_{ea,vnf} = \sum_{mec \in M} a^{mec}_{ea,vnf+1}, \forall ea \in E, \quad vnf \in V \setminus \{L_{ea}\} \tag{3.54}$$

$$b^{mec} \leq \sum_{ea,vnf} a^{mec}_{ea,vnf}, \forall mec \in M \tag{3.55}$$

$$0 \leq a^{mec}_{ea,vnf} \leq 1, \forall mec \in M, ea \in E, vnf \in V \tag{3.56}$$

$$0 \leq b^{mec} \leq 1, \forall mec \in M \tag{3.57}$$

The model constraints are detailed as follows:

Constraint (3.52) guarantee that the VNF slices $vnf \in V$ of the autopilot chain $ea \in E$ is placed on a unique MEC server $mec \in M$. Constraint (3.53) ensures that the vGPU resources in the selected MEC server is enough to process the VNF slices of the autopilot chain. Constraint (3.54) ensure the respect of VNFs chaining during the process in the edge servers, where the allocation of one VNF should be done completely in the edge server to allocate the next VNF in the chain. Constraint (3.55) are formulated to select the minimum number of edge servers for the VNFs allocation. Finally, constraints (3.56) and (3.57) guarantee the non-negativity of the decision variables.

3. OVEAP complexity

   OVEAP is NP-hard which is feasible only with a few instances within a reasonable time in small-scale networks. For this, a suitable algorithm for dense networks is presented in the next chapter.

Edge Autopilot Performance Evaluation

We considered different autopilot VNFs: *vPerception*, *vLocalisation*, *vPlanner*, and *vControl*. The service chain is composed of a set of autopilot VNFs. The objective is to place autopilot services in the edge server while guaranteeing the VNFs chaining. Tables 3.10 and 3.11 show the different configurations used in the evaluation.

Table 3.10 Autonomous Vehicles Configuration.

Autonomous Vehicles	***Autopilot Chain***
Vehicle 1	*vPerception*
Vehicle 2	*vPerception, vLocalisation*
Vehicle 3	*vPerception, vLocalisation, vPlanner*
Vehicle 4	*vPerception, vLocalisation, vControl*
Vehicle 5	*vPerception, vLocalisation, vPlanner, vControl*
Embedded Autopilots	***Number of servers***
Embedded vehicle	1

1. Key Performance Indicators (KPIs) To evaluate the proposed model, we proposed the following KPIs:

Table 3.11 Summary of autopilot edge servers (AES) parameters.

Parameter	*Min value*	*Max value*
Number of AES	3	5
Number of Autonomous vehicles	5 (small scale)	100
Available vCPU in AES	50 (slot)	200 (slot)
Available vGPU in AES	10 (slot)	150 (slot)
Available vRAM in AES	50 (Gigabytes)	200 (Gigabytes)
Available vStorage in AES	10 (Terabytes)	100 (Terabytes)
Available Bandwidth in AES	10 (Mbps)	150 (Mbps)
Latency in Autopilot Chain	1/8 (ms)	1 (ms)

(a) **Average Network Delay**: it represents the network delay between the edge servers and the autonomous vehicles.

(b) **Service Time**: represents the end-to-end time to accomplish the service from the autopilot VNF submission to the results forwarding to the autonomous vehicle. It is depend on the offloading decisions and availability of resources where an efficient service has low execution time.

(c) **Processing Time**: represents the duration needed to complete the execution of the autopilot VNFs. In general, the processing time is related to the availability of resources in edge servers.

2. Comparative Analysis

To assess the behavior of the proposed OVEAP at the network edge, we compared three different computing architectures. We summarize the architecture as follows:

(a) **Embedded Computing:** allows the processing of autopilot VNFs in the autonomous vehicle (local processing) while the edge is still active to receive VNFs.

(b) **Edge Computing:** this architecture prioritizes the edge servers for autopilot VNFs processing.

(c) **Cloud Computing:** this architecture allows the use of the centralized cloud for autopilot VNFs processing.

3. Obtained Results

Figure 3.19 outlines the performance evaluation of the proposed approach.

In Figure 3.19a, We plot the network average delay between autonomous vehicles and the computing servers. The result shows that the delay is low for embedded computing because the processing happens locally, while the delay of the edge is lower than the cloud since the edge offers a short response time because is near to autonomous vehicles. In Figure 3.19b we show that edge reduces the service time compared to cloud architecture. Indeed, the heavy autopilot VNFs are offloaded to the near edge for efficient processing. In Figures 3.19c and 3.19d we show that the edge computing reduces the service/processing time compared to both cloud and embedded computing since the edge offer efficient processing service near to autonomous vehicles.

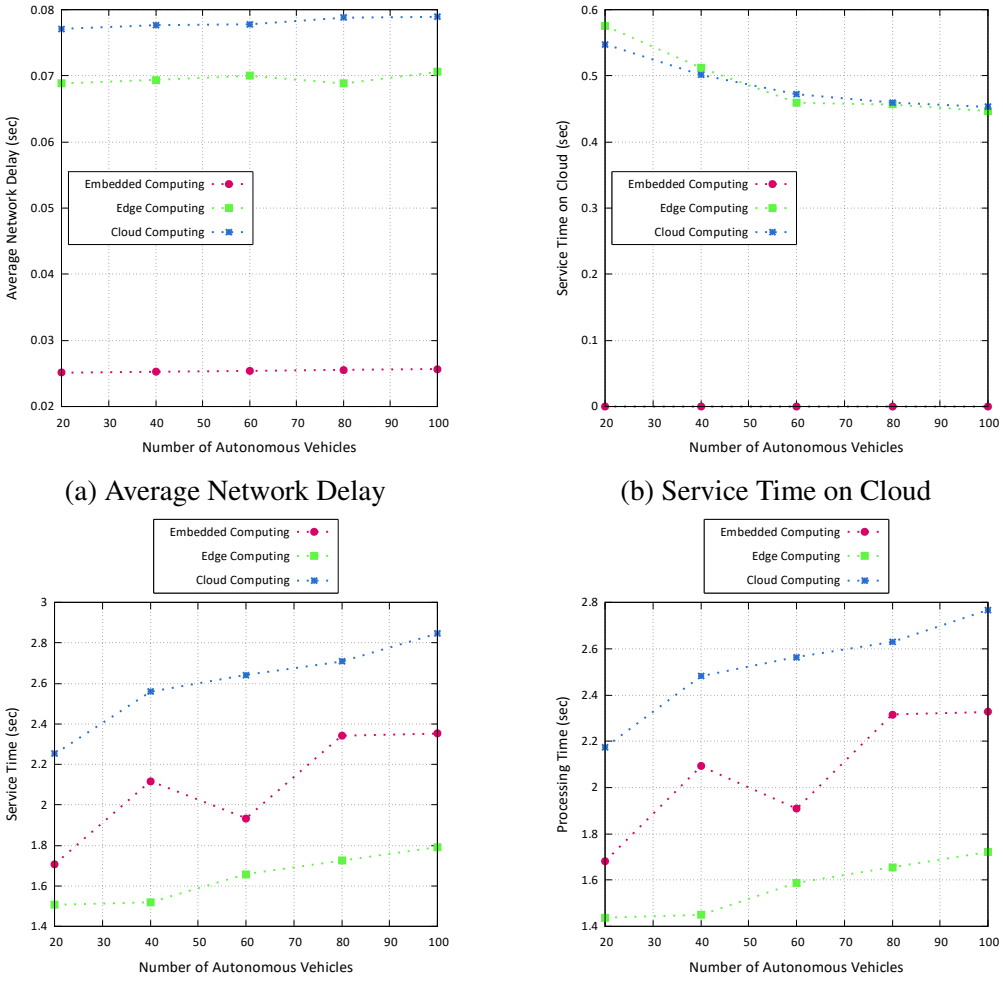
3.2.3 Use Case 3: Edge Computing Assisted Autonomous UAV

We propose to use autonomous mobile UAVs as mobile edge servers to offer video chunks to connected vehicles in vehicular edge computing architecture. Each connected vehicle inside the city can receive video chunks from a UAV(s) that cover the vehicle route.

Proposed UAV-assisted V2X Architecture

Figure 3.20 display the proposed *Follow Me UAV (FMU)* architecture with the communication entities described as follow:

1. **Connected Vehicles**: represent the end-user layer consists of a high number of vehicles.
2. **Autonomous Mobile UAVs**: is an intermediate layer that represents the MEC servers responsible for reducing the communication load on the centralized cloud. It guarantees ultra low latency communication (ULLC) and safety for sensitive delay applications. In ULLC, UAVs serve the high dynamic connected vehicles according to their quality requests.
3. **The Centralized Cloud**: represents a set of servers that provide high computing and storing services for insensitive delay applications.



(a) Average Network Delay

(b) Service Time on Cloud

(c) Service Time

(d) Processing Time

Fig. 3.19 OVEAP Performance Evaluation [5].

FMU Optimization Algorithm

1. The Optimal FMU (OFMU) Algorithm The FMU objective is to find the shared path between UAVs and connected vehicles to maximize the vehicles coverage using available UAVs. For this, we propose an optimal model based on optimization technique. Table 3.12 displays the used notations in the FMU model.

   Let $S = \{s_1, s_2, s_3, ..., s_n\}$ represent UAVs routes as a collection of sets.

   Let $W = \{w_1, w_2, w_3, ..., w_n\}$ a collection of sets represent the energy consumed in each UAV routes.

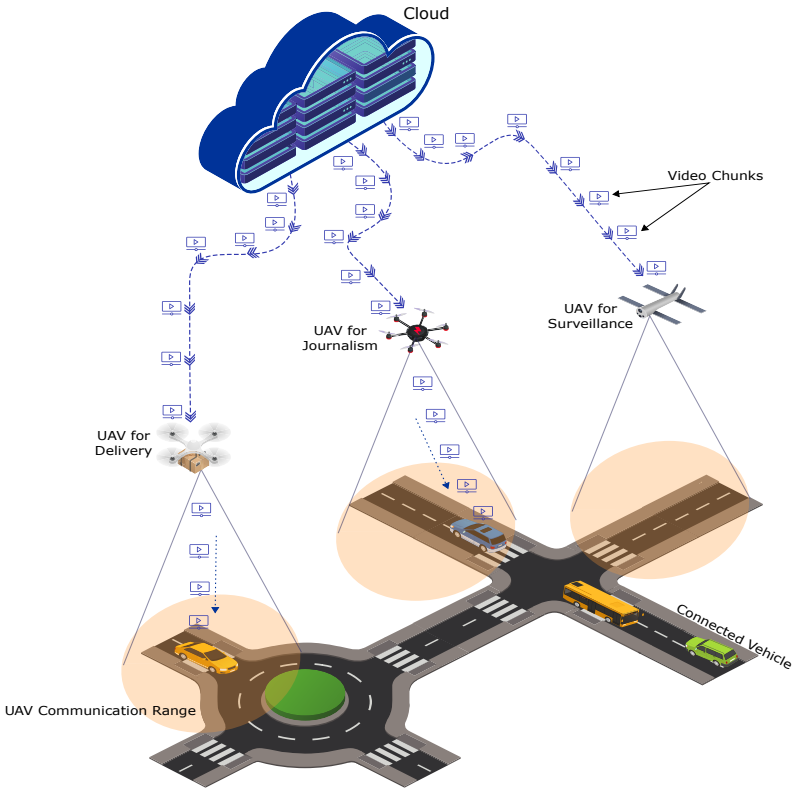


Fig. 3.20 Proposed FMU Architecture [3].

Table 3.12 The used notations in the FMU model.

Notations	Definition
C	Cloud service.
CV	The set of Connected Vehicles $cv \in CV = \{1..\|CV\|\}$
U	Set of connected UAVs in service.
$D_{v,u}$	Vehicle $v \in V$ consumes the content from $u \in U$
$\overline{C}_{v,c}$	Vehicle $v \in V$ is connected with $c \in C$
h^{uav}	The handover delay required by UAV $uav \in UAV$
w^{uav}	The energy consumption required by UAV $uav \in UAV$
M^{uav}_{max}	Maximum association links of the UAV $uav \in UAV$
θ	It represents the minimum threshold of the covered connected vehicles $cv \in CV$

$E = \{e_1, e_2, e_3, ..., e_m\}$ is a set of adjacent segments $e_i, i \in \{1..m\}$ represent the vehicle route where $\bigcup S = E$.

$H = \{h_1, h_2, h_3, ..., h_m\}$ a set of handovers of UAVs route.

The main objective is to find a subset X of S such that $\bigcup X = E$ and $|X|$, $h(\bigcup X)$ are small as possible while minimizing the UAVs energy consumption and the number of handover.

To clarify better the proposed OFMU algorithm, we show in Figure 3.21 an example of UAV selection to cover the vehicle route while minimizing the number of handovers and the UAVs energy cost. We have as an input the predicted vehicle's route using deep learning (detailed in the Section 3.1.3), and the UAVs routes with predefined energy cost for each UAV. The OFMU algorithm selects the most cost-effective UAVs according to the algorithm constraints.

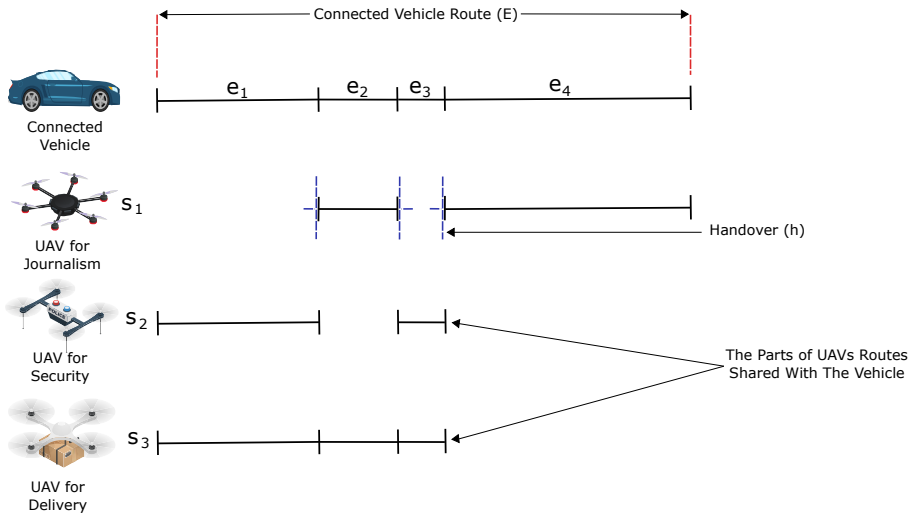


Fig. 3.21 Abstraction Model For FMU [3].

2. Decision variables

The binary variable x indicates the UAV usage. It is formulated as:

$$x^{uav} = \begin{cases} 1 & \text{if the UAV } uav \text{ is active} \\ 0 & \text{Otherwise} \end{cases} \tag{3.58}$$

The binary variable y represents the association link between the UAV and the vehicle. It is defined as follow:

$$y_{cv}^{uav} = \begin{cases} 1 & \text{if CV } cv \text{ is associated with UAV } uav \\ 0 & \text{Otherwise} \end{cases} \tag{3.59}$$

The binary variable z represents the communication links among UAVs. It is defined as follows:

$$z^{uav_1,uav_2} = \begin{cases} 1 & \text{if UAV } uav_1 \text{ is within UAV } uav_2\text{'s} \\ & \text{communication range} \\ 0 & \text{Otherwise} \end{cases} \tag{3.60}$$

3. The FMU exact formulation and constraints

The objective function of the problem is formulated as follows:

$$\min \sum_{uav \in U} x^{uav}(\alpha h^{uav} + (1-\alpha) w^{uav}) \tag{3.61}$$

Where α is a metric related to the operator strategy.

Moreover, the OFMU constraints are defined as follows:

Maximum Coverage: constraint (3.62) guarantees the maximum coverage of vehicles using UAVs:

$$\sum_{\{uav | e_j \in s^{uav}\}} x^{uav} \geq 1 \quad \forall e_j \in E \tag{3.62}$$

Minimum Energy: guarantees that each UAV must respect the maximum energy threshold authorized for mission achievement:

$$\sum_{uav \in U} x^{uav} \times w^{uav} \leq |N| \times W \tag{3.63}$$

The set W represents the maximum energy consumed for each UAV.

Minimum Handover Delay: This constraint guarantees that the UAV route handover must be lower as possible:

$$\sum_{uav \in U} x^{uav} \times h^{uav} \leq H \tag{3.64}$$

The set of handover H, is the number of handover for each UAV route.

Dynamic Association: this constraint ensures that the connection of connected vehicles is not permitted to deleted UAVs:

$$\forall uav, cv : y_{cv}^{uav} \leq x^{uav} \tag{3.65}$$

Single UAV-CV Association: ensures that a connected vehicle can be served by at most one UAV:

$$\forall cv \in CV : \sum_{uav \in UAV} y_{cv}^{uav} \leq 1 \tag{3.66}$$

UAV Association Capacity: represents the upper bound number of connected vehicles that can be covered by one UAV:

$$\forall uav : \sum_{cv \in CV} y_{cv}^{uav} \leq M_{max}^{uav} \tag{3.67}$$

QoE Constraints: this constraint ensures that the percentage of non successfully served vehicles should not exceed a predefined threshold θ. The constraint formulation is as follow:

$$\sum_{uav \in UAV} \sum_{cv \in CV} y_{cv}^{uav} \geq (1 - \theta) \times |CV| \tag{3.68}$$

Bi-connection: this constraint ensures the bi-connection between the UAVs:

$$\sum_{uav1, uav2 \in UAV} z^{uav_1, uav_2} \geq 2 \tag{3.69}$$

Non negativity: this constraint ensure the non-negativity of the used variable for the UAV usage:

$$x^{uav} \in \{0,1\}, \forall uav \in \{1...n\} \tag{3.70}$$

FMU Performance Evaluation

To assess the efficiency of the proposed approach we proposed different communication scenarios as follows:

1. **Low Quality Cloud (LQC):** The connected vehicles receive low-quality video chunks for live streaming from the cloud only.
2. **Low Quality Edge (LQE):** The UAVs (edge servers) are introduced as intermediate nodes to offer a low-quality video for live streaming to connected vehicles.
3. **Medium Quality Cloud (MQC):** The cloud offers medium-quality video for live streaming to connected vehicles.
4. **Medium Quality Edge (MQE):** The intermediate UAVs offer medium-quality video for live streaming to connected vehicles.
5. **High Quality Cloud (HQC):** The cloud offers high-quality video for live streaming to connected vehicles.
6. **High Quality Edge (HQE):** The UAVs offer high-quality video for live streaming to connected vehicles.

For the above cloud scenarios (LQC, MQC, and HQC), the vehicles are connected directly to the cloud without the use of edge servers (UAVs).

Table 3.13 outlines the FMU parameters.

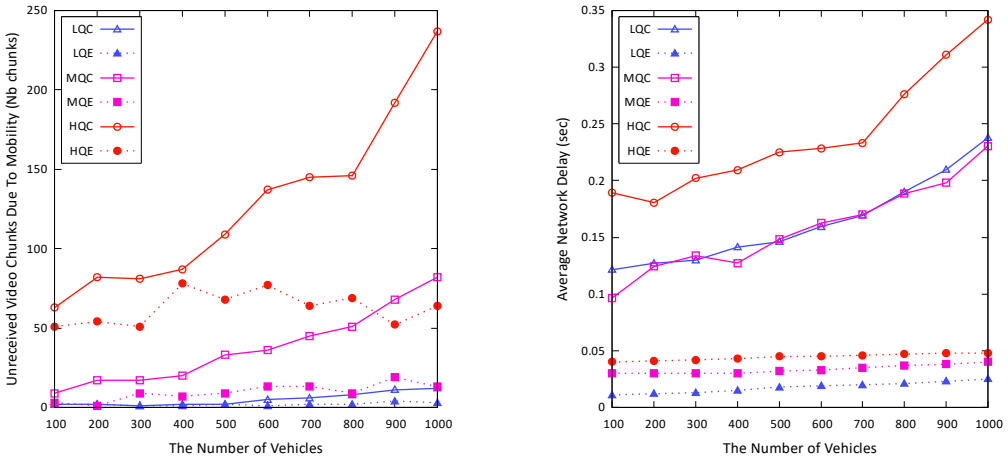
Figure 3.22a shows the lost video chunks due to mobility. From the figure, we show that the use of edge servers (UAVs) decrease the lost video chunks compared to the use of cloud only. This lead to increase both the user satisfaction index and the QoS. Moreover, from the obtained results, the use of suitable computing is recommended (cloud and/or edge) regarding the streaming quality and the Service Level Agreement between network operators and Over the Top (OTT) providers.

Table 3.13 FMU parameters.

Parameter	Value
Max Data Download	4000 Megabytes
Number of UAVs per zone (Min/Max)	(2/10)
Number of Vehicle (Min/Max)	(100/1000)
Max Low Quality Video Size	200 Megabytes
Max Medium Quality Video Size	400 Megabytes
Max High Quality Video Size	600 Megabytes

Figure 3.22b, we measure the average network delay for the proposed scenarios against the number of vehicles. The result show that the UAV edge layer reduces the network delay which enhances the quality of service. We remark that all the edge based scenarios (LQE, MQE, and HQE) gives less network delay comparing to the cloud based scenarios (LQC, MQC, and HQC). Therefore, the results suggest always the use of UAV as mobile edges.

Figure 3.22b shows the average network delay for different video quality. The results show that the use of UAV as edge servers reduce the delay compared to the use of cloud only because the content (video chunks) is near to connected vehicles which leads to decrease the delay and enhances the QoS.



(a) Unreceived Video Chunks. (b) Average Network Delay.

Fig. 3.22 System Evaluation for Different Communication Scenarios [3].

Conclusion

We proposed different optimal models based on optimization techniques to enhance the QoS and improve user satisfaction according to the offered services. We studied two different communication technology-based edge computing called *service offloading* and *vehicular edge computing*, where for each technology we proposed optimal models each one is dedicated to a problem. The common factor between all the proposed models is the high complexity that restricts the use of these models. This makes its usability in only small-scale networks with a limited number of devices. For this, approximate algorithms are proposed in the next chapter to support the huge amount of requests and connected users in large-scale networks.

Chapter 4

Large Scale Solutions For Edge Computing Networks

Large-scale algorithms are introduced as the alternative solution for the optimal solutions in dense networks due to their feasible execution time and the given efficient solutions. Different large-scale algorithms are proposed by the researchers to resolve several networks problems such as services placement at the edge servers taking into account the network, storage, and computing resources constraints. Most of these algorithms proved their efficiency in dense networks with a huge amount of produced data and users demands.

4.1 Service Offloading In EC

4.1.1 Use Case 1: 5G VNF Slices Placement in EC

We proposed to use *Reinforcement Learning (RL)* and *Deep Reinforcement Learning (DRL)* techniques for large scale VNF slices placement in edge computing.

The RL algorithm

In this part, we present our proposed scheme for online VNF slices placement with reinforcement learning. We formulate the problem (3.3) and represent it as a RL model.

Q-learning is the main algorithm used for the RL model. The Q-function allows estimating the future VNF placement reward. We define hereafter the proposed model.

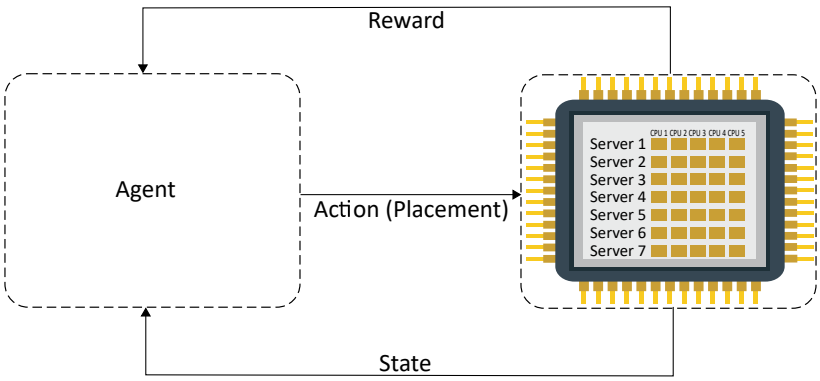


Fig. 4.1 RL application to VNF placement in Edge Computing [7].

1. The proposed model

   We consider a cluster with the CPU as a computing resource. The incoming VNF slices to the cluster are jobs that arrive in discrete time steps. The cluster manager places the VNF in the server according to the VNF size and the available CPU slots in the server. Figure 4.1 show the RL application to VNF placement in edge computing.

2. The proposed formulation

   We represent the system state as the current VNF slices placement in the server slot. Figure 4.1 (right side) display the proposed state-space (edge server) that is considered as the environment.

   The action represents the placement function that takes the decision on the efficient placement of the VNF slice by taking into consideration the server capacity in terms of CPU slots. Moreover, the agent can not place a VNF slice in an occupied server slot. The action allows the agent to process one by one the incoming VNFs until process all the arrived requests.

   The proposed reward represents the placement cost of VNFs. It is calculated based on the used server after performing an action. It is defined as follows:

$$R_t = \begin{cases} 100 \times i & \textit{if } i \textit{ servers are opened} \\ 0 & \textit{Otherwise.} \end{cases} \tag{4.1}$$

As shown in the equation 4.1, the main objective is to minimize the total used servers. In other words, the objective of the agent is to minimize the total discounted cumulative rewards.

In Algorithm. 1 we present the proposed RL algorithm.

Algorithm 1 : RL-driven VNF slices placement in Edge Computing Over 5G Network [7].

Input: Servers, the number of slots per Server (CPUs or GPUs), the number of VNFs, and the number of slices per VNF
Output: $Q^*(state(s), action(a))$
Initialise Q(state(s),action(a)) arbitrary.
Observe an initial state s (current allocation and first incoming VNF slice)
repeat
    Select and place a VNF slice on the edge computing server
    r, $s' \leftarrow$ Observe the placement cost r and the new state s' (new allocation and another incoming VNF slice)

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \times \max_{a'} Q(s',a') - Q(s,a)) \tag{4.2}$$

    s = s'
until no incoming VNF slices

The DRL algorithm

In the DRL algorithm, the deep neural network is used to approximate the RL algorithm. A succession of neural network layers are used to map the input to the output (state to action). The Stochastic Gradient Descent (SGD) algorithm [118] is used in the deep neural network to train the deep q-learning network (DQN) [119]. The objective of the DRL model is to reduce the complexity of both the exact model (3.3) and the RL model by reducing the iterations to be considered in the optimization. Algorithm 2 presents the proposed DRL algorithm.

Performance Evaluation

To assess the performance of the proposed algorithms, we evaluate three key performance indicators: *placement time*, *server utilization*, and *energy consumption*.

For each environment, we present a set of parameters. First, servers configuration as mentioned in table 4.1. Then, we specify the VNF configuration as depicted in table 4.2.

Figure 4.2 shows the performance evaluation of the proposed algorithms: the exact ILP model (3.3), Q-learning, and DQN in dense networks.

Algorithm 2 : DRL-driven VNF slices placement in Edge Computing Over 5G Network [7].

Input: Servers, the number of slots per Server (CPUs or GPUs), the number of VNFs, and the number of slices per VNF
Output: $Q^*(state(s), action(a))$
Initialize a reply memory D
Initialise action-value Q with random weights
Observe initial state s
repeat
    Select an edge computing server a .

- with probability ε select a random edge computing server
- Otherwise select the server that has the $\max_{a'} Q(s,a')$

    Place the VNF slice on the selected edge computing server a.
    $r, s' \leftarrow$ Observe the placement cost r and the new state s' (new allocation and another incoming VNF slice)
    store the experience $\{s,a,r,s'\}$ in the replay memory.
    sample a random transition from the replay memory.
    Calculate the target for each mini-batch transition $(r + \gamma \times \max_a Q(s',a'))$
    Train the Q network using the following loss

$$Loss = \frac{1}{2} * (r + \gamma \times \max_{a'} Q(s',a') - Q(s,a))^2 \tag{4.3}$$

    s = s'
until no incoming VNF slices

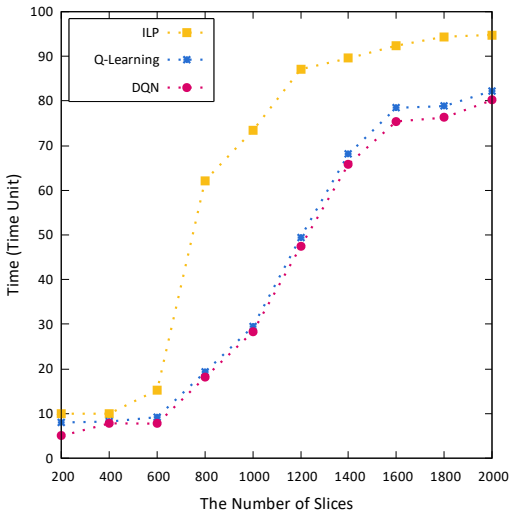
Table 4.1 Servers Configuration.

Environments	The number of servers	The number of cpu in servers (slots/servers)
Environment 1 (Small)	5	5/5
Environment 2 (Medium)	10	3/3, 4/3, 5/2, 6/2
Environment 3 (Large)	15	3/2, 4/3, 5/4, 6/3, 7/3

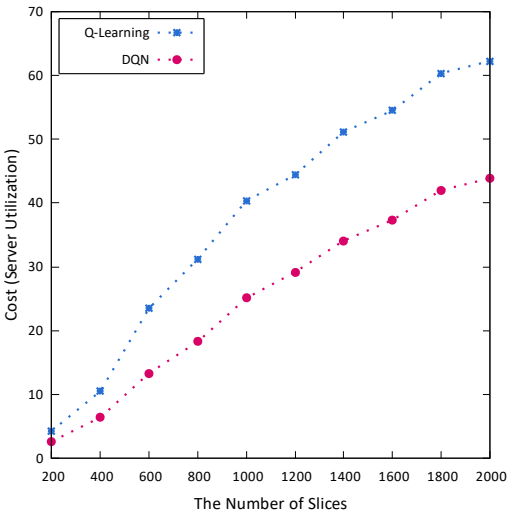
Table 4.2 VNF Configuration.

VNF Configuration	The number of vnf	The number of slices in vnf (slices/vnf)
Configuration 1	5	2/3, 3/2
Configuration 2	10	1/3, 2/3, 3/4
Configuration 3	15	1/8, 2/5, 3/1, 4/1

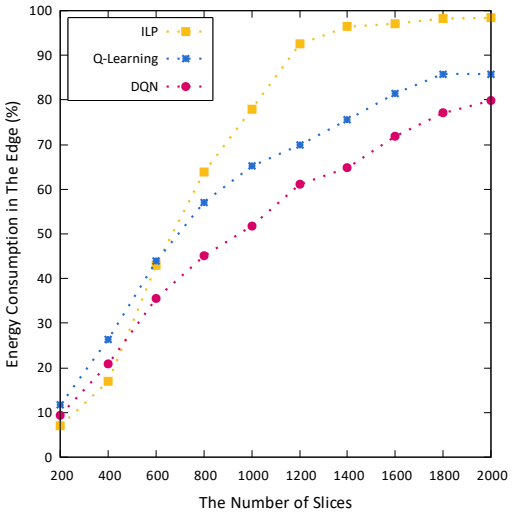
Figure 4.2a shows the placement time for the proposed algorithms. From the figure, we show that when the number of VNF increases, the placement time increase for the three models while the time for the ILP is very high compared to the other algorithms that place VNF efficiently in terms of time.



(a) Placement Time



(b) Server Utilization



(c) Energy Consumption

Fig. 4.2 Performance Evaluation of VNF Slices Placement in the edge using ILP, Q-Learning, DQN in Large Scale Networks [7].

Figure 4.2b display the server utilization for both Q-learning and DQN in dense networks. From the figure, we show that the DQN algorithm is more efficient compared to the Q-learning algorithm.

Figure 4.2c displays the energy consumption in the edge when using the proposed algorithms. We show that the ILP gives less energy consumption compared to both Q-learning and DQN when the number of VNF is less than 400. While the energy is very high for the ILP compared to the other algorithms when the number of VNF is higher than

400 because the ILP uses high computation resources. Moreover, the DQN is an efficient algorithm with lower energy consumption.

4.1.2 Use Case 2: Service Offloading in Virtual Mobile Edge Computing (SO-VMEC)

Deep Reinforcement Learning Model for SO-VMEC

The exact optimization algorithm is a high-cost model in terms of computation resources, which is not adapted for dense networks. For this, we use the deep reinforcement learning (DRL) to solve the exact model (3.26) that places the VNF slices in virtual edge servers.

1. The Proposed DRL Algorithm

   We used the equation $Q(s,a) = r(s) + \gamma \times \max_{a'} Q(s',a')$ to update the state-action pairs. Then, the policy equation $\pi(s) = \max_a Q(s,a)$ is used to select the action. By using deep learning, the Q-values are updated based on the loss value in $\min_\theta \; || \; r + \max_{a'} Q(s',a') - Q(s,a;\theta \; ||_2 + || \; \lambda \times \theta \; ||_2$ where r represents the reward, and θ represents the shared weights in the neural network. Deep Q-learning (DQN) represents a DRL instance. It is the brain of the proposed SO-VMEC where the agent is responsible for the placement process.

   Figure 4.3 display the DRL process, The main objective is to place the VNF slices in the VME.

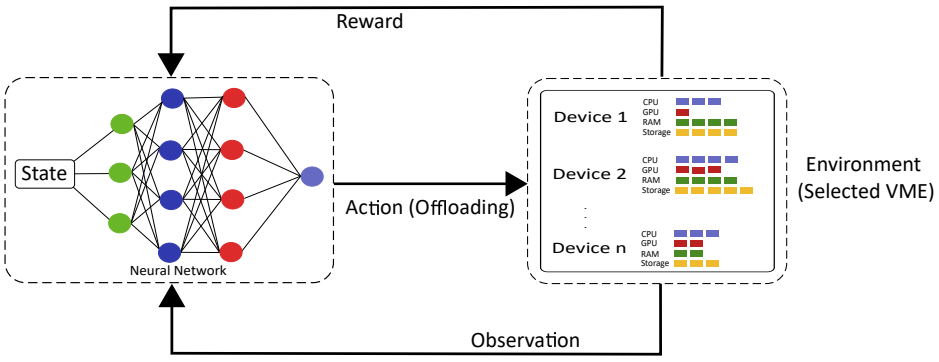


Fig. 4.3 Deep Reinforcement Learning for VNF Slices Offloading [1].

In the algorithm 3, we present the detail of the proposed DRL model.

Algorithm 3 : Deep SO-VMEC Algorithm for Service Offloading Optimization [1].

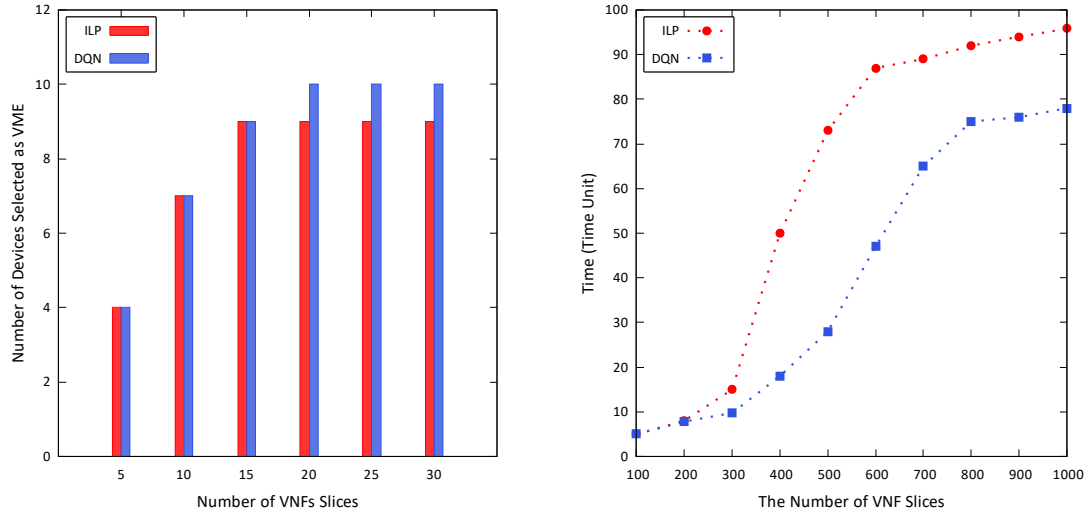
Input: The big network resources image (i.e., environment settings)
Output: $Q^*(state(s), action(a))$

Do a feed-forward pass for the current states to get predicted Q-values for all actions.
Do a feed-forward pass for the next states and calculate maximum overall network outputs $max_{a'}Q(s',a')$.
Set Q-value target for action to $r+\gamma \times \max_{a'} Q(s',a')$ (use the max calculated in step 2). For all other actions, set the Q-value target to the same as originally returned from step 1, making the error 0 for those outputs.
Update the weights using back-propagation. The loss function that should be minimized is $||\ (r+\max_{a'} Q(s',a') - Q(s,a)^2\ ||_2$

To simplify the behavior of the DQN algorithm during the execution, the agent must be able to match VNF slices to the server slots by minimizing the occupied servers.

Performance Evaluation

To assess the efficiency of the proposed approach, we compare the DQN algorithm with the exact ILP model (3.26) using the parameters mentioned in table 3.5. We measured both the number of selected VME and the time.



(a) The Number of Devices Selected in VME. (b) VME Selection Time.

Fig. 4.4 VME Based on ILP and DQN Models For VNFs Slices Offloading [1].

Figure 4.4 display the performance evaluation of the proposed SO-VMEC based on both DQN and ILP algorithms.

Figure (4.4a) displays the number of IoT devices in VME for different number of requests (VNF slices). The results prove the efficiency of the proposed DQN algorithm since it gave a solution equal/converge to the optimal ILP model.

Further, Figure (4.4b) shows the required time to select the VME for different VNF slices requests. The obtained results show that the DQN algorithm is efficient in terms of time in dense networks. In addition, the ILP is feasible since it gave a good selection time.

4.1.3 Use Case 3: Service Function Chains Orchestration in Virtual Mobile Edge Computing (VMEC)

The exact *OSPV* algorithm (3.36) is a high-cost model in terms of computation resources, which is no suitable for dense network. For this, we propose a heuristic algorithm for SFC placement in VMEC-IoT architecture.

ESPV: Efficient SFC Placement in VMEC over AI-IoT

The optimization targets a high number of SFC instances, where the objective relies on the Bin Packing problem [120]. The proposed algorithm allows satisfying the large SFC instances to deal with dense networks. it is noteworthy that the optimal *OSPV* algorithm could not be used for dense networks due to its complexity.

Algorithm 4 show the ESPV details.

Algorithm 4 : Efficient SFC Placement in VMEC (ESPV)

Input: $S_c, V_s, G_{nb}, V, Cloud$
Output: Z
repeat
  repeat
    Predict mobility sequences (x,y) of ve $\in V$ using deep learning (Section 3.1.3)
    Predict energy values of ve $\in V$ using deep learning (Section 3.1.3)
  until All servers *ve* in V are verified.
  repeat
    Check future positions from the prediction of *ve*.
    Check resource availability (vCPU, vRAM, vGPU, vStorage, and Energy) in the server *ve*.
  until Find a server *ve* to place the *vs* (True or False).

  if (resource available in *ve*) **then**
    Place the VNF slices vs on the server ve.

  else
    Offload the VNF slices vs to the Cloud.
  end if
until no incoming SFC (VNFs).

Performance Evaluation

To measure the efficiency and feasibility of the proposed algorithm, we propose different system parameters to cover real-world scenarios as shown in the table 4.3.

Table 4.3 Summary of the parameters.

Parameter	*Min value*	*Max value*
Number of gnb	3	5
Number of IoT	3	10
Available vCPU in IoT	5 (slot)	20 (slot)
Available vGPU in IoT	1 (slot)	15 (slot)
Available vRAM in IoT	20 (Megabytes)	2000 (Megabytes)
Available vStorage in IoT	50 (Megabytes)	20000 (Megabytes)
Available Energy in IoT	25 (Energy unit)	100 (Energy unit)

1. Key Performance Indicators (KPIs)

   To study the performance of the proposed algorithm, we propose to measure the following KPIs:

   (a) **The Total Allocated Servers**: represent the number of used servers (IoT devices) in the VME.

   (b) **The Placement Time** : represent the required time for SFC VNFs placement.

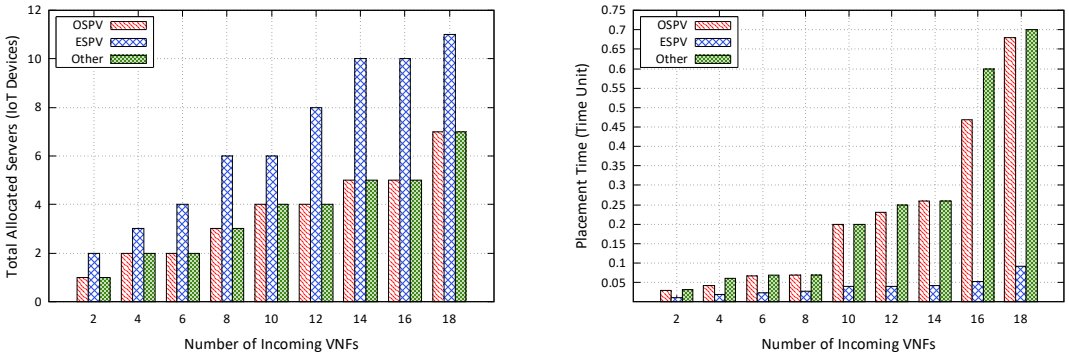
2. Comparative Analysis

   To assess the performance of the proposed approach, we compare the proposed **ESPV** algorithm with our **OSPV** exact model and the exact model (other) in [121].

3. Obtained Results

   Figure 4.5 displays the total allocated servers and the required time for the **EPSV** and **OPSV** compared with other algorithm.

   Figure 4.5a shows the total servers utilization for SFC VNFs placement, we show that both the exact algorithms (OPSV and Other) give equal results for servers allocation cost where the EPSV algorithm allocates more servers but is near the optimal model.



(a) VNFs Placement Cost (b) VNFs Placement Time

Fig. 4.5 Total Resources Utilization for Different Flow Rates of SFC VNFs.

Figure 4.5b displays the SFC VNFs placement time, we show that the OPSV gives a lower time compared with the Other algorithm which proves the efficiency our optimal OPSV model. Moreover, the ESPV algorithm gives a very low placement time compared to the exact models, since it provides an efficient placement cost with a lower time.

4.2 Vehicular Edge Computing

4.2.1 Use Case 1: Edge Computing Assisted Vehicular Networks for Service Offering

Rideshare Taxi Selection and Service Offering Algorithm

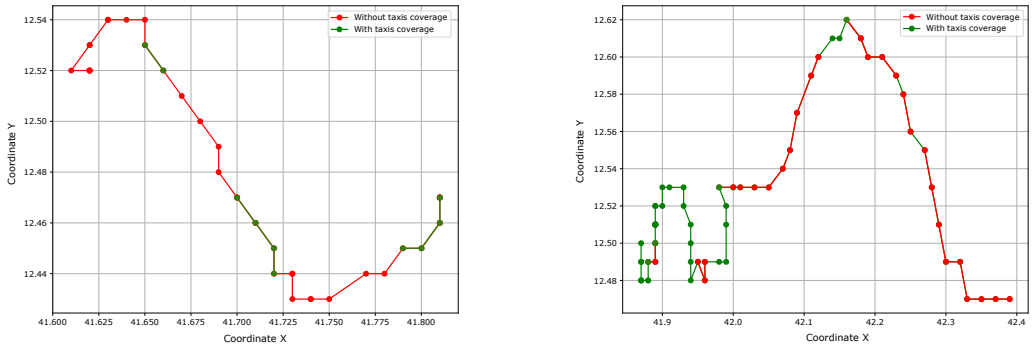
The formulated SCP problem (model 3.46) is NP-complete. For this, Algorithm 5 is proposed to deal with dense networks. it uses only a subset of the main set of taxi paths which by consequence reduces the execution time. This subset contains only the taxis that share part(s) of the route with the client vehicle, which minimize the data processing complexity.

Algorithm 5 : Maximum Coverage For Rideshare Taxi Selection [8].

Input: RideshareTaxisRoutesList, VehiclesRoutesList
Output: RideshareTaxisList
Find rideshare taxis that cover each vehicle route.
for (i in VehiclesRoutesList) **do**
  for (j in RideshareTaxisRoutesList) **do**
    if ($Intersect(i, j)$! = $Null$) **then**
      append j in RideshareTaxisList(i)
    end if
  end for
end for
return *RideshareTaxisList*.

Performance Evaluation

We defined a set of parameters for each proposed scenario such as the number of taxis and client vehicles where both of them use a predefined route to reach the destination. The route of each vehicle is represented as a connected road segments in Rome city.



(a) Taxis Coverage For Route (1)

(b) Taxis Coverage For Route (2)

Fig. 4.6 Real Routes With Taxis Coverage [8].

Figure 4.6 shows the real routes with taxis coverage in two different cases. In the figure 4.6a we show that there are only some few coverage of taxis in short distance, where in Figure 4.6b we show that the coverage of taxis is large in long distance.

4.2.2 Use Case 2: Edge Computing Aided Autonomous Vehicles

The AI-defined optimization tries to be an alternative solution for exact algorithms by the integration of AI models in the edge to facilitate the analysis and the process of data especially

for the huge amount of data coming from different sources that need to be processed efficiently in a short time.

DVEAP: DRL-based autopilot Placement Algorithm

The AI-edge for autopilot VNFs placement provides many benefits including efficient data processing in a short time that is the important factor for autonomous driving applications, which by consequence improve the network performance and of course the QoS.

1. The Proposed RL Model

   We propose a distributed edge servers that process autopilot VNFs coming from autonomous vehicles. At each time, the edge manager chose the efficient server to place the VNFs. We propose an RL model to manage the VNFs placement at the network edge. The different RL steps are described as follows:

   The state space: the state s_t represents the current placement of autopilot VNFs on server slots.

   The action space: the action a_t represents the placement process of the autopilot VNF on the server taking into account the server capacity.

   The reward space: represent the placement cost r. It is measured as the number of used servers after the VNF placement, is formulated as follows:

$$r_t = \begin{cases} i & \textit{if i Edge Servers are Occupied} \\ 0 & \textit{Otherwise.} \end{cases} \tag{4.4}$$

   As shown in the equation 4.4, the objective of the agent is to minimize the used servers by reducing the total rewards.

2. The Proposed DRL Algorithm

   The deep neural network is used in the DRL algorithm by adding a succession of neural network layers to map the input to the output (state to action). The main objective of the DRL algorithm is to reduce the placement complexity compared to the optimal OVEAP model (3.51). In algorithm 6 we present the DRL details.

Algorithm 6 DRL-based autopilot Placement (DVEAP) at The Network Edge [5].

Input: $\mathscr{EA}$, $\mathscr{VNF}$, $\mathscr{L}_{ea}$, $\mathscr{MEC}$
Output: Q^* (state(s),action(a))
Initialize a reply memory D
Initialise action-value Q with random weights
Observe the initial state s
repeat
    Select a server a .

- with probability ε select a random server
- Otherwise select the server that has the $\max_{a'} Q(s,a')$

    Place the autopilot SFC's VNF on the Edge Computing Server a.
    r, $s' \leftarrow$ Observe the incurred allocation cost r and the new edge state s' (new allocation and another incoming autopilot SFC's VNFs)
    Store the experience $\{s,a,r,s'\}$ in the replay memory.
    Sample a random transition from the replay memory.
    Calculate the target for each mini-batch transition $(r+\gamma \times \max_a Q(s',a'))$
    Train the Q network using the following loss $Loss = \frac{1}{2} * (r+\gamma \times \max_{a'} Q(s',a') - Q(s,a))^2$
    s = s'
until No incoming autopilot VNFs from all the SFC

Performance Evaluation

In figure 4.7 we compare the DVEAP algorithm with the exact OVEAP model (3.51) in terms of offloading time using the parameters in Tab.3.10 and Tab.3.11. From the figure we show that the DVEAP decrease the offloading time compared with the exact OVEAP model which proves the efficiency of the DRL algorithm for autopilot VNFs offloading.

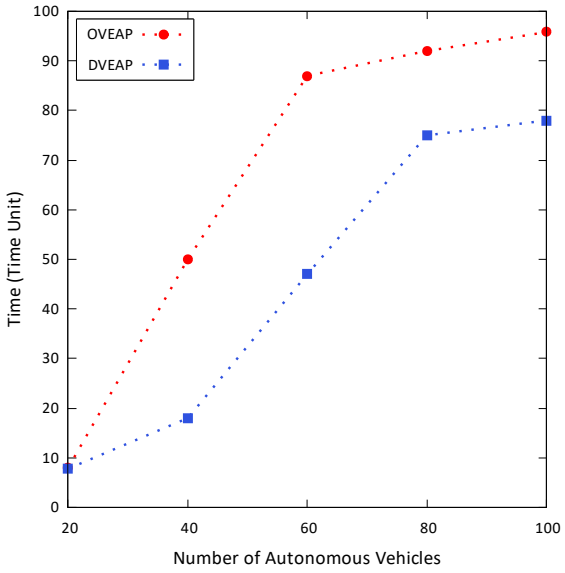


Fig. 4.7 OVEAP and DVEAP for autopilots VNFs Offloading [5].

4.2.3 Use Case 3: Edge Computing Assisted Autonomous UAV

The optimal OFMU model (3.61) is NP-hard which is not suitable for dense networks especially for the video streaming application that requires a short response time. For this, an approximate algorithm is proposed to deal with dense networks and support ultra-low latency applications.

The Approximate FMU (AFMU) algorithm

The proposed AFMU algorithm is based on the scalable weighted set cover problem (WSC) [122], it aim to select the UAVs with the high energy cost to satisfy the client vehicles requests. Algorithm 7 show the main steps of the AFMU solution where C represent the set of UAV, and α the average energy cost. The AFMU algorithm chooses the set which is the most cost-effective.

Algorithm 7 : Heuristic algorithm for FMU (AFMU) [3].

1: **Input:** C, E, S
2: **Output:** Picked UAV sets
3: $C \leftarrow 0$
4: **while** C != E **do**
5: Find the UAV whose energy cost effectiveness is smallest, say S
6: Let $\alpha = \frac{Cost(S)}{|S-C|}$
7: For each $e \in S - C$, set cost (e) = α
8: $C \leftarrow C \bigcup S$
9: **end while**

Table 4.4 show the complexity and the run-time of both OFMU and AFMU algorithms.

Table 4.4 OFMU and AFMU Algorithms Complexity.

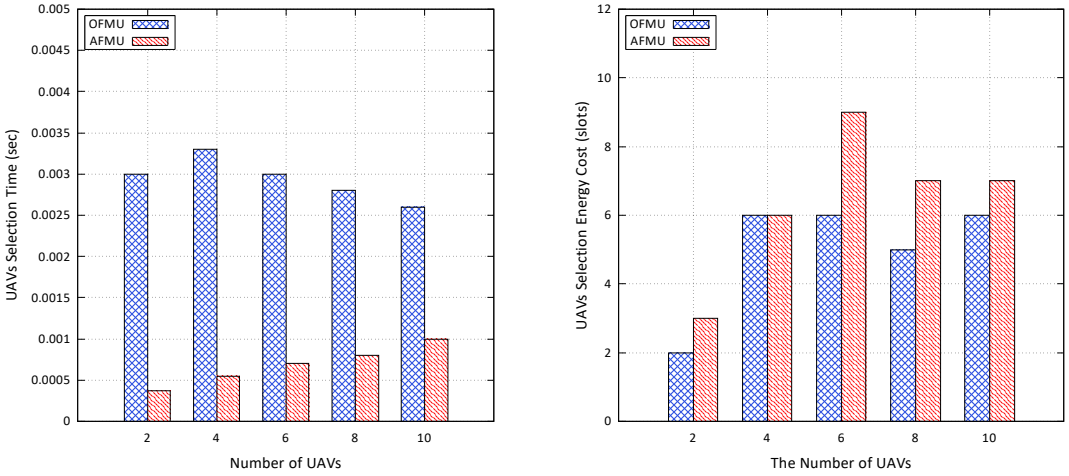
Metrics	***OFMU***	***AFMU***
Algorithm complexity	NP-Hard	$O(\|E\| * log\|S\|)$
Run-time	A few seconds in *Small Scale Network*; a few minutes in *Large Scale Network*	A few second in *Small/Large scale Network*

Performance Evaluation

To assess the performance of the proposed AFMU algorithm, we compare the obtained results with the optimal OFMU model (3.61). Figure 4.8 shows the algorithms evaluation for both selection time and energy cost.

Figure 4.8a displays that the OFMU algorithm is efficient because it requires a short time for UAV selection. In addition, the AFMU algorithm gives a good UAV selection time.

Figure 4.8b shows that the OFMU outperforms the approximate AFMU algorithm in terms of cost, while both are feasible for UAV selection.



(a) UAVs Selection Time.

(b) UAVs Selection Energy Cost.

Fig. 4.8 OFMU & AFMU Evaluation [3].

Conclusion

Large-scale algorithms are suitable for the high number of user requests in dense networks. It offers efficient solutions to deal with user satisfaction especially the new generation of applications that requires a short response time with efficient processing. We proposed different large-scale algorithms for both service offloading and vehicular edge computing to deal with dense networks. The common factor of these algorithms is the lower complexity compared to the optimal models, this makes these algorithms suitable for the high number of

devices and users' requests. It gives good results for different metrics such as time and cost, where it was efficient and feasible for all the studied problems.

Chapter 5

Conclusion and Perspectives

This thesis presents edge computing for the Internet of things technology. The first chapter presents the general introduction. We reviewed in the second chapter the studied technologies, Internet of things, cloud computing, mobile cloud computing, edge computing, and mobile edge computing. After that, we detailed the relation and the evolution of these technologies. The second part of the chapter presents a review of the related works of two major edge computing research domains, service offloading and vehicular edge computing.

The third chapter presents the proposed optimal models to optimize the edge usage for different metrics such as time and resource utilization. We proposed optimal models for tasks, VNFs, and SFCs orchestration at the network edge (VEnPA, SO-VMEC, OSPV) by taking into account the computing resources constraints. Moreover, we proposed other optimal models for vehicular edge computing including maximum coverage (MVEC) for online video streaming applications by using taxis as mobile edge servers inside the city, optimal edge-autopilot VNFs offloading at the network edge for autonomous driving (OVEAP), and UAV-edge model (OFMU) for online streaming by using autonomous UAV as mobile edge servers. The numerical results show that the proposed algorithm gives good results in terms of resource utilization at the network edge for small-scale networks.

In chapter four, to deal with dense networks we proposed large-scale algorithms that support a huge amount of devices, data, and user requests. Heuristic algorithms are proposed for SFC orchestration (ESPV), maximize the vehicles coverage (AFMU) by mobile edge servers (Taxis and UAV). Moreover, The artificial intelligence algorithms (Q-learning and Deep Q-learning (DQN)) are used for VNFs placement, edge-autopilot VNFs placement (DVEAP),

and autonomous UAV navigation. The obtained results prove the efficiency and the feasibility of the proposed solutions compared to optimal algorithms.

Despite the existing solutions outlined above, several concerns and challenges must be investigated in order to fully integrate edge computing on top of IoT applications. The following challenges examine the major issues, as well as new concepts and guidelines that the research community should take seriously in future works.

• Service availability must be ensured for the next generation of edge computing applications through the use of various mechanisms such as monitoring, prediction, and system backups.

• High mobility is an important factor in the Edge-IoT system because most connected devices, such as vehicles, drones, and mobile devices, are highly mobile, resulting in frequent service degradations and link failure between servers and devices which decreases the QoS of the Edge-IoT system. For this, efficient and developed models for obstacles detection and traffic prediction, etc. need to be proposed to overcome the mobility issue.

• The edge computing architecture is made up of several distributed systems, energy consumption is projected to be considerable, raising expenses. As a result, many efforts must be made to tackle this problem by developing efficient energy models in edge computing systems, particularly virtual and Adhoc edge systems, such as resource optimization, Relying more on environmentally friendly energy, etc.

References

[1] Mohammed Laroui, Hatem Ibn-Khedher, Moussa Ali Cherif, Hassine Moungla, Hossam Afifi, and Ahmed E Kamel. SO-VMEC: Service offloading in virtual mobile edge computing using deep reinforcement learning. *Transactions on Emerging Telecommunications Technologies*, page e4211, 2021.

[2] Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa A Cherif, Hossam Afifi, and Mohsen Guizani. Edge and fog computing for IoT: A survey on current research activities & future directions. *Computer Communications*, 180:210–231, 2021.

[3] Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Autonomous UAV Aided Vehicular Edge Computing for Service Offering. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2021.

[4] Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Artificial Intelligence Approach for Service Function Chains Orchestration at The Network Edge. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2021.

[5] Hatem Ibn-Khedher, Mohammed Laroui, Mouna Ben Mabrouk, Hassine Moungla, Hossam Afifi, Alberto Nai Oleari, and Ahmed E Kamal. Edge Computing Assisted Autonomous Driving Using Artificial Intelligence. In *IEEE International Wireless Communications and Mobile Computing (IWCMC)*, pages 254–259, 2021.

[6] Mohammed Laroui, Hatem Ibn Khedher, Hassine Moungla, Hossam Afifi, and Ahmed E Kamal. Virtual Mobile Edge Computing based on IoT devices resources in smart cities. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[7] Mohammed Laroui, Moussa Ali Cherif, Hatem Ibn Khedher, Hassine Moungla, and Hossam Afifi. Scalable and Cost Efficient Resource Allocation Algorithms Using

Deep Reinforcement Learning. In *IEEE International Wireless Communications and Mobile Computing (IWCMC)*, pages 946–951, 2020.

[8] Mohammed Laroui, Boubakr Nour, Hassine Moungla, Hossam Afifi, and Moussa Ali Cherif. Mobile vehicular edge computing architecture using rideshare taxis as a mobile edge server. In *IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–2, 2020.

[9] Mohammed Laroui, Aicha Dridi, Hossam Afifi, Hassine Moungla, Michel Marot, and Moussa Ali Cherif. Energy management for electric vehicles in smart cities: a deep learning approach. In *IEEE International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 2080–2085, 2019.

[10] Mohammed Laroui, Akrem Sellami, Boubakr Nour, Hassine Moungla, Hossam Afifi, and Sofiane B Hacene. Driving path stability in VANETs. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[11] Maria Stoyanova, Yannis Nikoloudakis, Spyridon Panagiotakis, Evangelos Pallis, and Evangelos K Markakis. A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches and Open Issues. *IEEE Communications Surveys & Tutorials*, 2020.

[12] Asif Ali Laghari, Kaishan Wu, Rashid Ali Laghari, Mureed Ali, and Abdullah Ayub Khan. A review and state of art of Internet of Things (IoT). *Archives of Computational Methods in Engineering*, pages 1–19, 2021.

[13] Yazdan Ahmad Qadri, Ali Nauman, Yousaf Bin Zikria, Athanasios V Vasilakos, and Sung Won Kim. The Future of Healthcare Internet of Things: A Survey of Emerging Technologies. *IEEE Communications Surveys & Tutorials*, 22(2):1121–1167, 2020.

[14] Koen Tange, Michele De Donno, Xenofon Fafoutis, and Nicola Dragoni. A systematic survey of industrial Internet of Things security: Requirements and fog computing opportunities. *IEEE Communications Surveys & Tutorials*, 2020.

[15] Ermal Elbasani, Pattamaset Siriporn, and Jae Sung Choi. A Survey on RFID in Industry 4.0. In *Internet of Things for Industry 4.0*, pages 1–16. Springer, 2020.

[16] Vedat Coskun, Busra Ozdenizci, and Kerem Ok. A survey on near field communication (NFC) technology. *Wireless personal communications*, 71(3):2259–2294, 2013.

[17] Mobeen Shahroz, Muhammad Faheem Mushtaq, Maqsood Ahmad, Saleem Ullah, Arif Mehmood, and Gyu Sang Choi. IoT-Based Smart Shopping Cart Using Radio Frequency Identification. *IEEE Access*, 8:68426–68438, 2020.

[18] Abdulrahman Abuelkhail, Uthman Baroudi, Muhammad Raad, and Tarek Sheltami. Internet of things for healthcare monitoring applications based on RFID clustering scheme. *Wireless Networks*, pages 1–17, 2020.

[19] Rahul Priyadarshi, Bharat Gupta, and Amulya Anurag. Deployment techniques in wireless sensor networks: a survey, classification, challenges, and future research issues. *The Journal of Supercomputing*, pages 1–41, 2020.

[20] Naveenkumar Jayakumar and Dhanashri P Joshi. Big Data & Disruptive Computing Platforms Braced Internet of Things: Facets & Trends. In *Internet of Things, Smart Computing and Technology: A Roadmap Ahead*, pages 119–150. Springer, 2020.

[21] Wan Haslina Hassan et al. Current research on Internet of Things (IoT) security: A survey. *Computer networks*, 148:283–294, 2019.

[22] Thomas Welsh and Elhadj Benkhelifa. On Resilience in Cloud Computing: A survey of techniques across the Cloud Domain. *ACM Computing Surveys (CSUR)*, 53(3):1–36, 2020.

[23] Keke Gai, Jinnan Guo, Liehuang Zhu, and Shui Yu. Blockchain Meets Cloud Computing: A Survey. *IEEE Communications Surveys & Tutorials*, 2020.

[24] Peter Mell, Tim Grance, et al. The NIST definition of cloud computing. 2011.

[25] Michael Cusumano. Cloud computing and SaaS as new computing platforms. *Communications of the ACM*, 53(4):27–29, 2010.

[26] Claus Pahl. Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3):24–31, 2015.

[27] Hamid Talebian, Abdullah Gani, Mehdi Sookhak, Ahmed Abdelaziz Abdelatif, Abdullah Yousafzai, Athanasios V Vasilakos, and Fei Richard Yu. Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues. *Cluster Computing*, 23(2):837–878, 2020.

[28] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700, 2016.

[29] Gabriel Neagu, Ştefan Preda, Alexandru Stanciu, and Vladimir Florian. A Cloud-IoT based sensing service for health monitoring. In *IEEE E-Health and Bioengineering Conference (EHB)*, pages 53–56, 2017.

[30] Leila Ismail and Huned Materwala. Energy-aware vm placement and task scheduling in cloud-iot computing: Classification and performance evaluation. *IEEE Internet of Things Journal*, 5(6):5166–5176, 2018.

[31] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future generation computer systems*, 29(1):84–106, 2013.

[32] Mazliza Othman, Sajjad Ahmad Madani, Samee Ullah Khan, et al. A survey of mobile cloud computing application models. *IEEE communications surveys & tutorials*, 16(1):393–413, 2013.

[33] M Reza Rahimi, Jian Ren, Chi Harold Liu, Athanasios V Vasilakos, and Nalini Venkatasubramanian. Mobile cloud computing: A survey, state of art and future directions. *Mobile Networks and Applications*, 19(2):133–143, 2014.

[34] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611, 2013.

[35] Jianli Pan and James McElhannon. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal*, 5(1):439–449, 2017.

[36] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[37] Latif U Khan, Ibrar Yaqoob, Nguyen H Tran, SM Ahsan Kazmi, Tri Nguyen Dang, and Choong Seon Hong. Edge computing enabled smart cities: A comprehensive survey. *IEEE Internet of Things Journal*, 2020.

[38] Zhuoqing Chang, Shubo Liu, Xingxing Xiong, Zhaohui Cai, and Guoqing Tu. A Survey of Recent Advances in Edge-Computing-Powered Artificial Intelligence of Things. *IEEE Internet of Things Journal*, 2021.

[39] Mohammad Aazam and Eui-Nam Huh. Fog computing: The cloud-iot\/ioe middleware paradigm. *IEEE Potentials*, 35(3):40–44, 2016.

[40] Mung Chiang, Sangtae Ha, Fulvio Risso, Tao Zhang, and I Chih-Lin. Clarifying fog computing and networking: 10 questions and answers. *IEEE Communications Magazine*, 55(4):18–20, 2017.

[41] Andrzej M Goscinski, Zahir Tari, Izzatdin A Aziz, and Eidah J Alzahrani. Fog computing as a critical link between a central cloud and iot in support of fast discovery of new hydrocarbon reservoirs. In *International Conference on Mobile Networks and Management*, pages 247–261. Springer, 2017.

[42] Yi Pan, Parimala Thulasiraman, and Yingwei Wang. Overview of cloudlet, fog computing, edge computing, and dew computing. In *Proceedings of The 3rd International Workshop on Dew Computing*, pages 20–23, 2018.

[43] Mahadev Satyanarayanan. The emergence of edge computing. *IEEE Computer*, 50(1):30–39, 2017.

[44] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In *IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78, 2015.

[45] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[46] Behailu Negash, Amir M Rahmani, Pasi Liljeberg, and Axel Jantsch. Fog computing fundamentals in the internet-of-things. In *Fog computing in the internet of things*, pages 3–13. Springer, 2018.

[47] Yi Qian, Dan Wu, Wei Bao, and Pascal Lorenz. The internet of things for smart cities: Technologies and applications. *IEEE Network*, 33(2):4–5, 2019.

[48] Dingfu Jiang. The construction of smart city information system based on the Internet of Things and cloud computing. *Computer Communications*, 150:158–166, 2020.

[49] Sandeep Saharan, Neeraj Kumar, and Seema Bawa. An efficient smart parking pricing system for smart city environment: A machine-learning based approach. *Future Generation Computer Systems*, 106:622–640, 2020.

[50] Wei Liu, Yoshito Watanabe, and Yozo Shoji. Vehicle-Assisted Data Delivery in Smart City: A Deep Learning Approach. *IEEE Transactions on Vehicular Technology*, 69(11):13849–13860, 2020.

[51] R Dhaya, R Kanthavel, Fahad Algarni, P Jayarajan, and Amita Mahor. Reinforcement Learning Concepts Ministering Smart City Applications Using IoT. In *Internet of Things in Smart Technologies for Sustainable Urban Development*, pages 19–41. Springer, 2020.

[52] Haidong Luo, Hongming Cai, Han Yu, Yan Sun, Zhuming Bi, and Lihong Jiang. A short-term energy prediction system based on edge computing for smart city. *Future Generation Computer Systems*, 101:444–457, 2019.

[53] Adam Zielonka, Andrzej Sikora, Marcin Woźniak, Wei Wei, Qiao Ke, and Zongwen Bai. Intelligent Internet-of-Things system for smart home optimal convection. *IEEE Transactions on Industrial Informatics*, 2020.

[54] Dae-Man Han and Jae-Hyun Lim. Design and implementation of smart home energy management systems based on zigbee. *IEEE Transactions on Consumer Electronics*, 56(3):1417–1425, 2010.

[55] Danni Yuan, Kaoru Ota, Mianxiong Dong, Xiaoyan Zhu, Tao Wu, Linjie Zhang, and Jianfeng Ma. Intrusion Detection for Smart Home Security Based on Data Augmentation with Edge Computing. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[56] Anila Yasmeen, Nadeem Javaid, Obaid Ur Rehman, Hina Iftikhar, Muhammad Faizan Malik, and Fatima J Muhammad. Efficient resource provisioning for smart buildings utilizing fog and cloud based environment. In *IEEE International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 811–816, 2018.

[57] Wei-Yu Chen, Po-Yu Chou, Chih-Yu Wang, Ren-Hung Hwang, and Wen-Tsuen Chen. Live Video Streaming with Joint User Association and Caching Placement in Mobile Edge Computing. In *IEEE International Conference on Computing, Networking and Communications (ICNC)*, pages 796–801, 2020.

[58] Shaojun Zhang, Wei Li, Yongwei Wu, Paul Watson, and Albert Zomaya. Enabling edge intelligence for activity recognition in smart homes. In *IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 228–236, 2018.

[59] Baofeng Ji, Xueru Zhang, Shahid Mumtaz, Congzheng Han, Chunguo Li, Hong Wen, and Dan Wang. Survey on the Internet of Vehicles: Network Architectures and Applications. *IEEE Communications Standards Magazine*, 4(1):34–41, 2020.

[60] Nhien-An Le-Khac, Daniel Jacobs, John Nijhoff, Karsten Bertens, and Kim-Kwang Raymond Choo. Smart vehicle forensics: Challenges and case study. *Future Generation Computer Systems*, 109:500–510, 2020.

[61] Jaiveer Singh and Karan Singh. Congestion control in vehicular ad hoc network: A review. In *Next-generation networks*, pages 489–496. Springer, 2018.

[62] Jian Liu, Jiangtao Li, Lei Zhang, Feifei Dai, Yuanfei Zhang, Xinyu Meng, and Jian Shen. Secure intelligent traffic light control using fog computing. *Future Generation Computer Systems*, 78:817–824, 2018.

[63] G Revathi and VR Sarma Dhulipala. Smart parking systems and sensors: A survey. In *IEEE International Conference on Computing, Communication and Applications*, pages 1–5, 2012.

[64] Kai Lin, Jiming Luo, Long Hu, M Shamim Hossain, and Ahmed Ghoneim. Localization based on social big data analysis in the vehicular networks. *IEEE Transactions on Industrial Informatics*, 13(4):1932–1940, 2016.

[65] Yeliz Yoldaş, Ahmet Önen, SM Muyeen, Athanasios V Vasilakos, and İrfan Alan. Enhancing smart grid with microgrids: Challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 72:205–214, 2017.

[66] Kemas M Tofani, PAA Pramana, BBSDA Harsono, Dhandis R Jintaka, and KG H Mangunnkusumo. SCADA Systems Design to Optimize and Automate Microgrids Systems in Indonesia. In *IEEE International Conference on Technology and Policy in Energy and Electric Power (ICT-PEP)*, pages 187–192, 2020.

[67] Farzad Samie, Lars Bauer, and Jörg Henkel. Edge computing for smart grid: An overview on architectures and solutions. In *IoT for Smart Grids*, pages 21–42. Springer, 2019.

[68] Shehzad Ashraf Chaudhry, Hosam Alhakami, Abdullah Baz, and Fadi Al-Turjman. Securing Demand Response Management: A Certificate based Access Control in Smart Grid Edge Computing Infrastructure. *IEEE Access*, 2020.

[69] WenJing Hou, Yixin Jiang, Wenxin Lei, Aidong Xu, Hong Wen, and Songling Chen. A P2P network based edge computing smart grid model for efficient resources coordination. *Peer-to-Peer Networking and Applications*, pages 1–12, 2020.

[70] Mohamed Amine Ferrag, Messaoud Babaghayou, and Mehmet Akif Yazici. Cyber security for fog-based smart grid SCADA systems: Solutions and challenges. *Journal of Information Security and Applications*, 52:102500, 2020.

[71] Prabal Verma and Sandeep K Sood. Fog assisted-IoT enabled patient health monitoring in smart homes. *IEEE Internet of Things Journal*, 5(3):1789–1796, 2018.

[72] Lakmini P Malasinghe, Naeem Ramzan, and Keshav Dahal. Remote patient monitoring: a comprehensive study. *Journal of Ambient Intelligence and Humanized Computing*, 10(1):57–76, 2019.

[73] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017.

[74] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2017.

[75] ETSI MEC Leadership Team. ETSI MEC. https://www.etsi.org/images/files/technologies/ETSI-MEC-Public-Overview.pdf. [Online; accessed 11/2021].

[76] Dario Sabella, Alessandro Vaillant, Pekka Kuure, Uwe Rauschenbach, and Fabio Giust. Mobile-edge computing architecture: The role of MEC in the Internet of Things. *IEEE Consumer Electronics Magazine*, 5(4):84–91, 2016.

[77] Emiliano Miluzzo, Ramón Cáceres, and Yih-Farn Chen. Vision: mClouds-computing on clouds of mobile devices. pages 9–14. Proceedings of the third ACM workshop on Mobile cloud computing and services, 2012.

[78] Afnan Fahim, Abderrahmen Mtibaa, and Khaled A Harras. Making the case for computational offloading in mobile device clouds. pages 203–205. Proceedings of the 19th annual international conference on Mobile computing & networking (ACM), 2013.

[79] Duc Van Le and Chen-Khong Tham. An optimization-based approach to offloading in ad-hoc mobile clouds. pages 1–6. IEEE Global Communications Conference (Globecom), 2017.

[80] Ragib Hasan, Mahmud Hossain, and Rasib Khan. Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading. *Future Generation Computer Systems*, 86:821–835, 2018.

[81] Duc Van Le and Chen-Khong Tham. A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds. pages 760–765. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2018.

[82] Duc Van Le and Chen-Khong Tham. Quality of service aware computation offloading in an ad-hoc mobile cloud. *IEEE Transactions on Vehicular Technology*, 67(9):8890–8904, 2018.

[83] Md Golam Rabiul Alam, Mohammad Mehedi Hassan, Md ZIa Uddin, Ahmad Almogren, and Giancarlo Fortino. Autonomic computation offloading in mobile edge for IoT applications. *Future Generation Computer Systems*, 90:149–157, 2019.

[84] Ke Zhang, Yongxu Zhu, Supeng Leng, Yejun He, Sabita Maharjan, and Yan Zhang. Deep Learning Empowered Task Offloading for Mobile Edge Computing in Urban Informatics. *IEEE Internet of Things Journal*, 2019.

[85] Zongyao Wang, Xue Feng, Hongbo Zhu, and Changping Liu. Optimal data offloading via an ADMM algorithm in mobile ad hoc cloud with malicious resource providers. *Computer Communications*, 2020.

[86] Xueling Lin, Jingjie Jiang, Calvin Hong Yi Li, Bo Li, and Baochun Li. Circa: collaborative code offloading among multiple mobile devices. *Wireless Networks*, 26(2):823–841, 2020.

[87] Sajeeb Saha, Md Ahsan Habib, Tamal Adhikary, Md Abdur Razzaque, Md Mustafizur Rahman, Meteb Altaf, and Mohammad Mehedi Hassan. Quality-of-Experience-Aware Incentive Mechanism for Workers in Mobile Device Cloud. *IEEE Access*, 9:95162–95179, 2021.

[88] Palash Roy, Sujan Sarker, Md Abdur Razzaque, Md Mamun-or Rashid, Mohmmad Mehedi Hassan, and Giancarlo Fortino. Distributed task allocation in Mobile Device Cloud exploiting federated learning and subjective logic. *Journal of Systems Architecture*, 113:101972, 2021.

[89] Oanh Tran Thi Kim, Nguyen Dang Tri, Nguyen H Tran, Choong Seon Hong, et al. A shared parking model in vehicular network using fog and cloud environment. In *IEEE 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 321–326, 2015.

[90] Ke Zhang, Yuming Mao, Supeng Leng, Sabita Maharjan, and Yan Zhang. Optimal delay constrained offloading for vehicular edge computing networks. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2017.

[91] Chao Zhu, Giancarlo Pastor, Yu Xiao, Yong Li, and Antti Ylae-Jaeaeski. Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, 2018.

[92] Lee Gillam, Konstantinos Katsaros, Mehrdad Dianati, and Alexandres Mouzakitis. Exploring edges for connected and autonomous driving. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 148–153, 2018.

[93] Pedro Henrique Cruz Caminha, Felipe Ferreira da Silva, Roberto Gonçalves Pacheco, Rodrigo de Souza Couto, Pedro Braconnot Velloso, Miguel Elias Mitre Campista, and Luís Henrique MK Maciel Kosmalski Costa. Sensingbus: Using bus lines and fog computing for smart sensing the city. *IEEE Cloud Computing*, 5(5):58–69, 2018.

[94] Arnav Thakur and Reza Malekian. Fog computing for detecting vehicular congestion, an internet of vehicles based approach: A review. *IEEE Intelligent Transportation Systems Magazine*, 11(2):8–16, 2019.

[95] Zhaolong Ning, Jun Huang, and Xiaojie Wang. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications*, 26(1):87–93, 2019.

[96] Abdallah Moubayed, Abdallah Shami, Parisa Heidari, Adel Larabi, and Richard Brunner. Edge-enabled V2X service placement for intelligent transportation systems. *IEEE Transactions on Mobile Computing*, 2020.

[97] Ibrahim Shaer, Anwar Haque, and Abdallah Shami. Multi-Component V2X Applications Placement in Edge Computing Environment. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[98] Jie Tang, Shaoshan Liu, Liangkai Liu, Bo Yu, and Weisong Shi. Lopecs: A low-power edge computing system for real-time autonomous driving services. *IEEE Access*, 8:30467–30479, 2020.

[99] Ahmad Raza Hameed, Saif ul Islam, Ishfaq Ahmad, and Kashif Munir. Energy- and performance-aware load-balancing in vehicular fog computing. *Sustainable Computing: Informatics and Systems*, 30:100454, 2021.

[100] Fuhui Zhou, Yongpeng Wu, Haijian Sun, and Zheng Chu. UAV-enabled mobile edge computing: Offloading optimization and trajectory design. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.

[101] Qiyu Hu, Yunlong Cai, Guanding Yu, Zhijin Qin, Minjian Zhao, and Geoffrey Ye Li. Joint offloading and trajectory design for UAV-enabled mobile edge computing systems. *IEEE Internet of Things Journal*, 6(2):1879–1892, 2018.

[102] Jingyu Xiong, Hongzhi Guo, and Jiajia Liu. Task offloading in UAV-aided edge computing: Bit allocation and trajectory optimization. *IEEE Communications Letters*, 23(3):538–541, 2019.

[103] Cheng Zhan, Han Hu, Xiufeng Sui, Zhi Liu, and Dusit Niyato. Completion time and energy optimization in the uav-enabled mobile-edge computing system. *IEEE Internet of Things Journal*, 7(8):7808–7822, 2020.

[104] Gaoxiang Wu, Yiming Miao, Yu Zhang, and Ahmed Barnawi. Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading. *Computer Communications*, 150:556–562, 2020.

[105] Lei Yang, Haipeng Yao, Jingjing Wang, Chunxiao Jiang, Abderrahim Benslimane, and Yunjie Liu. Multi-UAV-enabled load-balance mobile-edge computing for IoT networks. *IEEE Internet of Things Journal*, 7(8):6898–6908, 2020.

[106] Xiaoheng Deng, Yajun Liu, Congxu Zhu, and Honggang Zhang. Air–Ground Surveillance Sensor Network based on edge computing for target tracking. *Computer Communications*, 166:254–261, 2021.

[107] Quan Chen, Hai Zhu, Lei Yang, Xiaoqian Chen, Sofie Pollin, and Evgenii Vinogradov. Edge Computing Assisted Autonomous Flight for UAV: Synergies between Vision and Communications. *IEEE Communications Magazine*, 59(1):28–33, 2021.

[108] Mobile Edge Computing (MEC), Deployment of Mobile Edge Computing in an NFV environment. *ETSI Group Report MEC 017*, 2018.

[109] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493, 2018.

[110] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[111] Sylvia Todorova Kouyoumdjieva, lafur Ragnar Helgason, and Gunnar Karlsson. CRAWDAD dataset kth/walkers (v. 2014-05-05). Downloaded from https://crawdad.org/kth/walkers/20140505/ostermalm_sparse_run2, May 2014. traceset: ostermalm_sparse_run2.

[112] Daniel Dias and Luis Henrique Maciel Kosmalski Costa. CRAWDAD dataset coppe-ufrj/riobuses (v. 2018-03-19). Downloaded from https://crawdad.org/coppe-ufrj/RioBuses/20180319/RioBuses, March 2018. traceset: RioBuses.

[113] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from https://crawdad.org/roma/taxi/20140717/taxicabs, July 2014. traceset: taxicabs.

[114] Martin Pielot. CRAWDAD dataset telefonica/mobilephoneuse (v. 2019-04-29). Downloaded from https://crawdad.org/telefonica/mobilephoneuse/20190429, April 2019.

[115] Andreas Reinhardt, Paul Baumann, Daniel Burgstahler, Matthias Hollick, Hristo Chonov, Marc Werner, and Ralf Steinmetz. On the accuracy of appliance identification based on distributed load metering data. pages 1–9. IEEE Sustainable Internet and ICT for Sustainability (SustainIT), 2012.

[116] Changho Shin, Eunjung Lee, Jeongyun Han, Jaeryun Yim, Wonjong Rhee, and Hyoseop Lee. The ENERTALK dataset, 15 Hz electricity consumption data from 22 houses in Korea. *Scientific data*, 6(1):1–13, 2019.

[117] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[118] Bai-cun Zhou, Cong-ying Han, and Tian-de Guo. Convergence of stochastic gradient descent in deep neural network. *Acta Mathematicae Applicatae Sinica, English Series*, 37(1):126–136, 2021.

[119] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[120] Michael R Garey and David S Johnson. Approximation algorithms for bin packing problems: A survey. In *Analysis and design of algorithms in combinatorial optimization*, pages 147–172. Springer, 1981.

[121] Panpan Jin, Xincai Fei, Qixia Zhang, Fangming Liu, and Bo Li. Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 267–276, 2020.

[122] Fatema Akhter. A heuristic approach for minimum set cover problem. *Int. J. Adv. Res. Artif. Intell*, 4:40–45, 2015.

Université Paris Cité

LIPADE
Laboratoire d'Informatique PAris DEscartes

Thesis Summary

M^{r}. Mohammed Laroui

Distributed Edge Computing For Enhanced IoT Devices and New Generation Network Efficiency

Thesis Committee :

Pr.	Hassine Moungla	University of Paris Cite	Thesis Director
Dr.	Zohra Slama	UDL SBA	Supervisor
Pr.	Hind Castel	Telecom SudParis	Reviewer
Pr.	Ken Chen	Sorbonne University	Reviewer
Pr.	Hossam Afifi	Telecom SudParis	Examiner
Pr.	Badr Benmammar	University of Tlemcen	Examiner
Dr.	Nassim Dennouni	University of Chlef	Examiner

Year : 2022

Université Paris Cité

En cotutelle avec L'Université Djillali Liabes de Sidi Bel Abbes

École Doctorale Informatique, Télécommunication et Électronique «Edite de Paris *: ED130»*
*Laboratoire d'Informatique Paris Descartes (*LIPADE*)*

Résumé

Distributed Edge Computing For Enhanced IoT Devices and New Generation Network Efficiency

Par Mohammed Laroui

Thèse de doctorat en Informatique (Réseaux)

Dirigée par Hassine Moungla
Et par Zohra Slama

Présentée et soutenue publiquement le 21 Juillet 2022

Devant un jury composé de :

Hind Castel, Professeur, Telecom SudParis, Rapporteur

Ken Chen, Professeur, Sorbonne Université, Rapporteur

Hossam Afifi, Professeur, Telecom SudParis, Examinateur

Badr Benmammar, Professeur, Université de Tlemcen, Examinateur

Nassim Dennouni, Docteur, Université de Chlef, Examinateur

Travaux de Recherche Publié

1. Mohammed Laroui, Hatem Ibn-Khedher, Moussa Ali Cherif, Hassine Moungla, Hossam Afifi, and Ahmed E Kamel. SO-VMEC: Service offloading in virtual mobile edge computing using deep reinforcement learning. *Transactions on Emerging Telecommunications Technologies (ETT), e4211, 2021. [1]*

2. Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa Ali Cherif, Hossam Afifi, and Mohsen Guizani. Edge and Fog Computing for IoT: A Survey on Current Research Activities & Future Directions. *Computer Communications (ComCom), 2021. [2]*

3. Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Autonomous UAV Aided Vehicular Edge Computing for Service Offering. In *IEEE Global Communications Conference (Globecom), 7-11 December 2021, Madrid, Spain. [3]*

4. Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Artificial Intelligence Approach for Service Function Chains Orchestration at The Network Edge. In *IEEE International Conference on Communications (ICC), (pp.1-6), 14-23 June 2021, Montreal, QC, Canada. [4]*

5. Hatem Ibn-Khedher, Mohammed Laroui, Mouna Ben Mabrouk, Hassine Moungla, Hossam Afifi, Alberto Nai Oleari, and Ahmed E Kamal. Edge Computing Assisted Autonomous Driving Using Artificial Intelligence. In *IEEE International Wireless Communications and Mobile Computing (IWCMC), (pp.254-259), 28 June-2 July 2021, Harbin City, China. [5]*

6. Mohammed Laroui, Hatem Ibn Khedher, Hassine Moungla, Hossam Afifi, and Ahmed E Kamal. Virtual Mobile Edge Computing Based on IoT Devices Resources in Smart Cities. In *IEEE International Conference on Communications (ICC), (pp.1-6), 7-11 June 2020, Dublin, Ireland. [6]*

7. Mohammed Laroui, Moussa Ali Cherif, Hatem Ibn Khedher, Hassine Moungla, and Hossam Afifi. Scalable and Cost Efficient Resource Allocation Algorithms Using

Deep Reinforcement Learning. In *IEEE International Wireless Communications and Mobile Computing (IWCMC), (pp.946-951), 15-19 June 2020, Limassol, Cyprus. [7]*

8. Mohammed Laroui, Boubakr Nour, Hassine Moungla, Hossam Afifi, and Moussa Ali Cherif. Mobile Vehicular Edge Computing Architecture using Rideshare Taxis as a Mobile Edge Server. In *IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), (pp.1-2), 10-13 January 2020, Las Vegas, NV, USA. [8]*

9. Mohammed Laroui, Aicha Dridi, Hossam Afifi, Hassine Moungla, Michel Marot, and Moussa Ali Cherif. Energy Management For Electric Vehicles in Smart Cities: A Deep Learning Approach. In *IEEE International Wireless Communications and Mobile Computing (IWCMC), (pp.2080-2085), 24-28 June 2019, Tangier, Morocco. [9]*

10. Mohammed Laroui, Akrem Sellami, Boubakr Nour, Hassine Moungla, Hossam Afifi, Sofiane Boukli Hacene. Driving Path Stability in VANETs. In *IEEE Global Communications Conference (GLOBECOM), (pp.1-6), 9-13 December 2018, Abu Dhabi, United Arab Emirates. [10]*

1. Introduction générale

1.1 Motivation

L'Internet des objets (IoT) a été un facteur déterminant dans le développement de l'industrie de la haute technologie. Avec le grand nombre d'appareils et de profits à réaliser au cours des prochaines années, L'IoT aura un impact profond et dominant sur l'industrie de la haute technologie en général et sur l'industrie des semi-conducteurs en particulier. Les appareils IoT peuvent en fait être n'importe quel type de capteurs et de puces avec différentes capacités fabriqués par différents fabricants, et il existe de nombreuses applications qui peuvent être conçues pour permettre les villes intelligentes, les transports intelligents, les maisons intelligentes et les soins de santé intelligents.

Dans l'IoT, les appareils connectés peuvent générer une énorme quantité de données à très haut débit et certaines applications peuvent nécessiter une très faible latence. Les données sont directement envoyées dans le cloud pour être stockées et traitées dans les centres de données. L'infrastructure cloud traditionnelle sera confrontée à une série de défis en raison de la centralisation du stockage, calcule, et de la longue distance entre les appareils connectés et les centres de données.

Pour relever ce défi, edge computing semble être une technologie prometteuse qui fournit des ressources de calcul plus proches des appareils IoT.

Dans cette thèse, nous sommes motivés pour identifier et résoudre les problèmes d'architecture edge computing et l'intégration de cette architecture avec les applications IoT. De plus, la littérature manque d'architectures générales edge-IoT, ce qui nous amène à concevoir des architectures récentes, en particulier pour le service offloading et vehicular edge computing.

De plus, malgré l'importance des tâches d'optimisation, de telles fonctions efficaces manquent dans les architectures globales edge-IoT.

1.2 Contribution

Notre contribution présente de nouveaux modèles d'optimisation différents (algorithmes exacts et à grande échelle) au niveau de la couche edge qui résolvent à la fois les problèmes de service offloading liés au placement dans le edge de réseau et les problèmes de vehicular edge computing qui utilisent les véhicules comme des serveurs edge mobile pour fournir des services aux utilisateurs tels que le calcule et le online streaming.

Tout d'abord, pour le service offloading, nous proposons des modèles de placement optimaux dans le edge du réseau pour les tâches des utilisateurs. Ensuite, pour vehicular edge computing, nous proposons des modèles exact pour la couverture des vehicules utilisant des serveurs edge mobile (Taxis et UAV), et la conduite autonome basé sur le edge computing.

De plus, pour faire face aux problèmes de haute complexité des algorithmes optimaux dans les réseaux denses, nous proposons des algorithmes à grande échelle pour les cas d'utilisation du service offloading et vehicular edge computing.

Pour évaluer l'efficacité des algorithmes proposés pour les réseaux à petite et grande échelle, nous utilisons différents outils, notamment, des plates-formes pour les modèles d'intelligence artificielle et des simulateurs pour les réseaux edge.

2. Edge Computing pour Internet des Objets: Definitions et État de l'art

2.1 Definitions

2.1.1 Internet des Objets (IoT)

Est une nouvelle technologie envisagée comme un réseau d'appareils et de machines qui communiquent entre eux et avec Internet. L'IoT est connu comme l'un des catalyseurs importants des technologies futures. Il suscite également un grand intérêt auprès des entreprises. Dans un sens plus large, l'IoT vise à créer des systèmes basés sur l'interconnexion d'objets intelligents. Ces objets échangent des informations entre eux en utilisant différents protocoles, tels que Wi-Fi, Bluetooth, ZigBee, etc. La principale caractéristique de l'IoT est l'intégration de différentes technologies de communication (par exemple, capteurs filaires et sans fil, etc.) pour améliorer la coopération et l'interaction entre les différentes technologies.

La figure 1 montre les domaines d'application de l'IoT.

L'augmentation rapide des appareils connectés/intelligents dans le monde et l'évolution des besoins des utilisateurs et des applications avec une grande quantité de traitement de données ont conduit à un ensemble de nouvelles technologies permettant un traitement rapide des données et des services fiables. Le cloud computing fait partie de ces technologies.

2.1.2 Cloud Computing

Le cloud computing [11] vise à fournir divers services aux utilisateurs dans le cloud. Différents types de cloud peuvent être déployés, notamment privé, public, hybride et commu-

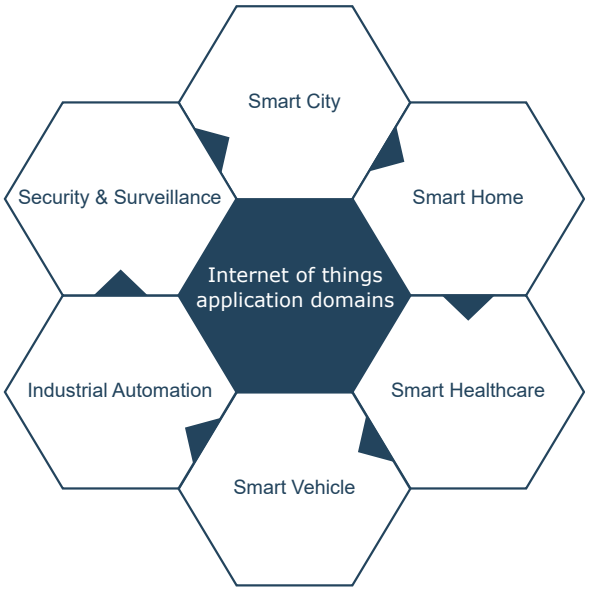


Fig. 1 Les Domaines d'application de l'IoT.

nautaire. Le cloud public [12] fournit des services à un grand nombre d'utilisateurs sur Internet. Le cloud privé [13] offre des services spécifiques aux organisations privées. Le cloud communautaire [14] vise à fournir des services à un groupe d'organisations. Enfin, le cloud hybride [15] permet aux organisations d'équilibrer les coûts et les problèmes de contrôle.

La figure 2 affiche l'architecture du cloud computing.

Dans le cloud computing, les utilisateurs se connectent directement au cloud via Internet et commencent à échanger des données sur le réseau, ce qui entraîne une quantité massive de données en peu de temps. Cependant, l'architecture centralisée du cloud n'est pas efficace pour traiter la quantité massive de données générées par les appareils IoT qui nécessitent un temps de réponse court. Pour surmonter un tel problème, une technologie alternatif, à savoir l'edge computing, a été utilisé qui traite les données dans les appareils connectés ou les passerelles locales.

2.1.3 Edge/Fog Computing

L'edge computing vise à être la future solution IoT qui résout différents problèmes, y compris les applications limitées dans le temps et basées sur le calcul. Les avantages du traitement des données dans le edge du réseau sont les suivants : réduire la charge du réseau et la latence

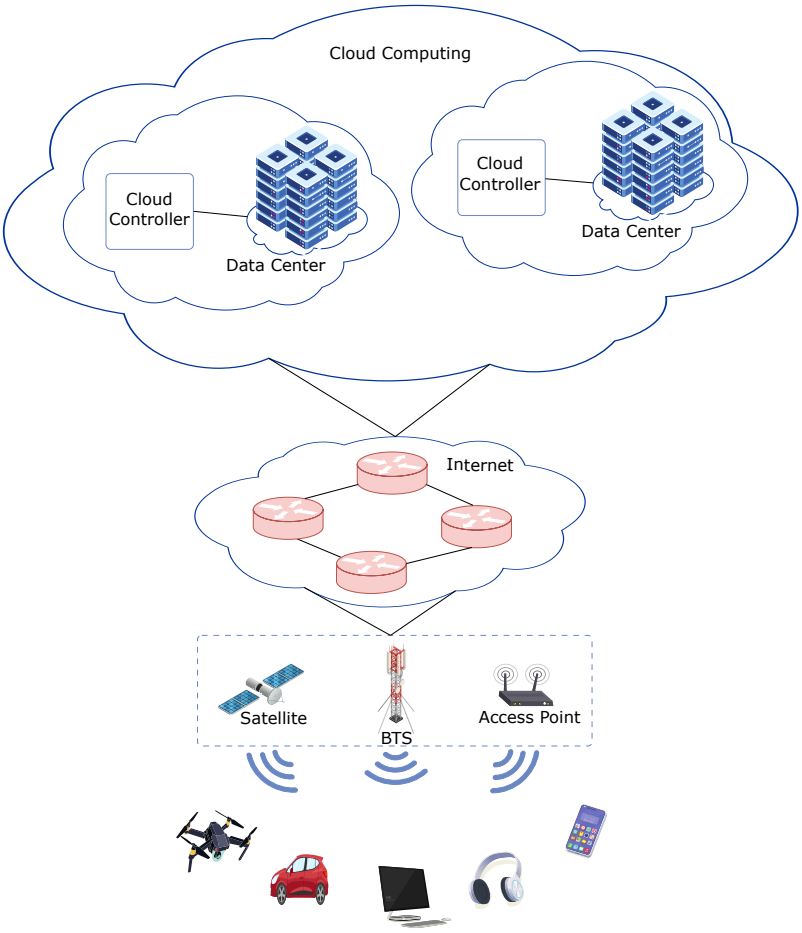


Fig. 2 Architecture du Cloud Computing.

des communications, donner aux petits et moyens inventeurs toutes les chances d'aider à soutenir les innovations futures, réduire la consommation d'énergie des nœuds mobiles et éliminer la congestion au sein du réseau central, tout en offrant plus de fiabilité, de sécurité et de protection de la vie privée. Dans la section suivante, nous présentons l'état de l'art pour les deux axes de recherche étudiés, service offloading et vehicular edge computing.

La figure 3 montre l'architecture edge computing.

2.2 État de l'art

L'intégration de l'edge computing avec l'Internet des objets (Edge-IoT) fait face à de nombreux problèmes et défis liés à la QoS. Pour cela, différentes axes de recherche liées à Edge-IoT ont été étudiées pour couvrir tous les modèles de communication afin de satisfaire

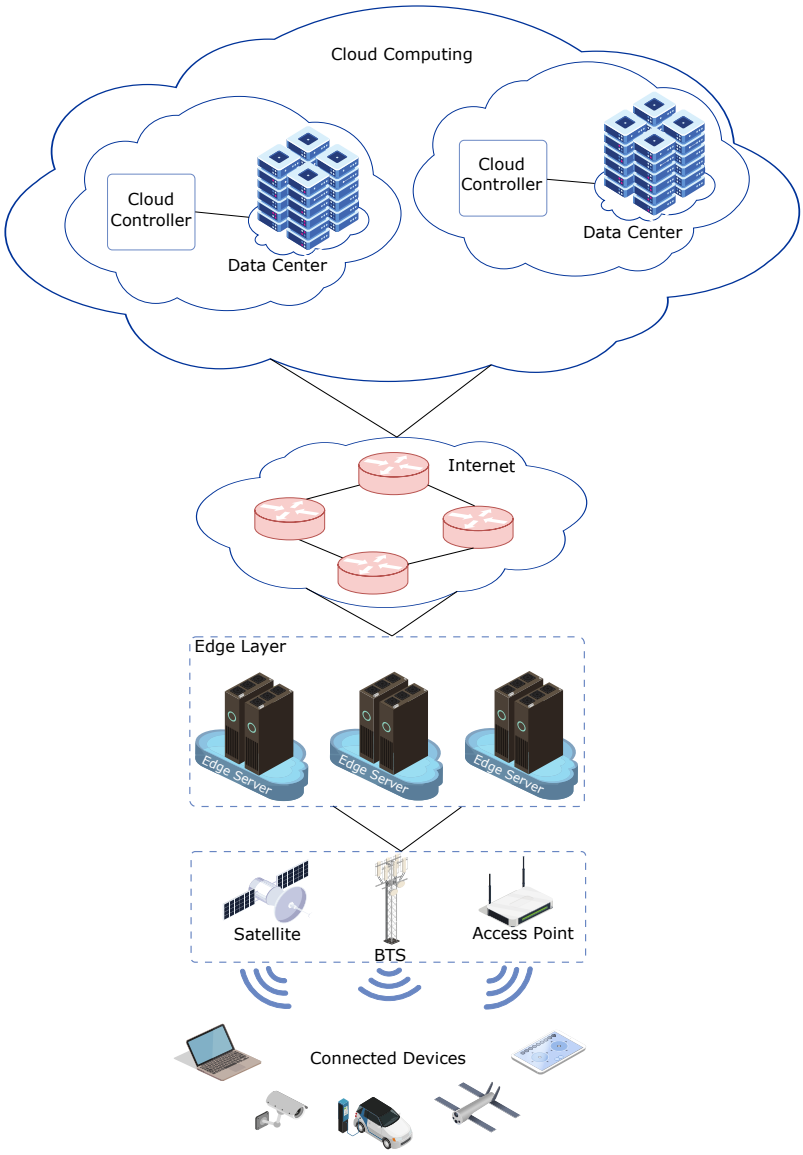


Fig. 3 Architecture du Edge Computing.

les exigences des applications et fournir des services efficaces aux utilisateurs. Cette partie présente l'état de l'art pour deux axes de recherche pour Edge-IoT; service offloading et vehicular edge computing.

2.2.1 Service Offloading

Service offloading fait référence au déplacé certains services à forte intensité de calcul vers les serveurs edge pour traitement. L'allocation de ressources pour cette procédure fait

référence à l'allocation de certaines ressources de calcul sur des serveurs edge en fonction des exigences du service. Le déplacement des services vers le edge du réseau facilite le stockage, la prestation de services, la mise en cache du contenu et la gestion de l'IoT, ce qui permet d'améliorer les temps de réponse et les taux de transfert, garantissant ainsi aux utilisateurs le service le plus rapide et le meilleur possible.

Le tableau 1 affiche le résumé des travaux de recherche existants pour le service offloading dans l'environment edge computing.

Article de recherche	***Idée principale***	***Formulation du problème***	***Position du edge***	***Avantages***	***Limites***
Miluzzo *et al.* [16]	Exécution des tâches dans des appareils mobiles intelligents.	Quelques équations pour le paiement des services, etc.	Mobile	Exécuter des tâches pour d'autres utilisateurs.	Temps de réponse long.
Fahim *et al.* [17]	Task offloading à l'aide d'appareils mobiles comme nœuds de calcul.	Aucune formulation mathématique n'est utilisée.	Mobile	Consommation d'énergie et temps de traitement efficaces.	L'impact de la mobilité n'est pas étudié.

Van *et al.* [18]	Traitement des tâches dans les cloudlets mobiles.	Processus décisionnel de Markov (MDP).	Mobile	Améliorer le temps et la consommation d'énergie.	Les détails de l'implémentation de la solution sont manquants.
Hasan *et al.* [19]	Aura, task offloading dans une architecture cloud ad-hoc mobile.	Équations pour la mesure des coûts, du temps et des ressources.	Mobile	Améliorer le coût et le temps d'exécution.	L'impact de la mobilité n'est pas étudié.
Van *et al.* [20]	Tasks offloading dans un environnement cloud mobile ad hoc.	- Processus décisionnel de Markov (MDP). - Deep-Q-Network (DQN).	Mobile	Améliorer la consommation d'énergie, le paiement et les délais.	L'évaluation dans les réseaux à grande échelle n'est pas étudiée.
Van *et al.* [21]	Technique de 'task offloading' dans le cloud mobile ad-hoc.	Processus décisionnel de Markov (MDP).	Mobile	Améliorer le taux de perte de tâches, le délai moyen et la consommation d'énergie.	La solution n'est pas étudiée dans un environnement réel.

Alam *et al.* [22]	Offloading des calculs dans un environnement mobile edge computing.	- Processus décisionnel de Markov (MDP). - Deep Q-learning.	Mobile	Minimiser la latence.	L'impact de la mobilité n'est pas étudié.
Zhang *et al.* [23]	Task offloading dans un environnement mobile.	Deep Q-learning.	Fixed	Task offloading fiable et optimal.	La complexité n'est pas étudiée.
Wang *et al.* [24]	Task offloading optimal dans un environnement cloud Adhoc mobile.	Algorithme d'optimisation distribué.	Mobile	Identifier les fournisseurs de ressources malveillants.	Le détail de l'implémentation de la solution est manquant.
Lin *et al.* [25]	Circa, offloading framework basé sur les appareils mobiles.	Algorithmes d'allocation des tâches.	Mobile	Améliorer le temps d'exécution.	Risque élevé d'échec de la communication.

Saha *et al.* [26]	Cloud framework pour les appareils mobiles.	Modèle optimal et algorithme 'greedy' pour le provisionnement des tâches.	Mixed	Faible coût.	La solution n'est pas étudiée dans un environnement réel.
Roy *et al.* [27]	Task allocation framework pour les appareils mobile.	Algorithmes pour task allocation.	Mobile	Efficace à la fois pour l'utilité de l'acheteur et la QoE.	La solution n'est pas étudiée dans un environnement à grande échelle.

Table 1 Résumé des recherches existant sur le service offloading dans le edge computing.

Les algorithmes de service offloading dans la littérature sont basés sur la formulation de modèles exacts pour mesurer le comportement optimal d'une telle solution. Cependant, dans la pratique, les opérateurs de réseau conçoivent toujours des heuristiques pour traiter les problèmes d'évolutivité dans les réseaux à grande échelle avec d'énormes quantités de données. Actuellement, ces heuristiques sont des solutions quasi-optimales et sont parfois très loin d'être optimales. De plus, les heuristiques ne sont pas utilisées lorsque l'entrée du système change et donne des erreurs fatales lorsqu'une entrée inattendue arrive dans le système.

2.2.2 Vehicular Edge Computing

Les véhicules intelligents sont considérés comme des appareils mobiles équipés de capteurs, ayant la capacité de collecter, de calculer et de communiquer des données. Les informations

sont collectées à la fois à partir de capteurs embarqués dans le véhicule et de l'environnement extérieur. Edge computing peut fournir une architecture efficace et évolutive pour les réseaux de véhicules en améliorant le traitement des données et le trafic en temps réel. Nous présentons dans cette partie les solutions existantes pour les réseaux véhiculaires basées sur le edge computing (vehicular edge computing).

Le tableau 2 présente le résumé des travaux existants de vehicular edge computing.

Article de recherche	***Idée principale***	***Formulation du problème***	***Position du edge***	***Avantages***	***Limites***
Kim *et al.* [28]	Gestion du stationnement dans un environnement fog-cloud.	'Many-to-one matching game model'.	Fixed	Fiabilité pour trouver une place de parking disponible.	L'évolutivité du système fog n'est pas étudiée.
Zhang *et al.* [29]	Task offloading dans un environnement mobile.	'Stackelberg game'.	Fixed	Délai optimal pour le calcul des tâches.	Le risque de coupure de liaison n'est pas étudié.
Zhu *et al.* [30]	Allocation des taches dans un environnement mobile.	Un modèle exact pour le problème d'allocation des tâches.	Mixed	Améliorer la latence.	Nécessite une ressource de calcul élevée.
Gillam *et al.* [31]	Architecture de calcul distribuée pour les véhicules.	N/A	Fixed	Réduire la latence.	Aucune évaluation n'est présentée.

Caminha *et al.* [32]	SensingBus, un système de collecte de données à partir de capteurs sur les bus urbains.	N/A	Mobile	Améliore l'utilisation du processeur et de la mémoire.	Les bus ne peuvent pas assurer une large couverture.
Thakur *et al.* [33]	Techniques d'identification de la congestion basées sur les véhicules connectés.	N/A	Fixed	Le système peut détecter la congestion pour atteindre les objectifs de gestion du trafic.	La solution nécessite une évaluation dans des réseaux à grande échelle.
Ning *et al.* [34]	Modèle de gestion du trafic dans un environnement vehiculaire-edge.	Modèle exact pour le 'offloading'.	Fixed	Réduire le temps de réponse.	L'impact de la mobilité des véhicules n'est pas étudié.
Moubayed *et al.* [35]	Placement de service V2X dans un environnement edge-mobile.	Modèle de placement de service V2X optimal.	Fixed	Améliorer le délai et l'utilisation des ressources.	L'impact de la mobilité des véhicules n'est pas étudié.

Shaer *et al.* [36]	Placement de service V2X dans un environnement edge computing.	Modèle optimal pour le placement des services.	Fixed	Minimiser le délai.	La métrique d'utilisation des ressources n'est pas étudiée.
Tang *et al.* [37]	Système de conduite autonome en temps réel dans l'environnement edge computing.	Modèle de 'Task offloading'.	Fixed	Utilisation efficace des ressources et consommation d'énergie.	L'utilisation des ressources et les métriques de QoS ne sont pas étudiées.
Hameed *et al.* [38]	Approche de 'clustering' dynamique dans un environnement 'vehicular edge computing'.	Modèle optimal pour l'utilisation du réseau.	Mobile	Améliorer la qualité de service et la consommation d'énergie.	Un seul modèle de mobilité est utilisé qui représente juste un cas de l'environnement réel.

Table 2 Résumé des recherches existant sur le vehicular edge computing.

Les convergences actuelles entre la technologie edge computing et les réseaux vehiculaire devraient prendre en compte les métriques de sécurité et les réactions en temps réel dans les situations critiques. Par conséquent, malgré les travaux existants dans les communications véhiculaires assistées par le Edge computing, les applications de nouvelle génération telles

que la sécurité routière et la conduite autonome nécessitent un déploiement efficace pour atteindre une latence ultra-faible dans les réseaux de véhicules denses.

3 Nos Solutions Proposées Pour Les Réseaux Edge Computing

La procédure d'optimisation du réseau reste une tâche importante qui doit être soigneusement étudiée en raison des évolutions rapides du réseau et des demandes des utilisateurs. Dans cette partie, nous présentons nos différents solutions pour le service offloading et vehicular edge computing.

3.1 Service Offloading dans Edge Computing

Avec l'évolution rapide des générations de réseaux et les exigences des applications, le service offloading est considéré comme l'une des tâches réseau les plus importantes qui permettent de 'offload' les services gourmands en ressources vers un centre de calcul optimal dans les serveurs edge et le cloud central pour le traitement. Cette tâche complexe nécessite une communication efficace, des stratégies d'optimisation et des technologies de virtualisation pour gérer différents services et fournir une qualité de service efficace aux utilisateurs.

Nous avons proposé des modèles optimaux [7, 6, 1, 4] d'orchestration des tâches, des VNF et des SFC dans le edge de réseau (Optimal Virtual Edge nodes Placement Algorithm (VEnPA), Service Offloading in Virtual Mobile Edge Computing (SO-VMEC), Optimal SFC Placement in IoT-VMEC (OSPV)) en tenant compte des contraintes de ressources de calcul.

Pour gérer les réseaux denses, nous avons proposé des algorithmes à grande échelle [7, 6, 1, 4] qui prennent en charge une énorme quantité d'appareils, de données et de demandes d'utilisateurs. Les algorithmes d'intelligence artificielle d'apprentissage par renforcement et d'apprentissage par renforcement profond sont utilisés pour le placement des tâches des utilisateurs et les tranches VNF (Q-learning et Deep Q-learning (DQN)), tandis

qu'un algorithme Efficient SFC Placement (ESPV) basé sur le modèle Bin Packing [39] est implémenté pour le placement des SFC dans les serveurs edge virtuel.

Dans ce qui suit, nous présentons nos solutions pour le vehicular edge computing.

3.2 Vehicular Edge Computing

Avec le développement récent des réseaux urbains, les véhicules intelligents sont considérés comme des appareils mobiles équipés de différents capteurs embarqués. De nos jours, les véhicules commencent à avoir la capacité de communiquer entre eux (véhicule à véhicule) et avec l'infrastructure urbaine (véhicule à infrastructure), de collecter des données et d'effectuer des calculs et des traitements dans le véhicule. Aujourd'hui, généralement, les applications de calcul intensif sont exécutées sur le cloud. Cependant, les applications sensibles aux retards peuvent souffrir de la réponse longue durée et peuvent souffrir d'une mauvaise qualité d'expérience.

Les nouvelles générations de réseaux véhiculaires permettent d'utiliser les véhicules comme serveur edge dans le contexte des systèmes vehicular edge computing (VEC) pour le traitement des données, etc. Il permet de fournir des informations à d'autres appareils connectés, y compris les appareils des utilisateurs, d'autres véhicules, etc.

Nous avons proposé des modèles optimaux [8, 3, 5] pour vehicular edge computing, y compris une couverture maximale (MVEC) pour les applications de streaming vidéo en ligne en utilisant des taxis comme des serveurs edge mobiles à l'intérieur de la ville, le offloading optimal des VNF de la conduite autonome (OVEAP) dans les serveurs edge, et le modèle UAV-edge (OFMU) pour la diffusion de video en ligne en utilisant des UAV autonomes comme serveurs edge mobile.

De plus, pour couvrir les fortes exigences des réseaux denses. nous avons proposé des algorithmes à grande échelle [8, 3, 5]. Pour le premier cas d'utilisation, afin de maximiser la couverture des véhicules clients à l'aide du serveur edge mobile (Taxis), nous proposons un algorithme heuristique qui sélectionne uniquement les taxis qui partagent le(s) même(s) segment(s) de chemin avec le véhicule client afin de réduire la complexité de l'algorithme. Pour le deuxième cas d'utilisation, nous proposons un algorithme heuristique pour FMU (AFMU) basé sur le modèle Weighted Set Cover (WSC) pour augmenter la disponibilité du

drone pendant le service. Enfin, pour le dernier cas d'utilisation, nous proposons Efficient Edge Autopilot Placement (DVEAP) basé sur l'apprentissage par renforcement profond pour placer efficacement les VNF de conduite autonome dans le serveur edge.

4 Conclusion et perspectives

Cette thèse présente edge computing pour l'Internet des objets. La première partie présente l'introduction générale. La deuxieme partie presente les technologies étudiées (l'Internet des objets, le cloud computing, l'edge computing) et les travaux proposés par la communauté de recherche pour deux grands domaines de edge computing, service offloading et vehicular edge computing.

La troisième partie présente nos solutions proposé (optimal et grand echelle) pour service offloading et vehicular edge computing.

les modèles optimaux proposés pour optimiser l'utilisation de edge pour différentes métriques telles que le temps et l'utilisation des ressources. Nous avons proposé des modèles optimaux d'orchestration des tâches, des VNF et des SFC dans le edge (VEnPA, SO-VMEC, OSPV). De plus, nous avons proposé d'autres modèles optimaux pour vehicular edge computing, y compris la couverture maximale (MVEC), la conduite autonome (OVEAP), et le modèle UAV-edge (OFMU) utilisant des UAV autonomes comme serveurs edge mobile. Les résultats numériques montrent que l'algorithme proposé donne de bons résultats en termes d'utilisation des ressources dans le edge pour les réseaux à petite échelle.

De plus, pour gérer les réseaux denses, nous avons proposé des algorithmes à grande échelle qui prennent en charge une énorme quantité d'appareils, de données et de demandes d'utilisateurs pour service offloading et vehicular edge computing. Des algorithmes heuristiques sont proposés pour l'orchestration SFC (ESPV), maximiser la couverture des véhicules (AFMU) par les serveurs edge mobile (Taxis et UAV). De plus, les algorithmes d'intelligence artificielle (Q-learning et Deep Q-learning (DQN)) sont utilisés pour le placement des VNF de conduite autonome (DVEAP) et la navigation autonome des drones. Les résultats obtenus prouvent l'efficacité et la faisabilité des solutions proposées par rapport aux algorithmes optimaux.

Malgré les solutions existantes décrites ci-dessus, plusieurs préoccupations et défis doivent être étudiés afin d'intégrer pleinement edge computing aux applications IoT. Les défis suivants examinent les principaux enjeux, ainsi que les nouveaux concepts que la communauté de la recherche devrait prendre au sérieux dans les travaux futurs.

• La disponibilité des services doit être assurée pour la prochaine génération d'applications de edge computing grâce à l'utilisation de divers mécanismes tels que la surveillance, la prédiction et les sauvegardes du système.

• La mobilité est un facteur important dans le système Edge-IoT, car la plupart des appareils connectés, tels que les véhicules, et les drones, sont mobiles, ce qui entraîne de fréquentes dégradations de service et des pannes de liaison entre les serveurs et les appareils, ce qui diminue la qualité de service de système Edge-IoT. Pour cela, des modèles efficaces et développés pour la détection des obstacles et la prévision du trafic, doivent être proposés pour surmonter le problème de mobilité.

• L'architecture edge computing est composée de plusieurs systèmes distribués, la consommation d'énergie devrait être considérable, ce qui augmentera les dépenses. En conséquence, de nombreux efforts doivent être faits pour résoudre ce problème en développant des modèles énergétiques efficaces dans les systèmes edge, tels que l'optimisation des ressources, le recours à une énergie plus respectueuse de l'environnement, etc.

References

[1] Mohammed Laroui, Hatem Ibn-Khedher, Moussa Ali Cherif, Hassine Moungla, Hossam Afifi, and Ahmed E Kamel. SO-VMEC: Service offloading in virtual mobile edge computing using deep reinforcement learning. *Transactions on Emerging Telecommunications Technologies*, page e4211, 2021.

[2] Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa A Cherif, Hossam Afifi, and Mohsen Guizani. Edge and fog computing for IoT: A survey on current research activities & future directions. *Computer Communications*, 180:210–231, 2021.

[3] Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Autonomous UAV Aided Vehicular Edge Computing for Service Offering. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2021.

[4] Mohammed Laroui, Hatem Ibn-Khedher, Hassine Moungla, and Hossam Afifi. Artificial Intelligence Approach for Service Function Chains Orchestration at The Network Edge. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2021.

[5] Hatem Ibn-Khedher, Mohammed Laroui, Mouna Ben Mabrouk, Hassine Moungla, Hossam Afifi, Alberto Nai Oleari, and Ahmed E Kamal. Edge Computing Assisted Autonomous Driving Using Artificial Intelligence. In *IEEE International Wireless Communications and Mobile Computing (IWCMC)*, pages 254–259, 2021.

[6] Mohammed Laroui, Hatem Ibn Khedher, Hassine Moungla, Hossam Afifi, and Ahmed E Kamal. Virtual Mobile Edge Computing based on IoT devices resources in smart cities. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[7] Mohammed Laroui, Moussa Ali Cherif, Hatem Ibn Khedher, Hassine Moungla, and Hossam Afifi. Scalable and Cost Efficient Resource Allocation Algorithms Using Deep Reinforcement Learning. In *IEEE International Wireless Communications and Mobile Computing (IWCMC)*, pages 946–951, 2020.

References

[8] Mohammed Laroui, Boubakr Nour, Hassine Moungla, Hossam Afifi, and Moussa Ali Cherif. Mobile vehicular edge computing architecture using rideshare taxis as a mobile edge server. In *IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–2, 2020.

[9] Mohammed Laroui, Aicha Dridi, Hossam Afifi, Hassine Moungla, Michel Marot, and Moussa Ali Cherif. Energy management for electric vehicles in smart cities: a deep learning approach. In *IEEE International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 2080–2085, 2019.

[10] Mohammed Laroui, Akrem Sellami, Boubakr Nour, Hassine Moungla, Hossam Afifi, and Sofiane B Hacene. Driving path stability in VANETs. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[11] Prince Kwame Senyo, Erasmus Addae, and Richard Boateng. Cloud computing research: A review of research themes, frameworks, methods and future research directions. *International Journal of Information Management (IJIM)*, 38(1):128–139, 2018.

[12] Peter Mell, Tim Grance, et al. The NIST definition of cloud computing. 2011.

[13] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future generation computer systems (FGCS)*, 28(3):583–592, 2012.

[14] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing—The business perspective. *Decision support systems (DSS)*, 51(1):176–189, 2011.

[15] Gabriel Mateescu, Wolfgang Gentzsch, and Calvin J Ribbens. Hybrid computing—where HPC meets grid and cloud computing. *Future generation computer systems (FGCS)*, 27(5):440–453, 2011.

[16] Emiliano Miluzzo, Ramón Cáceres, and Yih-Farn Chen. Vision: mClouds-computing on clouds of mobile devices. pages 9–14. Proceedings of the third ACM workshop on Mobile cloud computing and services, 2012.

[17] Afnan Fahim, Abderrahmen Mtibaa, and Khaled A Harras. Making the case for computational offloading in mobile device clouds. pages 203–205. Proceedings of the 19th annual international conference on Mobile computing & networking (ACM), 2013.

[18] Duc Van Le and Chen-Khong Tham. An optimization-based approach to offloading in ad-hoc mobile clouds. pages 1–6. IEEE Global Communications Conference (Globecom), 2017.

[19] Ragib Hasan, Mahmud Hossain, and Rasib Khan. Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading. *Future Generation Computer Systems*, 86:821–835, 2018.

[20] Duc Van Le and Chen-Khong Tham. A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds. pages 760–765. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2018.

[21] Duc Van Le and Chen-Khong Tham. Quality of service aware computation offloading in an ad-hoc mobile cloud. *IEEE Transactions on Vehicular Technology*, 67(9):8890–8904, 2018.

[22] Md Golam Rabiul Alam, Mohammad Mehedi Hassan, Md ZIa Uddin, Ahmad Almogren, and Giancarlo Fortino. Autonomic computation offloading in mobile edge for IoT applications. *Future Generation Computer Systems*, 90:149–157, 2019.

[23] Ke Zhang, Yongxu Zhu, Supeng Leng, Yejun He, Sabita Maharjan, and Yan Zhang. Deep Learning Empowered Task Offloading for Mobile Edge Computing in Urban Informatics. *IEEE Internet of Things Journal*, 2019.

[24] Zongyao Wang, Xue Feng, Hongbo Zhu, and Changping Liu. Optimal data offloading via an ADMM algorithm in mobile ad hoc cloud with malicious resource providers. *Computer Communications*, 2020.

[25] Xueling Lin, Jingjie Jiang, Calvin Hong Yi Li, Bo Li, and Baochun Li. Circa: collaborative code offloading among multiple mobile devices. *Wireless Networks*, 26(2):823–841, 2020.

[26] Sajeeb Saha, Md Ahsan Habib, Tamal Adhikary, Md Abdur Razzaque, Md Mustafizur Rahman, Meteb Altaf, and Mohammad Mehedi Hassan. Quality-of-Experience-Aware Incentive Mechanism for Workers in Mobile Device Cloud. *IEEE Access*, 9:95162–95179, 2021.

[27] Palash Roy, Sujan Sarker, Md Abdur Razzaque, Md Mamun-or Rashid, Mohmmad Mehedi Hassan, and Giancarlo Fortino. Distributed task allocation in Mobile Device Cloud exploiting federated learning and subjective logic. *Journal of Systems Architecture*, 113:101972, 2021.

References

[28] Oanh Tran Thi Kim, Nguyen Dang Tri, Nguyen H Tran, Choong Seon Hong, et al. A shared parking model in vehicular network using fog and cloud environment. In *IEEE 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 321–326, 2015.

[29] Ke Zhang, Yuming Mao, Supeng Leng, Sabita Maharjan, and Yan Zhang. Optimal delay constrained offloading for vehicular edge computing networks. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2017.

[30] Chao Zhu, Giancarlo Pastor, Yu Xiao, Yong Li, and Antti Ylae-Jaeaeski. Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, 2018.

[31] Lee Gillam, Konstantinos Katsaros, Mehrdad Dianati, and Alexandres Mouzakitis. Exploring edges for connected and autonomous driving. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 148–153, 2018.

[32] Pedro Henrique Cruz Caminha, Felipe Ferreira da Silva, Roberto Gonçalves Pacheco, Rodrigo de Souza Couto, Pedro Braconnot Velloso, Miguel Elias Mitre Campista, and Luís Henrique MK Maciel Kosmalski Costa. Sensingbus: Using bus lines and fog computing for smart sensing the city. *IEEE Cloud Computing*, 5(5):58–69, 2018.

[33] Arnav Thakur and Reza Malekian. Fog computing for detecting vehicular congestion, an internet of vehicles based approach: A review. *IEEE Intelligent Transportation Systems Magazine*, 11(2):8–16, 2019.

[34] Zhaolong Ning, Jun Huang, and Xiaojie Wang. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications*, 26(1):87–93, 2019.

[35] Abdallah Moubayed, Abdallah Shami, Parisa Heidari, Adel Larabi, and Richard Brunner. Edge-enabled V2X service placement for intelligent transportation systems. *IEEE Transactions on Mobile Computing*, 2020.

[36] Ibrahim Shaer, Anwar Haque, and Abdallah Shami. Multi-Component V2X Applications Placement in Edge Computing Environment. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[37] Jie Tang, Shaoshan Liu, Liangkai Liu, Bo Yu, and Weisong Shi. Lopecs: A low-power edge computing system for real-time autonomous driving services. *IEEE Access*, 8:30467–30479, 2020.

[38] Ahmad Raza Hameed, Saif ul Islam, Ishfaq Ahmad, and Kashif Munir. Energy-and performance-aware load-balancing in vehicular fog computing. *Sustainable Computing: Informatics and Systems*, 30:100454, 2021.

[39] Michael R Garey and David S Johnson. Approximation algorithms for bin packing problems: A survey. In *Analysis and design of algorithms in combinatorial optimization*, pages 147–172. Springer, 1981.