On the Edge of the Deployment: A Survey on Multi-Access Edge Computing

PEDRO CRUZ*, Inria, France
NADJIB ACHIR, Inria, France and Université Sorbonne Paris Nord, France
ALINE CARNEIRO VIANA, Inria, France

Multi-Access Edge Computing (MEC) attracts much attention from the scientific community due to its scientific, technical, and commercial implications. In particular, the ETSI standard convergence consolidates the discussions around MEC. Still, the existing MEC practical initiatives are incomplete in their majority, hardening or invalidating their effective deployment. To fill this gap, it is essential to understand a series of experimental prototypes, implementations, and deployments. The early implementations can reveal the potential, the limitations, the related technologies, and the development tools for MEC adoption. In this context, this work first brings a discussion on existing MEC initiatives regarding the use cases they target and their vision (i.e., whether they are more network-related or more distributed systems). Second, we survey MEC practical initiatives according to their strategies, including the ETSI MEC standard. Besides, we compare the strategies according to related limitations, impact, and deployment efforts. We also survey the existing tools making MEC systems a reality. Finally, we give hints to issues yet to be addressed in practice. By bringing a better comprehension of MEC initiatives, we believe this survey will help researchers and developers design their own MEC systems or improve and simplify the usability of existing ones.

CCS Concepts: • **Networks** → **Cloud computing**; **Network experimentation**.

Additional Key Words and Phrases: Multi-Access Edge Computing, experimentation, edge computing, mobile edge computing

1 INTRODUCTION

The edge of the current Internet consists of dense deployment of wireless devices ranging from smartphones to smart vehicles and sensors/actuators to intelligent appliances. Consequently, individuals are immersed in a highly connected and ubiquitous cyber-physical context. The satisfaction of network end-users and the provision of numerous services have thus become the main focus. All such factors challenge application developers and service providers.

One of these challenges lies in the fact that, even though the connected devices are typically resource-constrained, their users run resource-hungry applications [6]. This situation means that the applications need computing and storage support from some other source, for instance, the cloud. However, the cloud and its resources are usually far away from the devices. Consequently, cloud resource consumption implies higher latency in the application-cloud-application communication interaction while increasing upstream Internet traffic. For delay-sensitive applications, high latency degrades the Quality of Experience (QoE) [109].

A solution to provide users with the expected QoE is to afford cloud-like resources close to the edge of the network [116]. To the users, the edge resources are topologically closer than the cloud. For this reason, the

*Authors contributed equally to this research.

Authors' addresses: Pedro Cruz, Inria, Palaiseau, France, pedro.cruz-caminha@inria.fr; Nadjib Achir, Inria, Palaiseau, France and Université Sorbonne Paris Nord, Villetaneuse, France, nadjib.achir@inria.fr, nadjib.achir@univ-paris13.fr; Aline Carneiro Viana, Inria, Palaiseau, France, aline.viana@inria.fr.

0360-0300/2022/4-ART $15.00
https://doi.org/10.1145/3529758

constrained devices can consume edge resources with a lower communication latency than those associated with the cloud. Moreover, bringing computing and storage resources to the network edge can also benefit the Mobile Network Operators (MNOs) in many ways: higher QoE for users, finer network resource control and management, independence from big data centers, cost decrease.

Applications running on the edge of the network serve the applications running on devices. For this reason, there is no need for their traffic to reach the cloud, and the core of the network is then relieved from this traffic. The resources deployed on the edge of the network can offer different services to lower the operating costs or add value to the business. As a consequence, the edge resources enable or improve the usage of applications such as X-reality [28], autonomous driving [77], low-latency stream processing [87], to cite a few. Finally, one very impacting benefit favored by MEC deployment is the overall decrease in the energy consumption related to the Internet core usability.

To coordinate the efforts of bringing computing resources closer to the edge of the network, the European Telecommunications Standards Institute (ETSI) started designing the standard for Mobile Edge Computing in 2015 [31]. The specifications for MEC were firstly defined and uniquely intended for 5G networks. Later, the ETSI broadened its focus to consider other networking technologies and use cases (i.e., WiFi, LTE, MuLTEfireTM) [36]. Therefore, MEC is referred to as Multi-Access Edge Computing [61, 93] since 2018.

There are other paradigms offering resources close to the edge of the networks: transparent computing (TC) [109], fog computing [133], and Cloudlets [135]. The main distinguishing difference between these paradigms and MEC is the fact that MEC resources are usually tied to the telecommunication operator or the network administrator [102]. Typically, TC, fog, and Cloudlets resources are managed by different stakeholders. As a consequence, MEC is more network-aware than the other paradigms.

The convergence of the ETSI standard consolidates the discussions around MEC. The expected step before MEC‘s full adoption is the analytical survey of small and large-scale prototypes, enabling tools, and implementations. This was also the case for previous technologies, such as Software Defined Networks (SDN) [54, 55, 112, 115], outdoors-indoors localization [79], segment routing [134] and homomorphic encryption [4]. The early initiatives can reveal the potential, the limitations, the related technologies, and the development tools for MEC adoption, pointing to the directions to which MEC research and development should take in the following years.

The expectations around the MEC paradigm instigate a number of discussions, culminating in interesting literature. Therefore, it is possible to find surveys elaborating on different MEC aspects. In general, to the best of our knowledge, related surveys (1) discuss MEC's fundamentals, architecture, orchestration options [78, 127] as well standardization efforts [3, 101, 122], (2) enumerate computing- and communication-related models [83, 108, 137] or (3) investigate and compare other different edge paradigms [24, 144]. Besides, in an IoT context, some works survey the IoT-MEC relationship, in particular: (1) the applications and possibilities of MEC for IoT [102] and (2) the availability of edge computing systems for IoT [68, 96]. Table 1 lists the publication year and main contributions of MEC-related surveys in literature.

In this paper, we survey MEC literature from a practical point of view, a different perspective from related surveys mentioned here above. We aim to understand the functionalities that MEC systems offer and the issues targeted in different practical implementation initiatives. We center our efforts on the broad paradigm of Multi-Access Edge Computing systems. Therefore, this work offers the following novel contributions:

- We review different literature's visions and definitions of MEC and how researchers and developers implement MEC architectures, taking them from a more theoretical level to a more practical one.
- We particularly survey and compare MEC practical initiatives according to their strategies, including the ETSI MEC standard. We compare the strategies according to related limitations and impact. In the practical context, we target MEC working prototypes.

Table 1. Related literature surveys and their main contributions.

Year	Main contributions	Main differences to present work	Reference
2017	Edge evolution, use cases, and enabling Architecture, orchestration and deployment.	Enabling tools, architectures, technologies. and deployment in practice.	[127]
2017	MEC for computing offloading. Offloading decision and applications partitioning. Mobility issues.	Offloading implementations. Applications compatibility. Open issues on decision.	[78]
2017	MEC communication aspects. Task and communication models. Resource management.	MEC implementations communication aspects. Network adaptations.	[83]
2017	Comparison between fog, cloudlets, and MEC architectures. Expected implementations	MEC implementations, their goals, and design decisions.	[24]
2018	Applications, technical aspects, enabling technologies, and projects on MEC-enabled IoT.	Focus on MEC implementations, not related specific applications.	[102]
2018	MEC concepts, definitions, technologies, and architectures. Security and privacy.	Practical MEC aspects. Materialization of theory and implementation-related issues.	[3]
2018	Techniques and strategies for VM, containers, and services migration,	Focus on complete MEC implementations and architectures.	[137]
2019	Comparison between fog and related paradigms. Fog classification, frameworks, tools, and testbeds.	Focus on MEC standard implementations, their use cases and enabling tools.	[144]
2019	Mobility-related migration for edge. Migration architectures, prototypes, and simulations.	Focus on complete MEC standard implementations	[108]
2020	Review of the MEC research themes. MEC affinity with other technologies and paradigms.	Challenges, solutions, technologies, and design decisions of MEC implementations.	[101]
2020	MEC standardization. Provisioning and deployment on vertical industries.	Survey of MEC implementations regardless of targeted applications.	[122]
2020	Analysis of MEC for IoT. Commercial MEC systems for IoT applications.	Survey of MEC implementations regardless of targeted applications.	[68]
2020	Edge computing definitions and their general framework. Applications overview.	Definitions and use cases of existing implementations for MEC.	[96]
2021	MEC architectures and technical aspects for applications on augmented reality.	Architectures of MEC implementations regardless of the target application.	[120]
2021	Security aspects of MEC on 5G and projects for MEC security and privacy	MEC implementations regardless of security level and networking technology.	[104]

- We outline the deployment impact of the surveyed MEC systems and overview the tools employed in their implementation.
- We conclude this survey with a discussion on the open issues related to MEC systems, which provides a vision on the maturity level of MEC-related opportunities as well as on the development efforts.

We believe that a better understanding of MEC practical initiatives will help researchers and developers design their own MEC systems or improve and simplify the usability of existing ones.

For clarity, we illustrate the organization of this work in Figure 1. Section 2 discusses MEC basic concepts, the different definitions found in the literature, the use cases that existing MEC initiatives target, and the visions that drive their implementation. In Section 3, we describe the ETSI standard and its architectures. In Section 4, we survey the practical MEC systems in the literature, classifying them according to the followed strategies. Section 5 discusses the deployment effort for each MEC system. In Section 6, we describe the tools authors used to build the MEC implementations. In Section 7, we discuss the issues that MEC systems are yet to address. Section 8 concludes this paper. Table 2 lists the abbreviations we use in this text.

very short.pdf very short.pdf

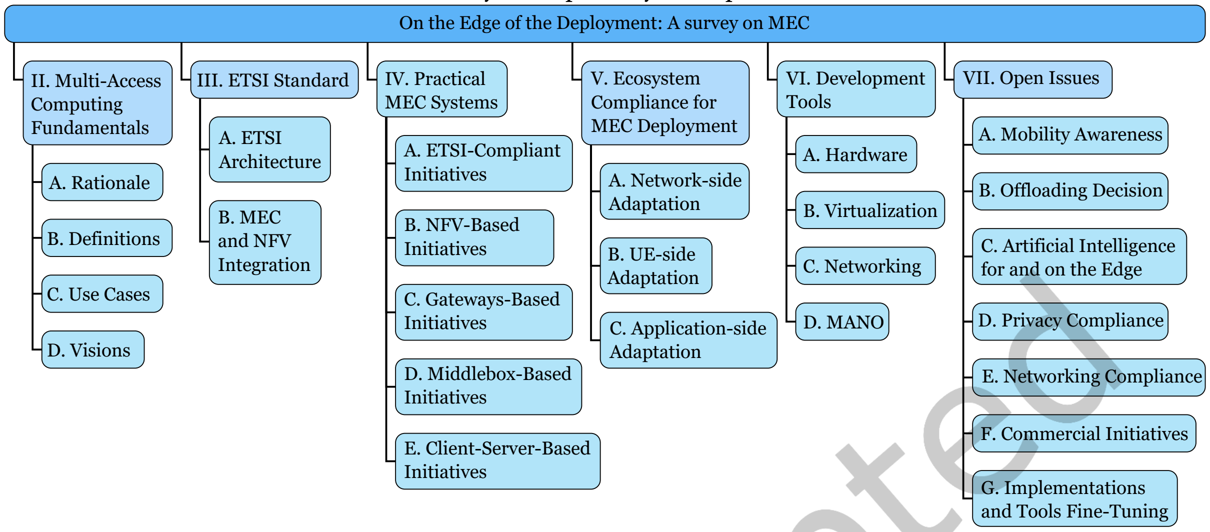
On the Edge of the Deployment: A survey on MEC
II. Multi-Access Computing Fundamentals
A. Rationale
B. Definitions
C. Use Cases
D. Visions
III. ETSI Standard
A. ETSI Architecture
B. MEC and NFV Integration
IV. Practical MEC Systems
A. ETSI-Compliant Initiatives
B. NFV-Based Initiatives
C. Gateways-Based Initiatives
D. Middlebox-Based Initiatives
E. Client-Server-Based Initiatives
V. Ecosystem Compliance for MEC Deployment
A. Network-side Adaptation
B. UE-side Adaptation
C. Application-side Adaptation
VI. Development Tools
A. Hardware
B. Virtualization
C. Networking
D. MANO
VII. Open Issues
A. Mobility Awareness
B. Offloading Decision
C. Artificial Intelligence for and on the Edge
D. Privacy Compliance
E. Networking Compliance
F. Commercial Initiatives
G. Implementations and Tools Fine-Tuning

Fig. 1. The outline of this paper.

Table 2. Abbreviations in this work and their meanings.

Abbreviation	Term	Abbreviation	Term
AP	Access Point	NFVO	Network Function Virtualization Orchestrator
CDN	Content Delivery Network	OS	Operating System
CFS portal	Customer Facing Service portal	OSS	Operations Support System
en-gNB	Enhanced Next Generation NodeB	QoE	Quality of Experience
EPC	Evolved Packet Core	QoS	Quality of Service
ETSI	European Telecommunications Standards Institute	RAN	Radio Access Network
GTP	GPRS Tunneling Protocol	SDN	Software Defined Networks
ICN	Information-Centric Networking	SEG	Service Execution Gateway
LCM	Life Cycle Management	SGW-LBO	Serving Gateway with local Breakout
MANO	Management and Orchestration	TC	Transparent Computing
MEC	Multi-Acces Edge Computing	UE	User Equipment
MEO	Multi-Access Edge Orchestrator	User App LCM Proxy	User application life cycle management proxy
MEPM-V	MEC Platform Manager for NFV	VAF	Virtual Application Function
MNO	Mobile Network Operator	VIM	Virtualization Infrastructure Manager
NAT	Network Address Translation	VM	Virtual Machine
NDN	Named Data Networking	VNF	Virtual Network Function
NFV	Network Function Virtualization	VNFM	Virtual Network Function Manager

2 MULTI-ACCESS COMPUTING FUNDAMENTALS – MEC

This section describes the context of MEC systems. We present a general model of the networks offering MEC services and the definitions related to MEC practical implementations. Besides, we discuss the scientific communities related to the research and development of MEC systems and the use cases that MEC can serve. In our discussions, we use the surveyed MEC systems as examples.

2.1 Rationale

As depicted in Figure 2, a general use case of a MEC system is a mobile user running a latency-sensitive and resource-intensive application within their User Equipment (UE). Since UEs are typically constrained in computing, energy, and storage resources, it might be convenient for the UE to leverage external resources. In a situation without using the MEC, the UE sends a request to the cloud. To reach the cloud, the request travels through the different networks between the UE and the server in the cloud that fulfills the request. This means that the UE has to wait for a delay from the instant when it performs a request until the instant it receives a response. We can divide this delay into three parts. The first is the time from the request to travel the network from the UE to the cloud server. The second is the time the cloud server takes to process the request and generate a response. The third is the time the response takes to travel from the cloud server to the UE. The first and third parts of this delay are named *network delay*.

When the UE runs latency-sensitive applications, the network delay between UE and the cloud might significantly degrade the user experience or even entirely hinder the application. A possible solution to fulfill the latency requirement of the application is that an application running on a cloud-like infrastructure topologically closer to the UE answers the request. Therefore, a MEC system is a system capable of running applications and services at the edge of the network, such that it is close to the UE.

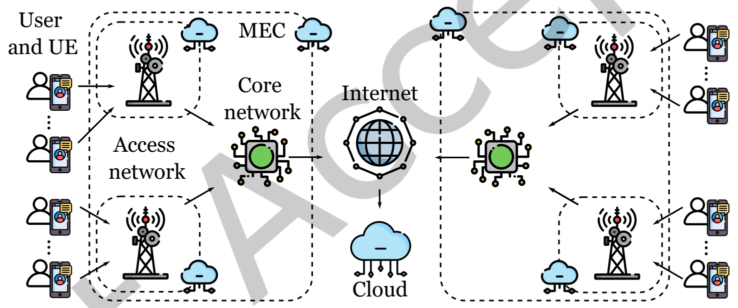

Fig. 2. The general use case of MEC Systems[1].

MEC is meant to work with different networking technologies, such as WiFi, LTE, and 5G. In Figure 2, we bring a general model of these networks. In this general model, we divide the network into three main hierarchical parts: the access network, the core network, and the Internet. MEC can operate in the access network or in the core network. Usually, the same MNO owns and operates these two networks. The function of each network depends on the used technology.

In WiFi, the access network is usually an access point (AP), installed inside users' houses, companies, or in public places. The core network forwards traffic from several users to the Internet. This means that MEC implementations working in the access or core network can behave similarly, mostly handling IP traffic.

In LTE (often called 4G), the Radio Access Network (RAN) serves as the first contact with the UE, and its main component is the eNodeB. The core network, namely Evolved Packet Core (EPC), forwards packets from the RAN to the Internet and deals with user mobility, billing, lawful interception, and other functions. The interface between the RAN and the EPC is the S1 interface [1]. The eNodeB encapsulates the traffic from each UE in a GTP tunnel (GPRS Tunneling Protocol) [2], which is decapsulated by the EPC before sending the packets through the Internet. Therefore, MEC implementations working in the LTE must deal with the tunneling and sending information to the LTE functions to work correctly.

Finally, in 5G networks, RAN slicing supplies RAN slice subnets [34, 37] and relies on optimizations of its main component, the en-gNBs (enhanced Next Generation NodeBs), i.e., the base stations directly connected to UEs. Similar to the LTE case, the en-gNBs also use GTP tunnels to send the data generated by UE to the core network. The 5G core network is different from the LTE core network, but it is also responsible for user mobility, billing, and lawful interception. Consequently, MEC systems operating in 5G have to deal with tunneling and communicating with 5G functions.

2.2 Definitions

Most of the survey literature defines MEC by referencing other paradigms that bring the cloud services physically and topologically closer to the UE [24, 83, 96, 102, 122, 127]. We believe this approach is essential to highlight the differences between MEC and other similar concepts. The comparison makes it possible to understand the possibilities and the limitations of each approach, their use cases, their evolution, and their perspectives for the future. To get a different view from MEC systems, we grasp from the literature the definitions researchers use when they claim to develop MEC implementations and prototypes. In this sense, we can discuss the *de facto* definition and how it reflects on the design of practical MEC systems.

ESTI defines a MEC system as the set of hardware and software components necessary to run mobile edge applications in the domain of an MNO [38]. Their definition also states that a MEC system consists of MEC hosts together with the management and virtualization infrastructure needed to support MEC applications. We detail the ETSI MEC standard in Section 3.

The MEC system and the MEC platform have different definitions according to the ETSI standard. Nevertheless, we observe in the literature that the term *platform* is often treated as an equivalent to the term *system*. Therefore, when comparing the definitions from different authors, we took the initiative of using the ETSI nomenclature to refer to the closest concept mentioned by each author.

As expected, all the works we review offer resources and make the deployment of applications possible to the edge of the network. In almost every case, these are virtual resources. The works that follow ETSI standard offer virtualized hosting to third-party applications and services regarding service discovery, context information, or network conditions to these applications [20, 38, 58, 124]. Some works offer optimized offloading for applications [17, 84]. Some other works help applications to fine-tune the network services according to their needs [131, 145]. Other works can perform traffic redirection to the benefit of the applications and of the MNOs [51, 52, 74]. We discuss the details of each of these works in Section 4.

Based on what we observe from the surveyed MEC practical implementations, we define the MEC systems as *network-aware infrastructures for application deployment at the edge of the network*. Because of their network awareness, MEC systems can extrapolate the provision of computing infrastructure, creating an ecosystem of services to optimize what deployed applications can offer.

2.3 Use Cases

MEC can help to solve different categories of problems. These problems are often modeled as use cases and, more specifically, applications. The ETSI defines certain use cases for MEC, divided into the categories: (1) consumer-oriented services; (2) operator and third-party services; and (3) network performance and QoE improvements [36]. In the following, we describe the use cases and the applications found in the MEC implementations so far.

The consumer-oriented services are the ones that improve the end-user experience directly. However, these services are considered too computationally intensive to be executed by the UE and too latency-sensitive for

[1]Figure 2, Figure 3, Figure 4, and Figure 5 have been designed using resources from Flaticon.com.

execution in the cloud. Among the surveyed MEC systems, ACACIA [17] implements an X-reality [82] application that leverages MEC computing, feeding the MEC with the context gathered in the UE. MEC-ConPaaS [131] implements another example of consumer-oriented service. In their service, the UE captures video that contains text in a language unknown to the user and sends it to the MEC application. The MEC application identifies the text in the video and then translates the text.

The operator and third-party services use the MEC infrastructure to create services that are not directly aimed at the end-user. They offer services to the applications that are end-user-oriented. These services can take advantage of the low latency of MEC infrastructure, but also of the redundancy reduction of generating a single service that can serve several users.

One important use case is related to smart objects, since they are often resource-constrained, working for latency-sensitive applications. OpenNESS offers an example application for smart cameras [57], while LightEdge focuses their proof-of-concept in an autonomous driving application [20]. The work from Cattaneo *et al.* implements a MEC application that converts the format of videos as a network service, for UEs or IoT cameras. The MEC also helps to distribute the video to users that are close to the streaming device [14]. Application offloading can further improve UE battery life and even create multi-platform compatibility. The implementation of eRAM makes it possible that applications offload their tasks to the MEC [84].

The network performance and QoE improvements are oriented to enhance the network and the QoE without offering new applications or services to end-users. These services can reduce the costs of the MNOs while improving the network's efficiency. Information-Centric Networking (ICN) is a service that keeps track of the location of information, so users can search for information, instead of searching for its location [143]. PiCasso implements an ICN using smart MEC gateways [72]. Content Delivery Networks (CDNs) are related to ICNs, working in the application layer and using caches to provide content to users [100]. OpenNESS provides a certain CDN application for caching, as a proof-of-concept [57]. The work from Li *et al.* uses MEC to cache applications of web browsing, audio, and video streaming [74]. The work P4EC uses the computing capabilities at the edge of the network to perform traffic offload decision [51], optimizing the network usage.

2.4 Visions

The existing practical MEC systems place different effort levels to tackle different challenges. This section discusses the point of view of two distinct scientific communities – i.e., the networking and distributed systems community – that might help grasp the expectations that every implementation aims to meet. While the networking community considers MEC as a network service, the distributed systems community is most concerned about MEC application execution and the related entities' integration. Of course, these points of view do not limit the systems, and works guided by one of the visions also address other points of view. We use exemplify with MEC implementations from the literature, which we detail in Section 4.

The vision of the networking community: The networking community regards MEC mainly as a service that the network offers to the users. These users can be either the final users or the developers that want to deploy their applications at the edge. Consequently, this vision recognizes a central role for the MNOs. Works that share the network vision often take advantage of the MEC knowledge about the network to improve the network itself. In this sense, MEC can run and provide input for routing and cache applications or as infrastructure for Network Function Virtualization (NFV) [29]. In this last scenario, MEC hosts can run Virtual Network Functions (VNFs), such as firewalls, virtual routers, and network address translation (NAT). In Section 3, we discuss further the association between NFV and MEC.

The MEC ETSI standard is an example of the network vision. It recognizes a central role to the MNO and concentrates resources in the network's services while using MEC. Naturally, the works implementing MEC as defined by ETSI are also examples of the networking community vision [14, 20, 58, 124].

Table 3. Approaches of the different communities.

ETSI-compliance \ Community	Network	Distributed Systems
Compliant	[20, 58, 124] [14, 145]	–
Non-compliant	[14, 51, 74] [10, 72, 117]	[17, 84] [131]

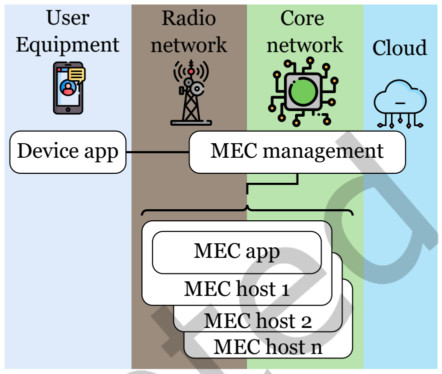

Fig. 3. MEC general use case[1].

There are also non ETSI-compliant MEC implementations that follow the networking community vision. The works that implement MEC as a middlebox, improving network efficiency [51, 74], the works that integrate MEC applications and VNFs [10, 14, 117], and also the work using smart gateways to offer MEC services [72]. According to the networking community, these works reflect the entanglement between MEC and the other elements managed by the MNOs. However, M^2EC [145] is between the two communities. We consider this work part of the network community because M^2EC recognizes MEC as a network service. Nevertheless, the fact that it works in the integration between MEC applications and the applications running in the UE brings this work close to the distributed systems community vision.

The vision of the distributed systems community: Here, MEC is a distributed system in which MEC hosts should cooperate to offer network-aware offloading services with low latency. To this vision, MEC does this by serving users and applications directly with resources or with more high-level services. Additionally, MEC is also an element of a more large distributed system composed of UE, MEC, and cloud. This approach raises questions regarding the execution of the MEC applications, orchestration, resource sharing, battery life saving, and user experience, to cite some.

The work MEC-ConPaaS [131] aims to facilitate the deployment of MEC applications modeling MEC applications as a collection of services, orchestrating services among different applications the services that are common to them. ACACIA [17] and eRAM [84] focus on the integration between UE applications and the MEC. These works place an effort in helping the UE applications to offload their computation to resources in the MEC system.

The networking community and the distributed systems community visions steer MEC implementations to behave more as a network service or a service offering distributed computing resources. Even though compatible, these visions affect the problems and the proposed MEC solutions.

MEC presents a heterogeneity of definitions, use cases, and visions. To create a more uniform environment, ETSI defined a standard, a definition, and a set of use cases for MEC [31]. Nevertheless, not every implementation follows the ETSI standard. Table 3 divides the works within the network and the distributed systems, indicating whether they are ETSI-compliant or not. Since the ETSI standard is network-oriented, no works sharing the distributed systems community vision implement the standard. In the next session, we discuss the ETSI standard before presenting the different strategies for implementing MEC systems.

3 ETSI STANDARD

There are different initiatives to bring computing resources to the edge of the network [119, 133, 135]. The ETSI defines a standard that serves as a ground stone for authors to build their MEC systems, for MNOs to deploy their infrastructure, and also for MEC application developers to understand the available services. The main idea is to provide compatibility between applications, implementations, and building blocks.

As described in Section 2.3, the standard is meant to operate within a large number of use cases. This means that a great number of protocols should influence its final standardization. For instance, ETSI states that MEC architecture should take into consideration the 5G, the TCP, the V2I, SDN, and NFV [36]. Additionally, literature shows that it is also important to consider the evolution from 4G to 5G [44], the standards involving cloud services [39], and IoT application protocols [26].

To tackle the challenges, the standard describes two architectures. One guides the implementation of a stand-alone MEC system, the other guides an implementation based on NFV. This section presents and discusses the ETSI standard for MEC, together with its two architectures.

As discussed in Section 2, ETSI defines a Multi-Access Edge System (MEC system) as the set of hosts and management infrastructure necessary to run MEC applications on a network. Additionally, ETSI defines a Multi-Access Edge Platform as a collection of services that enable mobile edge applications, also working as an interface between the MEC applications and the rest of the MEC system [31]. ETSI also defines MEC applications as applications that can take advantage of MEC resources, including its infrastructure and services. The user MEC applications are the applications instantiated in the MEC system to fulfill requests from UEs. ETSI standard requires that MEC offers at least radio network information, location, and bandwidth manager services. MEC applications can offer services themselves suited for other applications. Since the standard does not limit the number of services, implementations can offer additional services.

In a typical scenario, as we illustrate in Figure 3, an application running in the UE (Device App) requests the MEC management for the instantiation of a latency-sensitive application. The MEC management then decides whether to fulfill the request or not, based on MNO's policies. The MEC management also decides which of the MEC hosts should run the application or replicate the MEC application on several MEC hosts. The MEC management then instantiates the MEC application in the selected MEC host(s). Finally, the MEC management configures the traffic redirection on the network so that the UE can reach the MEC application. Even though MEC hosts are on the access network or the core network, the MEC management can be in the cloud. This is because the management of MEC applications is not always latency-sensitive. The ETSI architecture divides the MEC into different entities that run closer or further from the end-users, executing the necessary tasks to implement a MEC system [38].

3.1 ETSI Architecture

The ETSI MEC architecture divides a MEC system into two levels: the MEC system level and the MEC host level. These levels hold the building blocks that compose the MEC system. We illustrate this architecture on Figure 4. By convention, the reference points labeled as *Mp* are related to the MEC platform functionality; the reference points labeled as *Mm* are interfaces related to the management of the platform; the reference points *Mx* provide an interface with external entities. In the sequence, we define each of the entities in this architecture as well their interactions, also depicted in Figure 4.

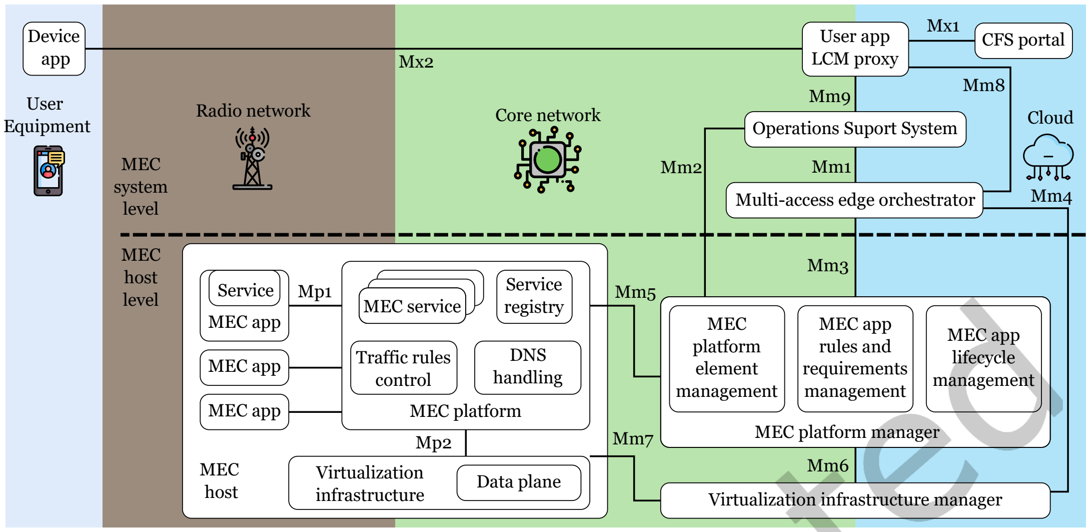

Fig. 4. ETSI MEC architecture[1]. Adapted from [38].

The MEC system level manages the different hosts of a MEC system. It is expected that a MEC system controls several MEC hosts. In this sense, the MEC system level holds the following entities that are not replicated to all the MEC hosts.

- **Device application:** (*Device app*) it is any application running in the UE that is capable of interacting with the MEC. It can request the User application life cycle management proxy (*user app LCM proxy*) to instantiate a MEC application.
- **Customer Facing Service portal:** (*CFS portal*) receives requests regarding the instantiation or termination of MEC applications. The *CFS portal* forwards these requests to the Operations support system (*OSS*) of the MNO through the *user app LCM proxy*.
- **User application life cycle management proxy:** (*user app LCM proxy*) is the entity that receives requests from the applications running on the UE to trigger the instantiation, termination, and relocation (when supported) of MEC applications. The *user app LCM proxy* also exposes to the UE the state of the applications running in the MEC system. It interacts with the *OSS* to make sure that the MNO authorizes the fulfillment of the requests.
- **Operations support system:** (*OSS*) is the entity responsible for receiving and validating the requests from UE applications or from the *CFS portal*. The *OSS* decides whether it should grant each request or not, based on the MNO policies. When the *OSS* authorizes a request, the OSS forwards the request to the Multi-access edge orchestrator (*MEO*) or to the *MEC platform manager*. The *user app LCM proxy* can bypass the *OSS* for already authorized operations.
- **Multi-access edge orchestrator:** (*MEO*) is the functional block that keeps a global overview of the MEC system. Additionally, it receives the MEC applications packages, validates them, and chooses the MEC host(s) to allocate each application. It also initiates the instantiation, termination, and possible relocation of the applications. When the *OSS* authorizes, the *MEO* communicates to the *MEC platform manager* to instantiate a MEC application, authorizing the Virtualization infrastructure manager (*VIM*) to orchestrate the resources.

The MEC host level is composed of the following entities instantiated to each MEC host.

- **MEC platform manager:** this entity performs three tasks. The first consists of managing the life cycle of applications, communicating to the *MEO* the relevant events. The second relates to providing and exposing the entity management functions to the MEC platform. The third is to manage application rules and requirements, such as traffic rules or service authorization. When triggered by the *MEO* and authorized by the *OSS*, the *MEC platform manager* orders the MEC platform to instantiate a MEC application, informing the Virtualization infrastructure manager *VIM* about the necessary resources.
- **Virtualization infrastructure manager:** (*VIM*) this entity manages the virtualization life cycle of the MEC applications (allocation, instantiation, releasing), applying the rules of the *MEO* and of the *MEC platform manager*. If the MEC system supports application relocation, the *VIM* is responsible for relocating the application to another host or the cloud. When it receives from the *MEC platform manager* a request for virtual resources, it checks with the *OSS* for authorization and, if positive, sets the virtual resources in the *MEC host*.
- **MEC host:** an entity providing compute, storage, and network resources to the MEC applications. To achieve that, it runs a virtualization infrastructure as well as a *MEC platform*. A MEC system is supposed to have at least one, but often many more *MEC hosts*. In addition, it hosts the MEC applications services.
- **MEC platform:** offers a service registry, so applications can advertise, discover, consume, and offer MEC services. When triggered by the *MEC platform manager*, the *MEC platform* instantiates a MEC application or a MEC service in the *virtualization infrastructure* and also configures the data plane of the virtualization infrastructure. According to the traffic rules from the *MEC platform*, internal or external entities should reach a certain MEC application or service. The data plane configuration enforces the *MEC platform* traffic rules.
- **Virtualization infrastructure:** offers the compute, storage, and network resources to the MEC applications. The data plane routes the traffic between MEC applications and all the other entities, applying the rules received from the MEC platform. It receives requests for resources from the VIM and data plane rules from the MEC platform.

The stand-alone ETSI MEC architecture uses many services and structures that are similar to services and structures that already exist in the context of the MNOs. The MNOs use these services and structures to implement NFV. For this reason, the ETSI also designed an architecture joining MEC and NFV.

3.2 MEC and NFV Integration

To run MEC applications in the network, a MEC system needs to manage a virtualization infrastructure, instantiate applications in this virtual infrastructure, install the correct data plane configurations, manage the applications life cycle, and offer MEC services. These requirements are very similar to the requirements to implement the Network Function Virtualization (NFV).

NFV is a paradigm that decouples the network functions from the physical equipment running these functions [89]. The advantages of decoupling are threefold. First, making network functions independent from the hardware executing them makes their evolution and maintenance independent from the hardware evolution and maintenance. Second, the deployment of functions is more flexible since software deployment is more flexible than hardware deployment. Third, the scaling of network functions is also more flexible since the virtualization can grow or shrink the hardware slice executing each function [90].

The implementations of NFV provide some virtualization infrastructure and services. For example, developers can deploy their software that performs network functions, namely Virtual Network Functions (VNFs). In addition, NFV infrastructure is responsible for the Management and Orchestration (MANO) of the VNFs, life cycle management, and traffic redirection. These duties have a significant intersection with the ones expected from a MEC implementation.

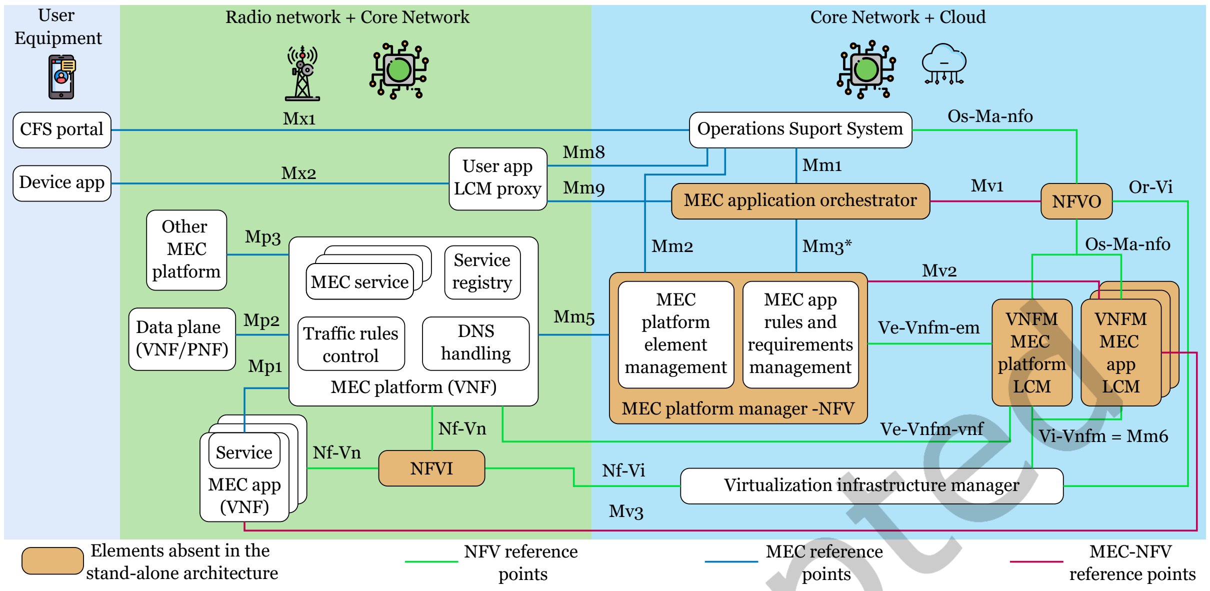

Fig. 5. MEC-NFV architecture as proposed by ETSI[1]. Adapted from [38].

The NFV design and adoption predates MEC conception and framing. For this reason, MEC implementations can reuse and take advantage of NFV deployments and their existing software and hardware infrastructure. There are many design possibilities and, to provide compatibility, ETSI also proposes a standard to implement MEC and NFV together.

In the ETSI integration, the MEC platform is instantiated as a VNF, as well as the MEC applications [35]. The MEC platform can delegate some management functions to the NFV MANO Life Cycle Management (LCM). According to the standardization, the NFV MANO remains unchanged to support the deployment of MEC applications. The MEC platform, running as a VNF, offers MEC-related functions. It can implement the functions itself or leverage the management and orchestration services that the NFV MANO provides.

We illustrate the architecture in Figure 5, adapted from the ETSI standard [38]. In the following, we list the entities from the MEC in NFV that are not in the original MEC standard or that are somehow modified.

- **MEC application orchestrator:** logically, this entity has the same functions as the *MEO*, from the stand-alone MEC architecture. The difference is in the implementations domain since it only implements MEC-specific functions. It delegates to the *NFVO* functions that are general to the orchestration of MEC applications and VNFs.
- **MEC platform manager - NFV:** (*MEPM-V*) this entity performs the same functions as the *MEC platform Manager* from the original architecture. Nevertheless, it does not implement LCM functions, delegating them to the *VNF Manager*.
- **Network Function Virtualization Orchestrator:** (*NFVO*) this entity is defined in [30]. It manages the life cycle of the VNFs. It orchestrates MEC applications the same way it orchestrates VNFs.
- **Virtual Network Function Manager:** (*VNFM*) this architectural entity is defined in [30]. This entity manages the life cycle of VNFs. In the case of the MEC-NFV architecture, it manages the life cycle of the *MEPM-V* and the MEC applications.

- **Network Function Virtual Infrastructure:** (Network Function Virtualization Infrastructure) this entity provides the virtualization infrastructure for the VNFs [30]. In the case of the MEC-NFV architecture, it is blind if it manages a VNF or a MEC application.

The ETSI MEC is an important reference for MEC authors. It defines two architectures, one stand-alone and one NVF-based. It also defines building blocks and interfaces to instantiate MEC applications and to manage their life cycle. Even though the ETSI standard is a relevant reference, not every initiative follows the ETSI standard. We discuss next the implementations of MEC and their different guiding concepts.

4 PRACTICAL MEC SYSTEMS

To implement a MEC system, it is essential to have a guiding concept or strategy. The ETSI architecture is an example of such a guiding concept. Nevertheless, the ETSI architecture is not the only possible guiding concept. Table 3 lists the implementations that are ETSI-compliant and the implementations that are not ETSI-compliant, following some other strategies. This section describes the MEC implementations found in the literature, classifying them according to the concept guiding their implementation. We identify five main concepts. The first concept follows the ETSI standard. The second leverages the NFV infrastructure to implement a MEC system. The third enhances gateways, making them smart enough to run MEC applications. The fourth adds an agent to the UE to optimize the offloading to the MEC. The fifth strategy consists of developing middleboxes that execute MEC applications. Table 4 lists the strategies, the name, a short description, and the references for each implementation we survey in the following.

4.1 ETSI-Compliant Initiatives

As we describe in Section 3, a MEC system is the set of hosts and management infrastructure to enable MEC applications, while the MEC platform is the part of the system that provides the services for MEC applications. We consider that a system is ETSI-compliant if it follows the definition of the ETSI architecture at least to some level. Hence, ETSI-compliant systems inherit the properties of the ETSI architecture, especially the compatibility with LTE and 5G environments. In the following set of works, some implement the whole MEC system, and some others implement just some entities of the ETSI architecture.

LightMEC implements the ETSI architecture of a MEC system [124]. LightMEC models the mobile edge services running on the mobile edge platform as Light Virtual Network Functions. With this model, it is possible to use Network Function Virtualization (VNF) tools to deliver the services. LightMEC hosts run MEC applications using a container-based infrastructure. The authors design the deployment of lightMEC on the aggregation points of the LTE network, intercepting the packets in the GTP tunnels that traverse the aggregation points. They use the messages exchanged in the attachment and handover procedures to discover the state of UE and to properly forward incoming packets. When a UE performs a request, lightMEC uses DNS to verify whether some hosted MEC applications can serve the request. If one of the MEC applications can serve the packet, the traffic is forwarded to the application. Otherwise, the request is rerouted to its original path. LightMEC implements the MEO, the MEC platform manager, the VIM, the MEC platform, and lightweight virtualization infrastructure.

LightEdge is designed to work in LTE and 5G environments so that it can operate during the transition between the two technologies [20]. Their architecture implements an ETSI-compliant MEC platform and the virtualization infrastructure. LightEdge also implements a MEC Platform Manager, but it is not ETSI-compliant. LightEdge sits on top of the S1 interface and hosts copies of applications that originally run in the cloud. As we discuss in Section 2, the S1 interface is the contact between the RAN and the core network in the 4G. LightEdge decapsulates the traffic passing by S1 and uses DNS to intercept traffic that the hosted applications can serve. Among other things, LightEdge offers RAN information for hosted applications and a REST interface for Operation Support

Table 4. Surveyed MEC initiatives in a glance.

Strategy	Name	Short description	Reference
ETSI-Compliant	LightMEC	Lightweight virtualization, with services as light VNFs	[124]
	LightEdge	ETSI MEC system with traffic decapsulation for S1 interface	[20]
	M^2EC	MEC broker for users to access MEC management	[145]
	MEC-NFV with LBO	MEC platform as a VNF, with SGW-LBO deployment	[14]
	OpenNESS	Infrastructure management and MEC application orchestration	[21, 58]
NFV-Based	MANO+	MANO for VNF and MEC applications orchestration	[117]
	NFV-based MEC with Open Baton	VNF-capable MEC infrastructure	[10]
Gateway-Based	MEC-ConPaaS	PaaS in the edge with low cost single-board computers	[131]
	Container-Based MEC for IoT	Enhanced gateways for container-based virtualization	[52, 53]
	PiCasso	MEC on information-centric community mesh networks	[72]
Middlebox-Based	MEC as a middlebox for LTE	Cloud application replication in the edge, with traffic redirection	[74]
	P4EC	Traffic offload from the core network	[51]
Client-Server-Based	ACACIA	MEC with UE client for MEC services discovery	[17]
	eRAM	MEC with UE client for offloading management	[84–86]

Systems (OSS). The system also proposes a strategy for billing, sending information to the respective entities of the LTE that are in the core network.

M^2EC is an orchestrator for MEC systems [145]. It is not a complete system but a building block of a MEC system. They propose a MEC Broker, an entity that can grant privileges for MEC tenants (i.e., third-party developers) over the infrastructure. The broker exposes users to the $Mm1$, $Mm2$, and $Mm8$ interfaces of the ETSI MEC architecture. This allows users to have access to the MEC orchestrator and the MEC platform manager. The broker decides which requests to fulfill based on the privileges and policies related to each user. The MEC Broker is located between the UE and the rest of the ETSI architecture. M^2EC is, therefore, designed to change the ETSI architecture and the way applications interact with it. To ensure compatibility with ESTI MEC, the authors suggest that the broker is implemented as an extension of the CFS portal, the OSS, the MEC orchestrator, the MEC platform, and the User app LCM proxy working together.

MEC-NFV with local breakout, from Cattaneo *et al.*, implements a MEC platform using the NFV integration standard [14]. They use an approach called "Distributed Serving Gateway with Local Breakout" (SGW-LBO), one of the deployment options described by the ETSI standard [45]. Cattaneo *et al.* implements a MEC platform as a VNF running in an NFV infrastructure. Their work focus on a video streaming application that can efficiently offload the video processing from the UE to the MEC. The UE records the video and uploads it to the edge. They show that the upload time to the edge is several times smaller than the upload time to the cloud. To process the video, the MEC application can use GPUs at the edge. Therefore, Cattaneo *et al.* uses a descriptor to indicate to the NFV orchestrator and the VIM that the application needs GPU nodes.

OpenNESS implements a MEC system that offers a virtualization infrastructure, a data plane and a virtualization manager [21, 58]. OpenNESS also implements the ETSI MEC platform and the ETSI MEC platform manager. Users can deploy applications, and OpenNESS handles the virtualization. It also installs tools to manage the virtualization aspects of ETSI architecture. OpenNESS has two distributions: one is fully open-source, and the other is a licensed distribution from Intel®. For this reason, some functions are optimized to run on Intel® hardware.

4.2 NFV-Based Initiatives

As discussed in Section 3.2, the ETSI standard has an architecture for MEC-NFV integration [38]. Nevertheless, MEC implementations can take the present NFV infrastructure to their advantage but not abide by the standard. Another alternative is to use MEC infrastructures to run VNFs, using the MEC implementation to replace the NFV infrastructures.

MANO+ is an architecture enhancement to allow the traditional NFV MANO to orchestrate and manage VNFs and MEC applications [117]. The MEC applications are modeled as Virtual Application Functions (VAFs), which describe the application's needs from the point of view of the orchestrator. Since NFV MANO and MEC are coupled to each other, they can cooperate to achieve different optimization levels. The main difference between the ETSI NFV-MEC standard described in Section 3.2 and MANO+ [117] is that in the ETSI proposal, the MEC orchestration is deployed as a VNF. In contrast, in their work, the MEC orchestration is deployed in a modified version of the NFV MANO. The main advantage of the ETSI approach is that it is compatible with previous versions of NFV MANO.

NFV-based MEC with Open Baton is a prototype by Carella *et al.* of a MEC infrastructure to deploy Virtual Network Functions (VNFs) [10]. Their work implements an interface that allows the VNF MANO to instantiate containers in the MEC infrastructure and run VNFs as MEC applications. Since the MEC system manages the virtualization infrastructure, it goes in the opposite direction of the ETSI standard that instantiates the MEC applications in the NFV infrastructure.

4.3 Gateways-Based Initiatives

A common strategy is to use gateways as MEC hosts. This means that gateways are enhanced to be capable of instantiating MEC applications. Since coordination between gateways is possible, a gateway that receives a request does not need to treat the request. An advantage to this approach is that gateways are already deployed and often can run applications. A disadvantage is that gateways are usually limited in resources, which means that they can not run very intensive applications.

MEC-ConPaaS offers a PaaS running on single-board computers [131]. They argue that the low cost of single-board computers can make it possible to offer computing resources in the same spot as the radio access, simplifying the system's architecture. In their architecture, hardware and a network layer offer computing, storage, and networking to a container-based virtualization layer. A cloud computing layer orchestrates the containers. On top of these layers, the ConPaaS layer manages the deployment of MEC applications.

Container-Based MEC for IoT is designed by Hsieh *et al.* and targeted to IoT applications [52, 53]. The authors enhance IoT gateways, providing container-based virtualization with gateway hardware. Then, these containers host functions related to the data flow between IoT equipment and the cloud.

PiCasso is a MEC system designed to deploy MEC services on information-centric community mesh networks [72]. PiCasso proposes a special type of node, called *Service Execution Gateway* (SEG), and incorporates these nodes into information-centric community mesh networks. These gateways are single-board computers capable of executing computing and storage services. PiCasso builds a virtualization layer on the SEGs, enabling the deployment of containers to offer services. The main component of PiCasso is its decision engine that chooses the SEG in which to deploy each service. The decision engine bases its decision on the service specifications and the availability of the hosting devices. The availability is an important metric because nodes of community mesh networks are prone to failure. A Service Controller stores all the possible services and installs the services in the edge hosts, following the decision engine. The services are exposed to the users using the Information-Centric Networking (ICN), decoupling the services to specific hosts.

4.4 Middlebox-Based Initiatives

Some works implement MEC as a middlebox. A middlebox is an entity in the path between source and destination host, performing any function on data that is not a normal IP function [11]. Middleboxes are usually conceived as independent and self-contained. These attributes ensure minimal interference with the rest of the network entities. The ETSI MEC architecture is a distributed system, where a single MEC orchestrator deals with many MEC hosts. In the MEC middlebox architectures, each middlebox works as an independent MEC system with a single MEC host. This brings some challenges to the middlebox approach related to scalability, load balancing, and user mobility.

MEC as a middlebox for LTE, from Li *et al.*, is a MEC system located on the S1 interface [74]. Their middlebox hosts copies of applications running in the cloud. Then, it intercepts requests from the UE to the applications in the cloud, redirecting the request to the local applications. One important thing is that UE's IP packets are encapsulated in GTP packets when they traverse the S1 interface. Therefore, the middlebox deals with depackaging and repackaging these packets when no hosted application can answer the request.

P4EC is another approach for edge as a middlebox [51], addressing some issues not regarded in [74]. In their work, the MEC application offloads traffic from the core network. P4EC works in the interface between the radio access network and the core network. Additionally, P4EC has a local exit that is capable of bypassing the core network. P4EC recognizes delay-sensitive traffic using the transport layer header. It also receives from the core network a list of the UEs that are authorized to perform traffic redirection and bypass the core network. When delay-sensitive traffic comes from authorized UE, this traffic is redirected to the local exit. The authors show that this approach can reduce the latency when compared to the approach of [74].

4.5 Client-Server-Based Initiatives

The QoE is very sensitive to the interaction between the UE applications and the MEC applications. Additionally, MEC applications can significantly improve their performance if they have context information about the UE. Some implementations use applications running in the UE as clients to the MEC to improve the performance of the MEC implementation as a whole.

ACACIA implements a device manager that helps the interaction between applications running on UE and applications running on servers in the edge [17]. ACACIA works with a device manager that runs in the UE and registers the UE interests in the edge services. When there is a match between the services UE requirements and the MEC server's services, ACACIA offloads the computation to the server. Additionally, ACACIA provides user context to the MEC application, arguing that this information is crucial to lowering the latency of the user's experience.

eRAM uses a client offloading middleware running in the UE to identify applications to offload [84–86]. The middleware runs a profiler that keeps track of each application's hardware and network resources usage and decides to offload the application to a MEC host.

5 ECOSYSTEM COMPLIANCE FOR MEC DEPLOYMENT

MEC systems operate inside networks, interacting with infrastructure, devices, and applications that are already developed. To deploy these MEC implementations, modifying the network infrastructure, the UE, or the applications to some level might be necessary. In this section, we discuss such modifications required when deploying MEC systems. In each subsection, we organize the implementations that require fewer adjustments to existing infrastructures to the ones that require more adjustments.

5.1 Network-Side Adaptation

The edges of the network deliver the MEC. Hence, changes to the network are inherent to a MEC implementation. We can think of a network as connected elements. A MEC implementation deployment might affect the elements of a network, the connections between them, or both. The strategy each implementation uses can significantly influence the level of network changes required to deploy the implementation.

The works that implement MEC as a middlebox [51, 74] require the insertion of MEC in some interface of the networks. These implementations are specifically designed to have a minimal impact on the other network elements, making them very easy to deploy.

MANO+ [117] and the work from Carella *et al.* [10] make changes to the existing NFV infrastructure, taking advantage of it to deploy a MEC system. Therefore, the deployment of these implementations is a software update to the existing NFV infrastructure. Of course, this is only possible in networks that already have an NFV deployment. Once this is achieved, the effort to deploy the MEC system is a software update.

The ETSI standard dictates that authors can deploy the MEC system inside the RAN, the core network, or sitting on top of some interface. Among the ETSI-compliant works, LightMEC [124] and the work from Cattaneo *et al.* have some specific deployment options. The other ETSI-compliant works [20, 58, 145] can modify existing elements of the networks, such as gateways or nodes in the core network. The same goes for the work ACACIA [17]. Nevertheless, these implementations can also sit on an interface and behave similarly to the middleboxes. Given this reasoning, their deployment effort is subjected to the deployment complexity.

PiCasso [72], eRAM [84], MEC-ConPaaS [131], and the work from Hsieh *et al.* [52] implement smart gateways that are capable of serving MEC applications to the UE or to IoT objects. This means that the area covered by MEC should also be covered by these smart gateways. In order to mitigate this impact, eRAM [84], MEC-ConPaaS [131], and PiCasso [72] use single-board computers to act as both as MEC hosts and gateways. These single-board computers are low cost and, therefore, have low deployment costs. Nevertheless, their deployment effort requires updating a number of gateways or, in a more complicated arrangement, even changing equipment.

MEC implementations adopt some measures to cope with the different networking protocols. The work from Hsieh *et al.* [52] and ACACIA [17] use OpenFlow for traffic redirection. The protocols for DNS are important for LightEdge [20] and Li *et al.* [74]. The works from Li *et al.*, ACACIA [17], and LightEdge [20] use the GTP to intercept traffic from the network. All the implementations must adapt their interfaces to the protocols used in their ecosystems.

5.2 UE-Side Adaptation

The UE is in the best position to know its local resources state. Therefore, it is necessary to make changes to the UE to give it the possibility to decide whether it is more interesting to offload an application or not, according to its requirement and locally available resources. The ETSI standard poses no modifications to the UE. This facilitates the MEC adoption from the UE developers perspective. Furthermore, it is possible to develop ETSI-compliant MEC applications that can benefit from UE changes if or when they are available.

ACACIA [17] runs a device manager inside the UE that controls which part of each application ACACIA should offload to a MEC host. The work MEC in NFV for Immersive Video uses a UE application to help in the interaction with the MEC application [14]. This means that users must download and install specific applications in the UE before being able to benefit from the MEC application.

Changing the UE can prove risky if it requires user intervention. The implementation of eRAM [84] runs a daemon in the operating system (OS) of the UE, something that can be present without the intervention of the user. This daemon follows the resources' usage of each application and sends the applications to offload in the MEC. To implement this modification, it is necessary to update the OS running in the UE. An OS update is not highly complex, but it is not as simple as installing or updating an application.

5.3 UE-Application Adaptation

Applications running in the UE are usually designed to interact with the cloud. MEC offers possibilities that are different from those in the cloud. To interact with the MEC, it might be necessary to add some changes to the applications. For instance, UE applications can consider communication delay when deciding whether to perform a request to different servers. Another option is that UE applications have annotations into their code to explicitly inform the OS about the possibility to execute such a block in a MEC host.

A possible strategy to avoid UE application changes is to use the MEC to replicate the cloud [52, 74]. This way, developers can benefit from the MEC lower latency without modifying the UE application.

Van Lingen *et al.* require that the MEC receives a model from the UE applications, so their orchestrator can decide which MEC resources should run the applications [132]. Van Lingen *et al.* uses the language YANG to model the UE applications [56]. This means that the application developers have an additional burden, and older applications must be adapted.

6 DEVELOPMENT TOOLS

The implementation of MEC systems requires tools, either hardware or software, to perform different tasks related to the MEC services. For instance, it is necessary to have hardware for computing power and software for virtualization and management. This section lists the tools employed in the implementation, development, and testing of the MEC systems we survey in this paper. Table 5 presents the complete list of the tools, with their names, a brief description of the tools, the works that use each tool, and a reference to a paper describing the tool or to a website where the tool is available. We also divide the tools according to their type or, more specifically, their role in the MEC implementations. In the following section, we follow the sequence presented in Table 5 to detail the role each tool plays in the different implementations.

6.1 Hardware

Most of the surveyed works do not target specific hardware. Any general-purpose hardware can run their implementations. However, some implementations develop MEC functionalities that target low-cost hardware.

Raspberry Pi is a single-board computer with a quad-core, 1.2 GHz 64 bit CPU and 1 GB RAM [106]. The works eRam [84–86], MEC-ConPaaS [131], and PiCasso [72] use Raspberry Pi 3 Model B as hardware infrastructure. eRam deploys Android virtual machines in the Raspberry to replicate the UE. Then, eRAM runs the Android-native applications Linpack [25], CPUBENCH [123], and PiBench [71], that make floating-point calculations such as finding the n^{th} digit of π. PiCasso and MEC-ConPaaS deploy container-based virtualization services and run applications on top of it. PiCasso runs ApacheBench [129] and Cloudsuite Web Serving benchmark [99], both web servers benchmarks. MEC-ConPaaS uses a face detection application [63] working on top of Apache Flink [128].

Table 5. Tools used by the surveyed MEC initiatives.

Type	Name	Short description	Works that use	Reference
Hardware	Raspberry Pi III	Single board, low cost computer	[72, 84, 131]	[106]
Virtualization	Kubernetes	Orchestrator for container-based virtualization	[20, 58, 124]	[18]
	Docker	Container-based virtualization engine	[72, 124]	[88]
	KVM	Virtualization for Linux distributions	[14]	[64]
	Hypriot OS	Container-based virtualization on Raspberry Pi	[72]	[46]
	OpenStack	OS for IaaS deployment	[14, 131]	[97]
	LXC	Containers for Linux distributions	[131]	[75]
Networking	Open vSwitch	Virtual L2 and L3 switch	[52, 58, 124]	[130]
	Click modular router	Packet processing elements for routing	[124]	[66]
	NDN	Routing over named content	[72]	[59]
	srsLTE	LTE software suite	[20, 124]	[121]
	nextEPC	3GPP-compliant EPC for LTE and 5G	[20, 51, 124]	[94]
	5G-EmPOWER	Controller for RANs	[20, 124]	[19]
	OpenAirInterface	Experimentation platform for LTE and 5G	[74]	[95]
	Athonet SGW-LBO	Connection between MEC and the network	[14]	[7]
MANO	LightMANO	MANO for edge and scattered environments	[124]	[111]
	OpenBaton MANO	ETSI-compliant MANO framework	[10, 14]	[9]

These applications are very different in terms of resource needs. In all the works, the authors claim that, despite the low cost, Raspberry Pi 3 can serve the desired applications.

6.2 Virtualization

Most MEC initiatives trust virtualization to deploy applications. For this reason, the implementations employ many tools related to virtualization. The works use both virtual machine- and container-based virtualization since both technologies present different trade-offs [43, 139].

Kubernetes is an open-source software to deploy, scale, and manage container-based virtualization. It creates a cluster from one or several hosts, enabling container instantiation and management. Kubernetes also offers automatic scaling and error recovery. The implementations of lightMEC [124], LightEdge [20], and OpenNESS [58] use Kubernetes [18] to orchestrate containers. OpenNESS enhances Kubernetes to deal with the specifics of MEC containers, such as considering latency when choosing a host to instantiate a container.

Docker is a software that provides container-based virtualization [88]. LightMEC [124] and PiCasso [72] use Docker explicitly to provide a PaaS for the applications running in their MEC hosts. Until version 1.20, Kubernetes used Docker as its container runtime engine [13]. Therefore, technically, all the projects that use older versions of Kubernetes also use Docker.

KVM is an acronym for Kernel-based Virtual Machine [64]. It can instantiate virtual machines (VMs) on a Linux environment when hardware support is present. The work from Cattaneo *et al.* uses KVM as a hypervisor, capable of instantiating VMs to the VIM [14].

Hypriot [41, 46] is an OS that enables container-based virtualization on the Raspberry Pi. The OS enables the deployment of Docker [88] applications on top of the single-board computer. PiCasso [72] uses Hypriot together with Docker to provide a PaaS for users' applications.

OpenStack is an OS to facilitate the deployment of an IaaS cloud. It can instantiate, orchestrate, and manage VMs, using pools of computing, storage, and networking resources [97]. Even though OpenStack is a popular virtualization solution, not many MEC implementations use it. Carella *et al.* argue that MEC needs a solution more lightweight than OpenStack [10]. MEC-ConPaaS [131], leaves the virtualization to LXC, but uses OpenStack to orchestrate the containers. The work from Cattaneo *et al.* uses OpenStack as the *VIM* in the MEC-NFV integration architecture [14].

LXC is an interface for Linux kernel that allows users to instantiate containers hosted by Linux OS [75]. MEC-ConPaaS [131] uses LXC together with Raspberry Pi's to offer containers to users.

6.3 Networking

Networking is an essential part of MEC systems. It is important to interconnect MEC applications with the UEs, the cloud, and other MEC applications - which can be in the same MEC host or other MEC hosts. In some cases, it is also essential to provide networking services to the MEC applications.

Open vSwitch (OVS) is a virtual switch [130], redirecting traffic between virtual machines and the outside world. LightMEC uses OVS to implement traffic rules that connect the eNodeB, the MEC applications, and the EPC [124]. OpenNESS uses OVS to their virtualization runtime, steering traffic from the containers [58]. The work from Hsie *et al.* [52] uses OVS to redirect requests from UE to applications running in MEC hosts or to send the requests to hosts in the cloud.

Click modular router is a set of packet processing elements, modular in their nature, that can be arranged in different ways to provide several routing functionalities [66]. LightMEC uses the click processing elements to offer the *MEC platform* services [124]. It also uses the processing elements to decapsulate or encapsulate traffic that comes from and to UE.

Named Data Networking (NDN) is a Information-Centric Networking implementation [59]. It identifies content on the Internet rather than hosts, as IP protocol does. For example, PiCasso uses NDN to identify services offered in the network, so users are not tied to a specific server for a given service [72].

srsLTE is a software suite for UE, eNodeB, EPC, and other entities of LTE architecture. For example, light-MEC [124] and LightEdge [20] use srsLTE to deploy a RAN and test the performance of their implementations on an LTE testbed, together with nextEPC.

nextEPC is a 3GPP-compliant EPC for the LTE and 5G. It provides the interfaces between EPC and the other entities of the LTE and 5G architecture. The implementations lightMEC [124], LightEdge [20], and P4EC [51] employ nextEPC to the EPC in their testbeds.

5G-EmPOWER is a controller for RANs [19]. It provides a series of abstractions for the RAN and makes these abstractions accessible through an API. In lightMEC, 5G-EmPOWER is responsible for interacting with the backhaul controller and deploying light virtual network functions to the edge hosts [124]. LightEdge [20] uses 5G-EmPOWER to manage the radio resources and for its Radio Network Information service [110].

OpenAirInterface is an experimentation platform for LTE and 5G [95]. It implements the RAN and the core networks. Li *et al.* use OpenAirInterface to provide a testbed for their middlebox-based MEC [74].

Athonet SGW-LBO solution for MEC is a tool to enable the MEC deployment in the SGW Local Break Out (SGW-LBO) [7]. The software can extract IP packets from the GTP tunnels and give them directly to the MEC Platform or applications. For example, the work from Cattaneo *et al.* uses Athonet SGW-LBO to connect the MEC applications and platform to the rest of the network [14].

6.4 MANO

In a certain sense, NFV represents the joint usage of networking and virtualization, contributing to integration proposals between MEC and NFV. Due to these proposals, some implementations use MANO tools to develop MEC systems.

LightMANO is a MANO for NFV deployment in scattered hosts [111]. Its design is similar to the ESTI NFV architecture and is supports basic NFV operations [40]. LightMEC employs LightMANO as its mobile edge platform orchestrator to deal with the scattered nature of MEC systems [124].

OpenBaton MANO [9] is a framework to orchestrate NFV services running on top of heterogeneous infrastructures. This feature motivated Carella *et al.* to use OpenBaton in their prototype [10]. The authors show that it is possible to use OpenBaton to manage and orchestrate resources for NFV and MEC applications. The work from Cattaneo *et al.* uses OpenBaton as an NFV orchestrator, and to manages the life cycle of the MEC platform and the MEC applications [14].

7 OPEN ISSUES

Although an evolving research field with varying commercial and business interests, MEC practical initiatives still lack enhancements that can significantly improve the user experience and the costs for MNOs. The ETSI standard mentions mobility awareness, offloading decision, and privacy as challenges that need answers [33, 36, 113]. Hereafter, we advance the ETSI statements by bringing a much more detailed discussion on five current MEC challenges. Additionally to the ETSI description, we provide discussions on implementations and tools fine-tuning challenges and commercial solutions' weaknesses. Per challenge, we enumerate open questions, discuss existing works, and highlight non-tackled requirements.

7.1 Mobility Awareness

The MEC quality of service (QoS) is highly correlated with the proximity between a UE and its serving MEC host. When a UE requests the service of a MEC host, the MNO can choose the optimal MEC host to serve the UE, starting a MEC application instance in this MEC host. Nevertheless, as a UE moves, it can move further away from the serving MEC host, reducing the QoS. Mobility can also make the UE approach other MEC hosts, changing the optimal MEC host location to the new one. When this happens, the MEC system can trigger the migration of a MEC application instance from the previous optimal MEC host to the current optimal one. In this situation, the MEC system instantiates the MEC application in the new MEC host by transferring the application context and reconfiguring the network to forward the traffic accordingly.

According to the literature, we can summarize the mobility-related open issues as:

(1) How to allocate MEC applications instances to MEC hosts on a mobility-perceptive basis, improving QoE and reducing costs?
(2) How to statically distribute MEC resources to reduce the migration overhead?
(3) What is the best instant to migrate an application?
(4) What exactly to migrate?

The different mobility patterns are essential factors to consider. For instance, it may not be optimal to perform immediate MEC application migrations in moving vehicles [69]. However, in cases where movement decisions can be planned, influenced, or intuitively anticipated, such as the case of UE-like tourists in a city, it is possible to suggest optimal itineraries, leading through paths that maximize the available MEC resources [22, 42]. Furthermore, contextual information (e.g., periods of the day, occasional special events, or still weather conditions) impacting mobility decisions of UEs can be leveraged in anticipation of high-probable movements to assist resource allocation in MEC systems adaptively.

The deployment of MEC systems can follow a static strategy, taking UE mobility into account to optimally and in advance choose the region that every MEC host should serve [47]. Otherwise, such development can anticipate or leverage UE mobility to migrate the MEC application when the current MEC host is not optimal anymore [83]. Besides, mobility affects the services that are shared among users.

It is not trivial to migrate MEC applications in real-time. First, the MEC system must migrate the application and its context. Then, it must configure the network so that UE requests are forwarded to the new MEC host. These procedures can hinder the performance of latency-sensitive applications. Therefore, the ETSI standard recommends that authors trigger the migration when applications are not in latency-sensitive periods [33]. The work of Kondo *et al.* is an interesting approach for MEC application migration. Their work acts in the network layer, making sure the MEC application instance mobility is seamless [67].

Some MEC applications can be divided into tasks. It is possible to migrate a part of the tasks, avoiding the migration of the complete application [49]. When UE moves, and this triggers a MEC application migration, it may be interesting to migrate the application partially by only migrating the tasks that mobility affects.

The survey of Rejiba *et al.* brings an interesting list of works that deal with mobility within the context of edge paradigms. Nevertheless, in the context of MEC, the works are still in very early stages. The literature on MEC lacks mobility-perceptive MEC allocation strategies. Proposing and implementing MEC systems addressing this issue is crucial for full-scale deployments, where UEs can have varying mobility degrees.

7.2 Offloading Decision

Task offloading is an important functionality of MEC systems [36]. Consider the case where it is possible to divide a UE application into tasks. The task offloading consists of executing one or more tasks of a UE application into external resourced locations (i.e., the MEC or the cloud). Offloading presents a trade-off. On the one hand, offloading an intensive task to a more resourceful device can improve the battery life of the UE and the time for task completion. On the other hand, offloading a task to some resource too far from the UE can increase the time for the task result to be available to the user, hindering QoE [49, 78]. The offloading decision is to decide whether and where to offload a certain task, and these are two separate problems [15]. ESTI standard does not define which entities should take the offloading decision. In this sense, the standard leaves for the implementation the responsibility for the offloading strategies. Therefore, four questions become important when considering offloading strategy:

(1) Which tasks should the UE offload?
(2) Given that the UE should offload a task, should it offload the task to the MEC or to the cloud?
(3) Given that the UE should offload a task to the edge, which MEC host should run the task?
(4) Which of the entities should take the offloading decision?

Different entities can take the offloading decision, holding different information that can help this decision. The UE OS is aware of the UE resources, so it knows when resources are critical. The UE application holds essential information about its intensive and latency-sensitive tasks. The MEC system knows its resources, their heterogeneity, and availability. Therefore, it is important to enforce interactions between MEC systems, UE OS', and UE applications to provide a robust offloading decision. One interesting example of dealing with these aspects is the work from Van Lingen *et al.*, proposing an architecture to unify NFV, 5G, and fog computing [132]. In their architecture, resources in the cloud or the edge are part of the same resources pool, and user applications are modeled using the data modeling language YANG [56]. An orchestrator then uses the data models to match the applications with the resources that should run them.

Offloading to the edge is similar to offloading to the cloud, but there are relevant differences. In both cases, network utilization, UE energy, and external resources availability are aspects to consider. In the specific case of edge offloading, network latency, UE mobility, and even context-awareness are imperative [136].

When deciding whether to offload a task, it is important to know the conditions of the network connecting the UE to the external resources [16]. Finally, mobility can alter the trade-off between the cost of local and external task execution, changing the offloading decision [50].

When choosing which MEC host should handle the offloading, choosing a MEC host with good network conditions for the requesting UE is crucial [15]. Nevertheless, the heterogeneity of MEC resources is also relevant to offloading decision [147]. The different resources and different network conditions between MEC hosts can present a trade-off that is not trivial to manage.

Mobility effects can also hinder computation offloading [62]. Some initiatives propose solutions to mitigate these problems. Hoang *et al.* propose an offloading decision strategy for vehicular networks [50]. Wei *et al.* study a scenario where MEC hosts are the ones moving, mounted on unmanned aerial vehicles (UAVs) [138]. They propose a method with deep reinforcement learning to optimize offloading decision. Another initiatives aim to reduce the time for handoff. Zhou *et al.* propose Comp-HO, an algorithm for faster handoff [148]. Implementations are yet to incorporate such solutions.

The ETSI MEC standard does not define which entity should perform the offloading decision. Nevertheless, it suggests that the UE or the UE application decides whether to offload a task to the MEC [32], and the MEC system decides which of the MEC hosts should execute the task [38].

The literature needs works that can properly orchestrate the tasks between UE, different MEC hosts, and cloud while adapting to UE's communication and mobility behavior and its needs.

7.3 Artificial Intelligence for and on the Edge

Artificial Intelligence (AI) can have a significant impact on MEC implementations, for two main reasons. First, AI can help the MEC system with the offloading decision [23]. Second, AI can help the MEC system with the scheduling of the tasks on a host level [60, 107]. Another important aspect of AI is that, since it can be an intensive task, the MEC system can offer an AI service for the UE, creating an AI service on the edge [23]. Among the several important questions that can be considered, we can highlight:

(1) How can MEC use AI to manage itself?
(2) How can MEC use AI to improve offloading?
(3) What are the limits of MEC to provide AI as a service?

Jiang *et al.* propose a MEC resource scheduling framework based on reinforcement learning [60]. In their framework, centralized training helps each MEC host to take its scheduling decisions in a distributed fashion. Their work shows that MEC implementations can use AI to decide on resource scheduling.

To improve the MEC offloading , Sun *et al.* develop ATOS, the Application-driven Task Offloading Strategy [125]. Their strategy uses deep reinforcement learning to perform offloading decision when tasks have dependency relations. Their goal is to optimize the costs in terms of delay, energy, and QoS. Li *et al.* follow a similar approach for vehicular networks [73]. Wei *et al.* propose a method with deep reinforcement learning to optimize offloading decision with mobile MEC hosts [138]. These solutions can form the core of the MEC applications orchestrator to allow intelligent placement of tasks among the possible MEC servers.

Finally, since the objective of a MEC infrastructure is to meet the computing needs of user applications, this also includes AI-based applications. Indeed, MEC AI-based use cases [36] introduce many challenges that need to be tackled by the community and have a direct impact on the used MEC architecture. For example, one of the challenges is how to divide the AI models between UE, MEC hosts, and cloud [98, 118, 140]. Basically, let's consider an AI-based application such as a Neural Network model. The question is then how to split and deploy this model on multiple entities (end device and MEC server) with multiple computational capabilities instead of using various models based on the capabilities of the hosting device. To answer this question, the MEC scheduling orchestrator plays an essential role in splitting the model into a subset of tasks (each one corresponds to a set of NN layers) and selecting the MECs where to deploy them.

Another critical aspect that should also be considered in future works on the MEC infrastructure design is how to cope with applications privacy requirements [91]. AI applications emphasize this need since they require access to data provided mainly by the user. A first solution is to divide the application into a task and have the task corresponding to the feature extraction performed by the user's terminal, which leads to improved privacy. Unfortunately, this solution can not hold if the user's device resources do not support the required processing or where the data comes from several sources. A second approach is to supply the MEC architecture with privacy solutions such as the use of data safes (data vaults), which are not currently supported by the architectures and implementations that we can find in the literature.

As a conclusion, the literature shows a strong interaction between MEC and AI. Nevertheless, there is still room for MEC systems implementations to use AI in practice.

7.4 Privacy Compliance

Privacy is a common concern to cloud computing since it means sending data and applications to a third-party custody [141]. MEC brings similar, but nor precisely the same concerns. We sum up MEC privacy issues as:

(1) How can different access networks (i.e., 5G, WiFi, LTE) influence users' privacy?
(2) How vulnerable is MEC when compared to the cloud?
(3) How much MEC hosts are heterogeneous when it comes to privacy vulnerability?

Since MEC is embedded into the network, it increases the value of compromising the security of the access and core networks [105]. In addition, MEC systems can operate within different networks, that can offer different security levels [146]. Moreover, the networks' users can have different privacy requirements.

The UE is usually owned and controlled by the user. Therefore, offloading a task to third-party equipment and software also incurs privacy issues [126]. Furthermore, in edge environments, hosts are geographically scattered. This distribution makes it harder to provide physical protection, leaving hardware more exposed to attackers. However, the attacks are geographically limited, limiting their possible profits as well [146]. Additionally, the security level of MEC hosts can be heterogeneous [27]. One possibility is to send sensitive tasks to a MEC host that has improved security infrastructure and is in a safer location.

Privacy is also related to mobility and offloading decision. If the MEC system leverages mobility information to predict the user trajectory, the MEC system must treat it as sensitive. From the offloading perspective, it is possible to adopt policies accordingly, adjusted to desired levels of privacy. As a result, user-sensitive data is not exchanged with MEC, while the user benefits from the MEC offload in other contexts [48].

Even though literature studies privacy for edge systems, MEC implementations still lack privacy-embedded services. For MEC adoption, MEC implementations must conform to users' privacy concerns and choices.

7.5 Networking Aspects

MEC is a service provided by the MNOs, playing the following roles: MEC needs the connectivity from the network it works within, at the same time it can provide services that improve the same network. This means that some MEC implementation's challenges come from networking challenges. In this context, some important questions to consider are:

(1) What are the limits that networks impose on MEC?
(2) How can MEC implementations improve networking services?

For the UE to benefit from the proximity of MEC hosts, it is important for the network to be capable of low latency communication. In some scenarios, this can be challenging. One important example is vehicular networks, projected to make extensive use of edge computing [76]. Vehicles move fast, therefore they perform hand-offs frequently. This hinders the proximity of the edge servers. Siyu *et al.* propose a MEC-based architecture that makes a fast hand-off [149].

MEC also shows a great potential to improve connectivity, by offering networking services on the edge. Makris *et al.* show that MEC can help 5G to optimise radio parameters, improving latency for UEs [80]. Iborra *et al.* use MEC to provide network slicing for IoT devices. General-purpose MEC implementations should take into account these sort of services [114].

We believe the MEC systems implementations should take into account the network challenges. Addressing these challenges can create significant incentives for MNOs to adopt MEC and some specific implementation.

Table 6. Examples of commercial initiatives.

Product name	**Developer**	**Scope**	**Keywords**	**Reference**
OpenNESS	Intel®	MEC software implementation	ETSI-compliant, software	[58]
Athonet Connectivity Platform	Athonet®	Private LTE or 5G with edge nodes	edge nodes, SGW-LBO	[8]
Affirmed Cloud Edge	Affirmed®	MEC software implementation	traffic steering software	[5]
MobiledgeX Edge-Cloud Platform	MobiledgeX®	Multiple edge sites management	cloud replication	[92]

7.6 Commercial Initiatives

For users and MNOs to benefit from MEC, it is essential to have commercially available MEC systems and implementations. Commercial initiatives must enable MNOs to deploy MEC systems, but it is also important that MNOs have access to edge-capable infrastructure. Moreover, a product might not deploy a complete MEC system but a tool or a building block for a MEC system. Table 6 summarizes the commercial initiatives discussed hereafter. The commercial initiatives raise some questions:

(1) Is it possible to find real off-the-shelf initiatives?
(2) Which commercial initiatives are compatible with the ETSI standard?
(3) How complete are the commercial initiatives?

One of the most promising commercial products is OpenNESS, a software for deploying MEC systems. We discuss the open version of OpenNESS in Section 4, but it also has a commercial distribution, licensed by Intel® [58]. It follows the same principles and architecture as its open version, except it is optimized to run on Intel® hardware. Just like its open counterpart, OpenNESS sticks to the ETSI standard.

Athonet has a product suite for MEC development [8]. They provide a private LTE or 5G network, with the possibility of edge nodes. Even though their product is not a MEC system, they claim that it is possible to easily build an ETSI MEC system on top of their suite. Athonet also has a tool, mentioned in Section 6, to allow the SGW-LBO deployment of MEC [7].

Affirmed has a MEC implementation that does not follow the ETSI architecture. Their MEC sits in the S1 interface and offers two main functionalities [5]. The first, similar to ETSI MEC, is to host MEC applications. The second is to steer traffic so that critical applications can bypass certain functions from the core network. It is possible to implement a traffic steering functionality as a MEC application in the ETSI MEC, but it is native to the Affirmed MEC implementation.

MobiledgeX is a solution to manage multiple edge sites and facilitate the deployment of applications [92]. MobiledgeX creates the applications in the cloud and migrates them to the edge, so they benefit from low latency. MobiledgeX also has a software development kit that allows applications in the UE to discover and use edge resources when they are available. However, MobiledgeX is not compatible with the ETSI standard. Additionally, MobiledgeX is not immediately suitable for MEC applications because it can instantiate in the edge applications copied from the cloud, but not stand-alone MEC applications.

The initiatives mentioned above show that the maturity level of MEC implementations is starting to reach the shelves. They also show that there is room in the commercial distributions for the development of MEC applications and UE applications that are MEC-aware. Nevertheless, it is not yet possible to find ETSI MEC solutions capable of deploying a MEC system from the infrastructure to its software.

7.7 Implementations and Tools Fine-Tuning

MEC implementations are not simply extensions of the cloud. Their distribution, heterogeneity, and ecosystem are significantly different. For this reason, tools meant to deploy clouds are not always adequate for deploying MEC systems. We believe it is important to answer these questions:

(1) How do software and hardware tools relate to ETSI and non-ETSI architectures?
(2) How tools currently available need to change so they provide better building blocks for MEC?
(3) Is it possible to have tools that are robust enough to the (most intensive) MEC use case but light enough (to the most constrained use case)?

The building blocks in software and hardware available for authors are not necessarily designed as the entities in the ETSI architecture [20, 58, 124, 145]. Non-ETSI initiatives also extrapolate virtualization tools to work in the edge [10, 14]. In both cases, implementations could benefit from edge-specific tools. The case of OpenNESS shows that one can optimize the cloud orchestrator Kubernetes to the edge specifics [58].

Some tools come close to fulfill the requirements of a MEC system but have a few problems. For instance, OpenStack is a very popular tool for cloud development, but it is suited for a federated environment, leading to significant overhead [70]. Kubernetes works with container orchestration, but it is not entirely adequate for the edge environment. Kubernetes has no native support for mapping between applications and MEC hosts, forcing its own MEC host orchestration policies [81].

In our survey, we find implementations that use virtualization tools with level-varying robustness, as we show in Table 5. It also shows some implementations using the lightweight Raspberry as main hardware, meaning that the optimal trade-off between robustness and overhead is not yet achieved [72, 84, 131].

Serverless computing is a promising technology. With it, application developers can execute functions in the cloud and edge without explicitly allocating the resources for it [142]. This means that the MEC infrastructure should be able to allocate the resources for function execution, a much more fine-grained operation than the traditional virtual machine or container virtualization. The literature gives special attention to the case of serverless computing to the edge for IoT [12, 65]. It is possible to find tools for serverless computing on the edge, such as OpenWhisk [103]. Nevertheless, it is still necessary to integrate serverless computing into the MEC implementations.

Most of available tools are aimed at cloud implementations. The success of MEC initiatives depends on relying on tools that have their costs and benefits adjusted to the MEC requirements. Furthermore, the MEC network-awareness enables several improvements. MEC initiatives can propose better solutions by supporting mobility, optimizing offloading decisions, preserving privacy, and taking the best available tools.

8 CONCLUSION AND FINAL REMARKS

MEC can play an important role in offering resources to the edge of the network. Users, application developers, and MNOs can benefit from this arrangement. For this reason, the scientific community has been making an effort so that MEC can achieve its full potential. One important step in this effort is to provide prototypes, proofs-of-concept, and implementations for the technology. The implementations can bring light into issues that are not so clear when discussing theoretical models - or even bring new, unexpected questions.

In this survey, we searched the literature for MEC initiatives with practical implementations and discussed their highlights. We examined their definition for MEC, their broad vision, and their stance regarding the ETSI MEC standard. We discuss the strategies and the tools authors used to develop these MEC implementations. Finally, we pinpointed the open issues related to MEC implementation. Table 7 summarizes the main remarks from the works we found. The column *Name* indicates the name of the implementation, the column *ETSI-Compliant* indicates whether the implementation follows the ETSI MEC standard, the column *Mobility Aware* indicates if the implementation takes UE mobility into account, the column *Offloading Support* indicates whether the

Table 7. MEC practical initiatives final remarks.

Name	**ETSI-Compliant**	**Mobility Aware**	**Offloading Support**	**Privacy Support**	**Commercial**	**Fine Tuning**	**MNO-Centric**	**Reference**
LightMEC	✓					✓	✓	[124]
LightEdge	✓					✓	✓	[20]
M^2EC	✓					✓	✓	[145]
MEC-NFV with LBO	✓						✓	[14]
OpenNESS	✓				✓	✓	✓	[21, 58]
MANO+							✓	[117]
NFV-based MEC with Open Baton								[10]
MEC-ConPaaS								[131]
Container-Based MEC for IoT							✓	[52, 53]
PiCasso								[72]
MEC as a middlebox for LTE							✓	[74]
P4EC							✓	[51]
ACACIA			✓					[17]
eRAM			✓					[84]

implementation has native support to offloading, the *Privacy Support* column indicates if the implementation has built-in privacy mechanisms, the *Commercial* column indicates whether the implementation has a commercial distribution, the *Fine Tuning* column indicates whether the implementation makes an effort to adjust existing tools to MEC environment, the column *MNO-Centric* indicates if the implementation is designed for the MNOs requirements, finally the column *Reference* points to a reference to the implementation.

It is possible to conclude from Table 7 there is much space for improvement regarding MEC implementations. This improvement can come in the form of new versions of existing implementations or even new implementations. Some factors contributing to the slow down of practical MEC solutions are the hardness of their implementations, the absence of tailored MEC systems providing high-level programming and modular software, the lack of flexibility and integration of solutions, and the deficiency of extensive deployment of advanced cellular networks (e.g., 5G), providing low latency communication capabilities. On the other side, MEC research is fast evolving, and its deployment benefits are increasingly attracting commercial interests. In this context, we believe in a future with large MEC deployment supporting what UEs and applications really need and ask for. We hope this survey on existing MEC initiatives will provide a starting point for researchers and developers to build their own MEC systems and validate their solutions. We also wish the significant limitations highlighted in our review can help to improve and optimize the existing initiatives.

REFERENCES

[1] 3GPP. 2014. *General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access.* Technical Report 3GPP 123.401. 3GPP, France.

[2] 3GPP. 2018. *3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C); Stage 3.* Technical Report 3GPP 29.274. 3GPP, France.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie. 2018. Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal* 5, 1 (2018), 450–465. https://doi.org/10.1109/JIOT.2017.2750180

[4] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–35.

[5] Affirmed. 2020. affirmed. <https://www.affirmednetworks.com/products-solutions/mec-solutions/>. Accessed on 29 Jan., 2021.

[6] Yuan Ai, Mugen Peng, and Kecheng Zhang. 2018. Edge computing technologies for Internet of Things: a primer. *Digital Communications and Networks* 4, 2 (2018), 77–86.

[7] Athonet. 2018. SGW-LBO solution for MEC Taking services to the Edge. <http://telecoms.com/wp-content/blogs.dir/1/files/2018/04/MEC_SGWLBO_WP_MWC_2018.pdf>. Accessed on 8 Feb., 2021.

[8] Athonet srl. 2021. EdgeInfra. <https://www.athonet.com/architecture/>. Accessed on 06 Jul., 2021.

[9] Giuseppe Antonio Carella and Thomas Magedanz. 2015. Open baton: a framework for virtual network function management and orchestration for emerging software-based 5g networks. *IEEE Newsletter* (2015).

[10] Giuseppe A Carella, Michael Pauls, Thomas Magedanz, Marco Cilloni, Paolo Bellavista, and Luca Foschini. 2017. Prototyping nfv-based multi-access edge computing in 5G ready networks with open baton. In *2017 IEEE Conf. on Network Softwarization (NetSoft)*. IEEE, 1–4.

[11] Brian Carpenter and Scott Brim. 2002. *Middleboxes: Taxonomy and issues*. Technical Report. RFC 3234, February.

[12] Gustavo André Setti Cassel, Vinicius Facco Rodrigues, Rodrigo da Rosa Righi, Marta Rosecler Bez, Andressa Cruz Nepomuceno, and Cristiano André da Costa. 2022. Serverless computing for Internet of Things: A systematic literature review. *Future Generation Computer Systems* 128 (2022), 299–316.

[13] Jorge Castro, Duffie Cooley, Kat Cosgrove, Justin Garrison, Noah Kantrowitz, Bob Killen, Rey Lejano, Dan "POP" Papandrea, Jeffrey Sica, and Davanum "Dims" Srinivas. 2020. Don't Panic: Kubernetes and Docker. <https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/>. Accessed on 19 Jan., 2021.

[14] Giorgio Cattaneo, Fabio Giust, Claudio Meani, Daniele Munaretto, and Pietro Paglierani. 2018. Deploying cpu-intensive applications on mec in nfv systems: The immersive video use case. *Computers* 7, 4 (2018), 55.

[15] Min Chen and Yixue Hao. 2018. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications* 36, 3 (2018), 587–597.

[16] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2015. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking* 24, 5 (2015), 2795–2808.

[17] Junguk Cho, Karthikeyan Sundaresan, Rajesh Mahindra, Jacobus Van der Merwe, and Sampath Rangarajan. 2016. ACACIA: context-aware edge computing for continuous interactive applications over mobile networks. In *12th Int. on Conf. on emerging Networking EXperiments and Technologies*. 375–389.

[18] CNCF. 2019. Kubernetes. <https://kubernetes.io>. Accessed on 7 Dec., 2020.

[19] Estefanía Coronado, Shah Nawaz Khan, and Roberto Riggio. 2019. 5G-EmPOWER: A software-defined networking platform for 5G radio access networks. *IEEE Transactions on Network and Service Management* 16, 2 (2019), 715–728.

[20] Estefania Coronado, Zarrar Yousaf, and Roberto Riggio. 2020. LightEdge: Mapping the Evolution of Multi-Access Edge Computing in Cellular Networks. *IEEE Communications Magazine* 58, 4 (2020), 24–30.

[21] Intel Corporation. 2019. Open Network Edge Services Software. <https://www.openness.org>. Accessed on 7 Dec., 2020.

[22] Sand L Correa, Kleber V Cardoso, Felipe F Fonseca, Lefteris Mamatas, and Aline C Viana. 2021. *Itinerary Recommendation Algorithm in the Age of MEC*. Technical Report hal-03147515. Inria Saclay Ile de France, France.

[23] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. 2020. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal* 7, 8 (2020), 7457–7469.

[24] Koustabh Dolui and Soumya Kanti Datta. 2017. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 1–6.

[25] Jack J Dongarra, Cleve Barry Moler, James R Bunch, and Gilbert W Stewart. 1979. *LINPACK users' guide*. SIAM.

[26] Praveen Kumar Donta, Satish Narayana Srirama, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. 2021. Survey on recent advances in IoT application layer protocols and machine learning scope for research directions. *Digital Communications and Networks* (2021).

[27] Miao Du, Kun Wang, Yuanfang Chen, Xiaoyan Wang, and Yanfei Sun. 2018. Big data privacy preserving in multi-access edge computing for heterogeneous Internet of Things. *IEEE Communications Magazine* 56, 8 (2018), 62–67.

[28] Mohammed S Elbamby, Cristina Perfecto, Mehdi Bennis, and Klaus Doppler. 2018. Toward low-latency and ultra-reliable virtual reality. *IEEE Network* 32, 2 (2018), 78–84.

[29] ETSI. 2013. *Network Functions Virtualisation (NFV); Use Cases*. Technical Report ETSI GS NFV 001 V1.1.1. European Telecommunications Standards Institute, France.

[30] ETSI. 2014. *Network Functions Virtualisation (NFV); Architectural Framework*. Technical Report ETSI GS NFV 002 V1.2.1. European Telecommunications Standards Institute, France.

[31] ETSI. 2016. *Mobile Edge Computing (MEC); Terminology*. Technical Report ETSI GS MEC 001 V1.1.1. European Telecommunications Standards Institute, France.

[32] ETSI. 2016. *Multi-Access Edge Computing (MEC); Technical Requirements.* Technical Report ETSI GS MEC 002 V1.1.1. European Telecommunications Standards Institute, France.
[33] ETSI. 2017. *Multi-Access Edge Computing (MEC); End to End Mobility Aspects.* Technical Report ETSI GR MEC 018 V1.1.1. European Telecommunications Standards Institute, France.
[34] ETSI. 2018. *5G;Management and orchestration; 5G Network Resource Model (NRM); Stage 1.* Technical Report 3GPP TS 28.540 version 15.0.0 Release 15. 3GPP, France.
[35] ETSI. 2018. *Multi-Access Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV environment.* Technical Report ETSI GS MEC 017 V1.1.1. European Telecommunications Standards Institute, France.
[36] ETSI. 2018. *Multi-Access Edge Computing (MEC); Phase 2: Use Cases and Requirements.* Technical Report ETSI GS MEC 002 V2.1.1. European Telecommunications Standards Institute, France.
[37] ETSI. 2019. *Digital cellular communications system (Phase 2+) (GSM); Universal Mobile Telecommunication System (UMTS); LTE; 5G;.* Technical Report 3GPP TR 21.915 version 15.0.0 Release 15. 3GPP, France.
[38] ETSI. 2019. *Multi-Access Edge Computing (MEC); Framework and Reference Architecture.* Technical Report ETSI GS MEC 003 V2.1.1. European Telecommunications Standards Institute, France.
[39] ETSI. 2021. *Multi-access Edge Computing (MEC); Study on Inter-MEC systems and MEC-Cloud systems coordination.* Technical Report ETSI GR MEC 035 V3.1.1. European Telecommunications Standards Institute, France.
[40] ETSI. 2021. *Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Architectural Framework Specification.* Technical Report ETSI GS NFV 006 V2.1.1. European Telecommunications Standards Institute, France.
[41] Govinda Fichtner, Dieter Reuter, Stefan Scherer, Mathias Renner, and Andreas Eiermann. 2020. HypriotOS. <https://github.com/hypriot>. Accessed on 21 Dec., 2020.
[42] Felipe F Fonseca, Lefteris Mamatas, Aline C Viana, Sand L Correa, and Kleber V Cardoso. 2019. Personalized Travel Itineraries with Multi-access Edge Computing Touristic Services. In *2019 IEEE Global Communications Conf. (GLOBECOM)*. IEEE, 1–6.
[43] Khasa Gillani and Jong-Hyouk Lee. 2020. Comparison of Linux virtual machines and containers for a service migration in 5G multi-access edge computing. *ICT Express* 6, 1 (2020), 1–2.
[44] Fabio Giust, Gianluca Verin, Kiril Antevski, Joey Chou, Yonggang Fang, Walter Featherstone, Francisco Fontes, Danny Frydman, Alice Li, Antonio Manzalini, et al. 2018. MEC deployments in 4G and evolution towards 5G. *ETSI White paper* 24, 2018 (2018), 1–24.
[45] Fabio Giust, Gianluca Verin, Kiril Antevski, Joey Chou, Yonggang Fang, Walter Featherstone, Francisco Fontes, Danny Frydman, Alice Li, Antonio Manzalini, Debashish Purkayastha, Dario Sabella, Christof Wehner, Kuo-Wei Wen, and Zheng Zhou. 2018. *MEC Deployments in 4G and Evolution Towards 5G.* Technical Report ETSI White Paper No. 24. European Telecommunications Standards Institute, France.
[46] Marcel Großmann, Andreas Eiermann, and Mathias Renner. 2016. Hypriot cluster lab: an arm-powered cloud solution utilizing docker. In *23rd Int. Conf. on Telecommunications (ICT 2016), May 16 - May 18, Thessaloniki.* OPUS.
[47] Xinjie Guan, Xili Wan, Jun-bo Wang, Xinxin Ma, and Guangwei Bai. 2018. Mobility aware partition of MEC regions in wireless metropolitan area networks. In *IEEE INFOCOM 2018-IEEE Conf. on Computer Communications WKSHs (INFOCOM WKSHPS)*. IEEE, 1–2.
[48] Xiaofan He, Juan Liu, Richeng Jin, and Huaiyu Dai. 2017. Privacy-aware offloading in mobile-edge computing. In *GLOBECOM 2017-2017 IEEE Global Communications Conf.* IEEE, 1–6.
[49] Xin He, Mingyue Meng, Shuang Ding, and Hao Li. 2021. A Survey of Task Migration Strategies in Mobile Edge Computing. In *IEEE 6th Int. Conf. on Cloud Computing and Big Data Analytics (ICCCBDA)*. IEEE, 400–405.
[50] Vu Huy Hoang, Tai Manh Ho, and Long Bao Le. 2019. Mobility-aware computation offloading in MEC-based vehicular wireless networks. *IEEE Communications Letters* 24, 2 (2019), 466–469.
[51] Max Hollingsworth, Jinsung Lee, Zhang Liu, Jihoon Lee, Sangtae Ha, and Dirk Grunwald. 2020. P4EC: Enabling Terabit Edge Computing in Enterprise 4G LTE. In *USENIX WKSH on Hot Topics in Edge Computing (HotEdge 20)*.
[52] Han-Chuan Hsieh, Jiann-Liang Chen, and Abderrahim Benslimane. 2018. 5G virtualized multi-access edge computing platform for IoT applications. *Journal of Network and Computer Applications* 115 (2018), 94–102.
[53] Han-Chuan Hsieh, Ching-Shiang Lee, and Jiann-Liang Chen. 2018. Mobile edge computing platform with container-based virtualization technology for IoT applications. *Wireless Personal Communications* 102, 1 (2018), 527–542.
[54] Fei Hu, Qi Hao, and Ke Bao. 2014. A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys & Tutorials* 16, 4 (2014), 2181–2206.
[55] Tao Huang, F Richard Yu, Chen Zhang, Jiang Liu, Jiao Zhang, and Yunjie Liu. 2016. A survey on large-scale software defined networking (SDN) testbeds: Approaches and challenges. *IEEE Communications Surveys & Tutorials* 19, 2 (2016), 891–917.
[56] IETF. 2010. *YANG — A Data Modeling Language for the Network Configuration Protocol (NETCONF).* Technical Report IETF RFC 6020. Internet Engineering Task Force, https://tools.ietf.org/html/rfc6020.
[57] Intel. 2021. OpenNESS Edge Applications. <https://github.com/open-ness/edgeapps>. Accessed on 1st February, 2021.
[58] Intel Corporation. 2019. Openness: enabling services and applications on the network and on-premise edge. <https://www.openness.org/templates/openness/images/the-network-and-onpremise-edge-new.pdf>. Accessed on 14 Dec., 2020.

[59] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. 2009. Networking named content. In *5th Int. Conf. on Emerging networking experiments and technologies*. 1–12.

[60] Feibo Jiang, Li Dong, Kezhi Wang, Kun Yang, and Cunhua Pan. 2021. Distributed Resource Scheduling for Large-Scale MEC Systems: A Multi-Agent Ensemble Deep Reinforcement Learning with Imitation Acceleration. *IEEE Internet of Things Journal* (2021).

[61] Sami Kekki, Walter Featherstone, Yonggang Fang, Pekka Kuure, Alice Li, Anurag Ranjan, Debashish Purkayastha, Feng Jiangping, Danny Frydman, Gianluca Verin, et al. 2018. MEC in 5G networks. *ETSI white paper* 28 (2018).

[62] Sadia Khizar, Marcelo Dias de Amorim, and Vania Conan. 2021. Offloading computing tasks beyond the edge: A data-driven analysis. In *2021 13th IFIP Wireless and Mobile Networking Conf. (WMNC)*. IEEE, 79–83.

[63] Kirschnick. 2018. Finding Faces. <https://github.com/jkirsch/senser>. Accessed on 18 Jan., 2021.

[64] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. 2007. kvm: the Linux virtual machine monitor. In *Proceedings of the Linux symposium*, Vol. 1. Dttawa, Dntorio, Canada, 225–230.

[65] Vojdan Kjorveziroski, S Filiposka, and V Trajkovic. 2021. IoT Serverless Computing at the Edge: Open Issues and Research Direction. *Computers* 10 (2021), 130.

[66] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. 2000. The Click modular router. *ACM Transactions on Computer Systems (TOCS)* 18, 3 (2000), 263–297.

[67] Tohru Kondo, Keita Isawaki, and Kaori Maeda. 2018. Development and Evaluation of the MEC Platform Supporting the Edge Instance Mobility. In *2018 IEEE 42nd Annual Computer Software and Applications Conf. (COMPSAC)*, Vol. 02. 193–198. https://doi.org/10.1109/COMPSAC.2018.10228

[68] Jarkko Kovala. 2020. *Edge computing platforms for Internet of Things*. Master's thesis. University of Helsinki, Helsinki, Finland.

[69] Ibtissam Labriji, Francesca Meneghello, Davide Cecchinato, Stefania Sesia, Eric Perraud, Emilio Calvanese Strinati, and Michele Rossi. 2021. Mobility Aware and Dynamic Migration of MEC Services for the Internet of Vehicles. *IEEE Transactions on Network and Service Management* 18, 1 (2021), 570–584. https://doi.org/10.1109/TNSM.2021.3052808

[70] Adrien Lebre, Jonathan Pastor, Anthony Simonet, and Frédéric Desprez. 2017. Revising openstack to operate fog/edge computing infrastructures. In *2017 IEEE Int. Conf. on cloud engineering (IC2E)*. IEEE, 138–148.

[71] Lucas Lersch, Xianpeng Hao, Ismail Oukid, Tianzheng Wang, and Thomas Willhalm. 2019. Evaluating Persistent Memory Range Indexes. *PVLDB* 13, 4, 574–587.

[72] Adisorn Lertsinsrubtavee, Mennan Selimi, Arjuna Sathiaseelan, Llorenç Cerdà-Alabern, Leandro Navarro, and Jon Crowcroft. 2018. Information-centric multi-access edge computing platform for community mesh networks. In *1st ACM SIGCAS Conf. on Computing and Sustainable Societies*. 1–12.

[73] Chao Li, Junjuan Xia, Fagui Liu, Dong Li, Lisheng Fan, George K Karagiannidis, and Arumugam Nallanathan. 2021. Dynamic offloading for multiuser muti-CAP MEC networks: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology* 70, 3 (2021), 2922–2927.

[74] Chi-Yu Li, Hsueh-Yang Liu, Po-Hao Huang, Hsu-Tung Chien, Guan-Hua Tu, Pei-Yuan Hong, and Ying-Dar Lin. 2018. Mobile edge computing platform deployment in 4G {LTE} networks: A middlebox approach. In *{USENIX} WKSH on Hot Topics in Edge Computing (HotEdge 18)*.

[75] Linux Containers. [n.d.]. LXC. <https://linuxcontainers.org/lxc/introduction/>. Accessed on 15 Dec., 2020.

[76] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. 2021. Vehicular edge computing and networking: A survey. *Mobile networks and applications* 26, 3 (2021), 1145–1168.

[77] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. 2019. Edge computing for autonomous driving: Opportunities and challenges. *Proc. IEEE* 107, 8 (2019), 1697–1716.

[78] Pavel Mach and Zdenek Becvar. 2017. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1628–1656.

[79] Halgurd S Maghdid, Ihsan Alshahib Lami, Kayhan Zrar Ghafoor, and Jaime Lloret. 2016. Seamless outdoors-indoors localization solutions on smartphones: Implementation and challenges. *ACM Computing Surveys (CSUR)* 48, 4 (2016), 1–34.

[80] Nikos Makris, Virgilios Passas, Christos Nanis, and Thanasis Korakis. 2019. On minimizing service access latency: Employing MEC on the fronthaul of heterogeneous 5G architectures. In *2019 IEEE Int. Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 1–6.

[81] Karim Manaouil and Adrien Lebre. 2020. *Kubernetes and the Edge?* Technical Report RR-9370. Inria Rennes-Bretagne Atlantique, France.

[82] Steve Mann, Tom Furness, Yu Yuan, Jay Iorio, and Zixin Wang. 2018. All reality: Virtual, augmented, mixed (x), mediated (x, y), and multimediated reality. *arXiv preprint arXiv:1804.08386* (2018).

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

[84] Houssemeddine Mazouzi. [n.d.]. eRAM. <https://github.com/ERAM-Project>. Accessed on 5 Jan., 2020.

[85] Houssemeddine Mazouzi, Nadjib Achir, and Khaled Boussetta. 2018. Maximizing mobiles energy saving through tasks optimal offloading placement in two-tier cloud. In *21st ACM Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 137–145.

[86] Houssemeddine Mazouzi, Khaled Boussetta, and Nadjib Achir. 2019. Maximizing mobiles energy saving through tasks optimal offloading placement in two-tier cloud: A theoretical and an experimental study. *Computer Communications* 144 (2019), 132–148.

[87] Dianne SV Medeiros, Helio N Cunha Neto, Martin Andreoni Lopez, Luiz Claudio S Magalhães, Natalia C Fernandes, Alex B Vieira, Edelberto F Silva, and Diogo MF Mattos. 2020. A survey on data analysis on large-Scale wireless networks: online stream processing, trends, and challenges. *Journal of Internet Services and Applications* 11, 1 (2020), 1–48.

[88] Dirk Merkel. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux journal* 2014, 239 (2014), 2.

[89] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2015. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications surveys & tutorials* 18, 1 (2015), 236–262.

[90] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2015. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications surveys & tutorials* 18, 1 (2015), 236–262.

[91] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. 2020. Shredder: Learning noise distributions to protect inference privacy. In *25th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*. 3–18.

[92] MobiledgeX Inc. 2021. MobiledgeX Inc. <https://mobiledgex.com/product/architecture>. Accessed on 29 Jan., 2021.

[93] Iain Morris. 2016. ETSI Drops Mobile From MEC. <https://www.lightreading.com/mobile/mec-(mobile-edge-computing)/etsi-drops-mobile-from-mec/d/d-id/726273>. Accessed on 7 Dec., 2020.

[94] NextEPC Inc. [n.d.]. NextEPC. <https://nextepc.com>. Accessed on 7 Dec., 2020.

[95] Navid Nikaein, Mahesh K Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. 2014. OpenAirInterface: A flexible platform for 5G research. *ACM SIGCOMM Computer Communication Review* 44, 5 (2014), 33–38.

[96] Huansheng Ning, Yunfei Li, Feifei Shi, and Laurence T Yang. 2020. Heterogeneous edge computing open platforms and tools for internet of things. *Future Generation Computer Systems* 106 (2020), 67–76.

[97] OpenStack Infrastructure Foundation. 2020. OpenStack. <https://www.openstack.org>. Accessed on 18 Jan., 2021.

[98] Roberto G Pacheco, Rodrigo S Couto, and Osvaldo Simeone. 2020. Calibration-Aided Edge Inference Offloading via Adaptive Model Partitioning of Deep Neural Networks. In *IEEE Int. Conf. on Communications*. IEEE.

[99] Tapti Palit, Yongming Shen, and Michael Ferdman. 2016. Demystifying cloud benchmarking. In *2016 IEEE Int. symposium on performance analysis of systems and software (ISPASS)*. IEEE, 122–132.

[100] Al-Mukaddim Khan Pathan and Rajkumar Buyya. 2007. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report* 4 (2007), 70.

[101] Quoc-Viet Pham, Fang Fang, Vu Nguyen Ha, Md Jalil Piran, Mai Le, Long Bao Le, Won-Joo Hwang, and Zhiguo Ding. 2020. A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art. *IEEE Access* 8 (2020), 116974–117017.

[102] Pawani Porambage, Jude Okwuibe, Madhusanka Liyanage, Mika Ylianttila, and Tarik Taleb. 2018. Survey on multi-access edge computing for internet of things realization. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 2961–2991.

[103] Sebastián Quevedo, Freddy Merchán, Rafael Rivadeneira, and Federico X Dominguez. 2019. Evaluating apache openwhisk-faas. In *2019 IEEE 4th Ecuador Technical Chapters Meeting (ETCM)*. IEEE, 1–5.

[104] Pasika Ranaweera, Anca Jurcut, and Madhusanka Liyanage. 2021. Mec-enabled 5g use cases: A survey on security vulnerabilities and countermeasures. *ACM Computing Surveys (CSUR)* 54, 9 (2021), 1–37.

[105] Pasika Ranaweera, Anca Delia Jurcut, and Madhusanka Liyanage. 2021. Survey on multi-access edge computing security and privacy. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 1078–1124.

[106] Raspberry. 2020. Raspberry Pi 3. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Accessed on 21 Dec., 2020.

[107] Thomas Rausch and Schahram Dustdar. 2019. Edge intelligence: The convergence of humans, things, and ai. In *2019 IEEE Int. Conf. on Cloud Engineering (IC2E)*. IEEE, 86–96.

[108] Zeineb Rejiba, Xavier Masip-Bruin, and Eva Marín-Tordera. 2019. A survey on mobility-induced service migration in the fog, edge, and related computing paradigms. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–33.

[109] Ju Ren, Deyu Zhang, Shiwen He, Yaoxue Zhang, and Tao Li. 2019. A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet. *ACM Computing Surveys (CSUR)* 52, 6 (2019), 1–36.

[110] Roberto Riggio. 2020. 5G-EmPOWER. <https://5g-empower.io>. Accessed on 19 Jan., 2021.

[111] Roberto Riggio, Shah Nawaz Khan, Tejas Subramanya, Imen Grida Ben Yahia, and Diego Lopez. 2018. LightMANO: Converging NFV and SDN at the Edges of the Network. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.

[112] Elisa Rojas, Roberto Doriguzzi-Corin, Sergio Tamurejo, Andres Beato, Arne Schwabe, Kevin Phemius, and Carmen Guerrero. 2018. Are we ready to drive software-defined networks? A comprehensive survey on management tools and techniques. *ACM Computing Surveys (CSUR)* 51, 2 (2018), 1–35.

[113] Dario Sabella, Alex Reznik, Kamal Ranjan Nayak, Diego Lopez, Fei Li, Ulrich Kleber, Alex Leadbeater, Kishen Maloor, Sheeba Backia Mary Baskaran, Luca Cominardi, Cristina Costa, Fabrizio Granelli, Vangelis Gazis, Francois Ennesser, and Xi Gu. 2021. *MEC security: Status of standards support and future evolutions.* Technical Report ETSI White Paper No. 46. European Telecommunications Standards Institute, France.

[114] Ramon Sanchez-Iborra, Stefan Covaci, Jose Santa, Jesus Sanchez-Gomez, Jorge Gallego-Madrid, and Antonio F Skarmeta. 2019. MEC-assisted end-to-end 5G-slicing for IoT. In *2019 IEEE Global Communications Conf. (GLOBECOM)*. IEEE, 1–6.

[115] David Espinel Sarmiento, Adrien Lebre, Lucas Nussbaum, and Abdelhadi Chari. 2021. Decentralized SDN Control Plane for a Distributed Cloud-Edge Infrastructure: A Survey. *IEEE Communications Surveys & Tutorials* (2021).

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

[117] Vincenzo Sciancalepore, Fabio Giust, Konstantinos Samdanis, and Zarrar Yousaf. 2016. A double-tier MEC-NFV architecture: Design and optimisation. In *2016 IEEE Conf. on standards for communications and networking (CSCN)*. IEEE, 1–6.

[118] Wassim Seifeddine, Cedric Adjih, and Nadjib Achir. 2021. Dynamic Hierarchical Neural Network Offloading in IoT Edge Networks. In *2021 10th IFIP Int. Conf. on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN)*. 1–6. https://doi.org/10.23919/PEMWN53042.2021.9664700

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

[120] Yushan Siriwardhana, Pawani Porambage, Madhusanka Liyanage, and Mika Ylianttila. 2021. A Survey on Mobile Augmented Reality With 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 1160–1192.

[121] Software Radio Systems. [n.d.]. srsLTE. <https://www.srslte.com>. Accessed on 18 Jan., 2021.

[122] Francesco Spinelli and Vincenzo Mancuso. 2020. Towards enabled industrial verticals in 5G: a survey on MEC-based approaches to provisioning and flexibility. *IEEE Communications Surveys & Tutorials* (2020).

[123] Suyash Srijan. 2014. CPUBENCH. <https://github.com/theblixguy/CPUBench>. Accessed on 18 Jan., 2021.

[124] Tejas Subramanya, Giovanni Baggio, and Roberto Riggio. 2018. lightmec: A vendor-agnostic platform for multi-access edge computing. In *2018 14th Int. Conf. on Network and Service Management (CNSM)*. ieee, 198–204.

[125] Ming Sun, Tie Bao, Dan Xie, Hengyi Lv, and Guoliang Si. 2021. Towards Application-Driven Task Offloading in Edge Computing Based on Deep Reinforcement Learning. *Micromachines* 12, 9 (2021), 1011.

[126] Hassan Takabi, James BD Joshi, and Gail-Joon Ahn. 2010. Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy* 8, 6 (2010), 24–31.

[127] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. 2017. On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1657–1681.

[128] The Apache Software Foundation. 2020. Apache Flink®. <https://flink.apache.org>. Accessed on 18 Jan., 2021.

[129] The Apache Software Foundation. 2020. ApacheBench. <https://httpd.apache.org/docs/2.4/programs/ab.html>. Accessed on 18 Jan., 2021.

[130] The Linux Foundation. 2016. Open vSwitch. <https://www.openvswitch.org>. Accessed on 18 Jan., 2021.

[131] Alexandre Van Kempen, Teodor Crivat, Benjamin Trubert, Debaditya Roy, and Guillaume Pierre. 2017. MEC-ConPaaS: An experimental single-board based mobile edge cloud. In *2017 5th IEEE Int. Conf. on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 17–24.

[132] Frank Van Lingen, Marcelo Yannuzzi, Anuj Jain, Rik Irons-Mclean, Oriol Lluch, David Carrera, Juan Luis Perez, Alberto Gutierrez, Diego Montero, Josep Marti, et al. 2017. The unavoidable convergence of NFV, 5G, and fog: a model-driven approach to bridge cloud and edge. *IEEE Communications Magazine* 55, 8 (2017), 28–35.

[133] Luis M Vaquero and Luis Rodero-Merino. 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review* 44, 5 (2014), 27–32.

[134] Pier Luigi Ventre, Stefano Salsano, Marco Polverini, Antonio Cianfrani, Ahmed Abdelsalam, Clarence Filsfils, Pablo Camarillo, and Francois Clad. 2020. Segment routing: A comprehensive survey of research activities, standardization efforts and implementation results. *IEEE Communications Surveys & Tutorials* (2020).

[135] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. 2012. Cloudlets: Bringing the cloud to the mobile user. In *3rd ACM WKSH on Mobile cloud computing and services*. 29–36.

[136] Jianyu Wang, Jianli Pan, Flavio Esposito, Prasad Calyam, Zhicheng Yang, and Prasant Mohapatra. 2019. Edge cloud offloading algorithms: Issues, methods, and perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–23.

[137] Shangguang Wang, Jinliang Xu, Ning Zhang, and Yujiong Liu. 2018. A survey on service migration in mobile edge computing. *IEEE Access* 6 (2018), 23511–23528.

[138] Dawei Wei, Jianfeng Ma, Linbo Luo, Yunbo Wang, Lei He, and Xinghua Li. 2021. Computation offloading over multi-UAV MEC network: A distributed deep reinforcement learning approach. *Computer Networks* 199 (2021), 108439.

[139] Miguel G Xavier, Marcelo V Neves, Fabio D Rossi, Tiago C Ferreto, Timoteo Lange, and Cesar AF De Rose. 2013. Performance evaluation of container-based virtualization for high performance computing environments. In *2013 21st Euromicro Int. Conf. on Parallel, Distributed, and Network-Based Processing*. IEEE, 233–240.

[140] Liyao Xiang, Jingbo Yang, and Baochun Li. 2019. Differentially-private deep learning from an optimization perspective. In *IEEE Int. Conf. on Computer Communications*. IEEE, 559–567.

[141] Zhifeng Xiao and Yang Xiao. 2012. Security and privacy in cloud computing. *IEEE communications surveys & tutorials* 15, 2 (2012), 843–859.

[142] Renchao Xie, Qinqin Tang, Shi Qiao, Han Zhu, F Richard Yu, and Tao Huang. 2021. When serverless computing meets edge computing: architecture, challenges, and open issues. *IEEE Wireless Communications* 28, 5 (2021), 126–133.

[143] George Xylomenos, Christopher N Ververidis, Vasilios A Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V Katsaros, and George C Polyzos. 2013. A survey of information-centric networking research. *IEEE communications surveys & tutorials* 16, 2 (2013), 1024–1049.

[144] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* 98 (2019), 289–330.

[145] Lanfranco Zanzi, Fabio Giust, and Vincenzo Sciancalepore. 2018. M2EC: A multi-tenant resource orchestration in multi-access edge computing systems. In *2018 IEEE wireless communications and networking Conf. (wcnc)*. IEEE, 1–6.

[146] Jiale Zhang, Bing Chen, Yanchao Zhao, Xiang Cheng, and Feng Hu. 2018. Data security and privacy-preserving in edge computing paradigm: Survey and open issues. *IEEE Access* 6 (2018), 18209–18237.

[147] Yutong Zhang, Boya Di, Pengfei Wang, Jinlong Lin, and Lingyang Song. 2020. HetMEC: Heterogeneous multi-layer mobile edge computing in the 6 G era. *IEEE Transactions on Vehicular Technology* 69, 4 (2020), 4388–4400.

[148] Pengyuan Zhou, Benjamin Finley, Xuebing Li, Sasu Tarkoma, Jussi Kangasharju, Mostafa Ammar, and Pan Hui. 2021. 5G MEC computation handoff for mobile augmented reality. *arXiv preprint arXiv:2101.00256* (2021).

[149] Siyu Zhou, Prasad Prakash Netalkar, Yanan Chang, Yang Xu, and Jonathan Chao. 2018. The MEC-Based Architecture Design for Low-Latency and Fast Hand-Off Vehicular Networking. In *2018 IEEE 88th Vehicular Technology Conf. (VTC-Fall)*. 1–7.