Beyond Multi-access Edge Computing: Essentials to realize a Mobile, Constrained Edge

Elisa Rojas*, Carlos Guimarães†, Antonio de la Oliva‡, Carlos J. Bernardos‡ and Robert Gazda§
*Universidad de Alcalá, Spain
†ZettaScale Technology, France
‡Universidad Carlos III de Madrid, Spain
§InterDigital, Inc., USA

***Abstract*—ETSI Multi-access Edge computing (MEC) main purpose is to improve latency and bandwidth consumption by keeping local traffic local while providing computing resources near the end-user. Despite its clear benefits, the next-generation of hyper-distributed applications (e.g., edge robotics, augmented environments, or smart agriculture) will exacerbate latency and bandwidth requirements, posing significant challenges to today's MEC deployments.**

In this work, we leverage on the current study item ETSI GR MEC 036, introducing a lightweight constrained version of a MEC platform that can be deployed in a mobile end terminal or in its closed locality. This work presents design options for cMEC, and how it can untangle the aforementioned gaps while being architectural compatible with a full-fledged MEC framework. Finally, key use cases and still open challenges are discussed, including recommendations to extend the current MEC standard towards constrained environments.

I. Introduction

In the field of edge computing, whose unquestionable benefits have boosted the emergence of new network services and applications, Multi-access Edge Computing (MEC) is the prevailing standardized framework. Under development of the European Telecommunications Standards Institute (ETSI), MEC is regarded as a key technology for the fulfillment of the core Key Performance Indicators (KPIs) of 5G [1] and beyond. Similarly to other edge computing paradigms (namely fog computing [2] and cloudlet computing [3]), MEC aims to decrease latency and traffic workload directed to a cloud infrastructure, consistently breaking down communications' latency and bandwidth utilization. In doing so, it provides clear benefits to massive Machine-Type Communication (mMTC), enhanced Mobile BroadBand (eMBB) and the Ultra-Reliable Low-Latency Communication (URLLC) use cases' families targeted by 5G technologies [1].

Forthcoming applications, namely the next-generation of hyper-distributed applications (e.g., edge robotics, augmented environments, or smart agriculture), are even stricter in their requirements, thus solely deploying MEC servers at the telco network edge might be insufficient. In fact, there are already scenarios in which the MEC framework prove to be limiting:

- **Loss of connectivity.** While on-the-move, devices might temporally lose their connectivity. Consequently, applications supported by a MEC server cannot guarantee service continuity. Although application relocation mechanisms exist, they either assume that the MEC infrastructure is deployed everywhere or that there are deployments in aggregation points of the infrastructure, making delays so large that the edge benefits are minimized.
- **Near-zero latency applications.** Computation offloading to an edge server might also be inadequate whenever applications require extremely low latency (i.e., sub-1ms robotics control loop). In addition, fluctuations on the communication would likely introduce undesirable jitter.
- **Privacy and security.** MEC is part of a multi-domain ecosystem composed by several stakeholders (e.g., infrastructure owners, service providers, system integrators and application developers) [4], thus placing generated data outside of the owner's domain. Although data privacy and security can be enforced by its owner, offloading functions to a MEC server increases the risk of a data leak or unauthorized access by a third-party [5].

The aforementioned challenges can be mitigated by exploiting dynamic computational offloading techniques. Complementary, integrating MEC platforms towards end-devices or constrained devices in the close vicinity of end-users is currently the subject of study in *ETSI GR MEC 036* [6], also devised by other Standards Development Organizations (SDOs), such as IETF [7]. A standardized method for integrating computation at constrained devices and traditional MEC servers, where the former preserves only subset of MEC capabilities, enables a holistic computational offloading while allowing resource orchestration at a finer granularity and exploitation of MEC services.

This article aims to contribute to such a vision by proposing the constrained MEC (cMEC) architecture, as a lightweight design of the MEC framework. By constrained device we refer to mobile end-devices or computational constrained mobile devices in the close locality of end-users. cMEC considers that constrained devices can on-board and support a subset of MEC functional elements to expand the computational reach of current MEC framework. MEC applications can then run locally and/or in a remote telco MEC system. In doing so, cMEC can take over on the applications execution whenever the connectivity to the network cannot be sustained, whether due to outage, mobility, or to incomplete coverage, and when the latency towards the edge MEC system is unreliable.

The remainder of the article is structured as follows. In Section II, we briefly describe the traditional MEC architecture for background reference; Section III presents a set of possible

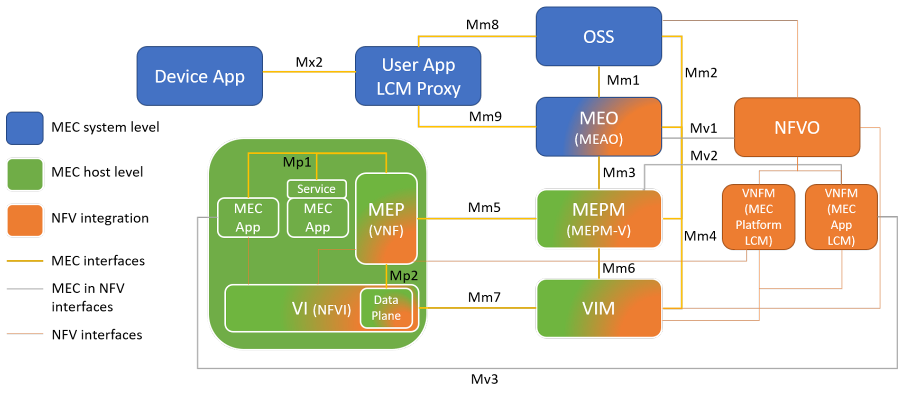

Figure 1: Simplified MEC reference architecture

cMEC use cases; Section IV illustrates the novel cMEC architecture with its general characteristic and innovations, proposing some workflows to integrate cMEC in the current MEC framework; the advantages and future challenges of this integration are discussed in Section V, and Section VI concludes the article.

II. MEC in a Nutshell

The MEC framework [8] was originally designed to be deployed at the edge with the goal of exempting end-devices from performing tasks locally, shifting the computation towards a virtualized platform of distributed elements with orchestrating and service capabilities.

Prevailing deployment option for MEC leverages Network Function Virtualization (NFV), which is an earlier (and thus more mature) network virtualization technology complementary to MEC. NFV provides a standardized framework for virtualizing network services, and its structure harmonizes with MEC. Additionally, as NFV represents the foundation of current 5G deployments and vendors are already exploiting NFV production-ready solutions, it is reasonable to think that even MEC, once sufficiently developed, would be integrated in this variant within real deployments, with MEC applications being treated as Virtual Network Functions (VNFs).

Figure 1 depicts a simplified architectural scheme of MEC and that of MEC in NFV, with the main functional components and reference points indicated. First, let us focus on the MEC architecture; a MEC system consists of a virtualized edge platform where MEC applications are executed and expose some API services. The general architecture can be divided into two levels: system level and host level. Hosts can be multiple, and their resources are handled by the system level components. At the system level, typically an Operational Support Systems (OSS) tool manages the instantiation and termination of MEC applications requested by a User Application Lifecycle Management (LCM) Proxy (UALCMP), receiving instructions from either an end user or a custom portal; the presence of a MEC Orchestrator (MEO) provides a general view on the whole MEC system, performs package on-boarding and selects the most suitable host where to deploy the application. At the host level, the MEC Platform Manager (MEPM) operates directly on the lifecycle of applications, while configuring traffic, security and DNS rules based on the application requirements; while the MEC Platform (MEP) is the environment that offers the MEC services to the MEC applications, and it also implements the DNS and traffic control rules for the applications. The computational, network, and memory resources of the platform are, eventually, managed by the Virtual Infrastructure Manager (VIM).

As for the NFVs integration, the assumption is that both MEPs and MECs applications are deployed as VNFs, which in the NFVs context are the virtual *bricks* of software constructing a specific Network Service (NS). Afterwards, the specific MEC management entities overlapping those of the NFV management and orchestration modules (NFV MANO) are cut out from the MEC blocks and delegated to the corresponding NFV functional elements. In practice, the MEPM becomes MEC Platform Manager - NFV (MEPM-V) and the part concerning the LCM of applications is delegated to a Virtual Network Function Manager (VNFM). Similarly, the MEO changes its name to Mobile Edge Application Orchestrator (MEAO), orchestrating a particular set of VNFs (e.g., MEC apps composing a NS) and delegating the orchestration of resources to the NFV Orchestrator (NFVO). The virtual infrastructure becomes that of the NFV framework (NFVI).

The cMEC proposed architecture (see Section IV) assumes the orchestration is held by the telco MEC deployment located in the telco infrastructure. The cMEC framework deployed in the constrained devices will leverage virtualization technology, running VMs or containerized applications, orchestrated by the integrated NFV/MEC functionality at the MEC. Hereinafter, to distinguish between both architectures, the constrained version will be referenced as cMEC, while the network infrastructure version as telco MEC (tMEC), to clarify they are both MEC frameworks following two distinct and complementary architectural approaches.

III. Overview of Use Cases Enabled by cMEC

Diverse use cases have fostered the need for cMEC. This section gathers four of the most distinctive.

A. Remote eHealth Monitoring

Remote monitoring in eHealth (e.g., on-board of ambulances in emergency situations) requires increased service reliability and availability while operating in very dynamic environments [9]. Different tasks might require distinct computing and/or data capabilities: *(i)* non-sensitive information can be offloaded into any shared computing resource; *(ii)* sensitive information cannot leave the vehicle and, therefore, it should not be processed remotely; and *(iii)* real-time information will need to be processed locally in the vehicle, or in a nearby infrastructure and devices to meet the latency requirements or lack of connectivity. While the former can be handled by standard telco MEC mechanisms, the later two require proper management and orchestration of local resources in constraint devices located at the ambulance or medical devices within it. Such requirements hinder a full end-to-end service provisioning by the standard MEC framework, requiring tMEC to take into consideration the capabilities of constraint devices in the ambulance. This consideration is currently out of the capabilities of ETSI MEC.

B. Zero-latency Augmented and/or Virtual Reality (AR/VR) applications

AR/VR applications are increasingly being adopted by both enterprise and end-customer domains to bring complete immersive experiences in numerous use cases (e.g., metaverse, 360 videos, or gaming). Since these applications are sensitive to human perception, they impose strict requirements in terms of latency in order to achieve accurate movements. Moreover, high computation power is also required to smoothly render virtual scenes. Although MEC appears as a suitable candidate to fulfil both requirements, it needs continuous connectivity between the user device and the MEC application in the telco edge, where any slight disruption will shatter the AR/VR user experience. In addition, any unexpected load in the link connecting the AR/VR application and the edge deployment may impact seriously on the user experience. A combination of both local and remote processing can be seen as a fallback solution: the tMEC resources are leveraged for high-resolution tasks, while on-device resources (mobile terminals, VR headsets, etc.) are responsible for lower-resolution tasks, triggered only if the offloaded computation arrives too late, or to intercede in case of connectivity failures. In addition, local devices can directly exchange information with nearby entities to enhance or enable new types of services (e.g., improve spatial coordinate-based scenes).

C. Smart Agriculture in Rural Areas

Smart agriculture presents a challenging use case to be supported by MEC, especially when it takes place in remote rural areas where connectivity is scarce and limited to specific points. In addition, isolated areas without permanent population (e.g., highly dense forests or Arctic areas) pose significant challenges for building a physical network infrastructure [10]. Therefore, the lack of a reliable connection towards a tMEC system hinders its utilization for applications that require a continuous synchronization and control. Resource-constrained platforms (e.g., Unmanned Aerial Vehicles (UAVs), harvesters, tractors, etc.) operating in remote areas could be transformed into functional mobile compute nodes, offering computing, storage and network resources under the control of MEC system to support the execution of applications, or interact with small servers deployed across the fields using radio access technologies [11]. Notwithstanding, supporting a MEC system (with all its complexity) on such battery-powered and resource-constrained devices exceeds the required functionalities and, consequently, reduces their lifespan between charges.

D. Advanced Collaborative Surveillance

Smart surveillance systems in cities are already envisioned for traditional MEC systems [12], where applications send their streams to a centralized server to be processed. However, a centralized solution is not only inefficient for such application, but also results in huge data traffic overhead. Several solutions implement on-board pre-processing as a way to reduce the traffic crossing the network. Such approaches hinder more dynamic scenarios where the location of cameras is changing, the application requires periodic updates, or where the surveillance resources are shared among different tenants (e.g., different departments of the city hall), each with different levels of access. Since cameras are not part of the MEC system, such actions must be performed via traditional (and manual) redeployment and reconfiguration procedures across the entire surveillance system. Integration of such devices into the MEC would ease updates while enabling its automation.

All previous use cases share a common requirement: constrained devices should support MEC functional elements on board to enable end-to-end management and orchestration of services. Notwithstanding, such requirement does not prevent task offloading to a tMEC system when local devices are not capable of efficiently running the task or when the requirements are more relaxed. With a consistent and flexible architecture to enable the integration of all the available resources in the entire cloud-to-things continuum, resource harvesting could be pooled together to enable a full and dynamic service provisioning between multiple and heterogeneous devices. Consequently, the proposed cMEC architecture aims to build the necessary adaptation to support MEC functional elements in constrained devices and provide a set of interfaces to interconnect both cMEC and ETSI standard tMEC.

IV. The Proposed cMEC Architecture

The pervasiveness of resources available in the end-user domain provide diverse computing and data capabilities. Consequently, they appear as a promising complement to traditional MEC systems in order to support novel latency and/or data sensitive applications. Still, such resources mostly comprise constrained devices with limited computational power, battery-powered and/or mobile, therefore hindering a straightforward support of full-fledged MEC solutions.

The cMEC architecture defines a lightweight design of MEC capabilities by extending the *cloud-edge-user*-layer architectural model with a new layer representing the user-domain devices (as depicted in Figure 2). The inclusion of this additional layer should be transparent to developers and users, handling its complexity the MEC system and its APIs.

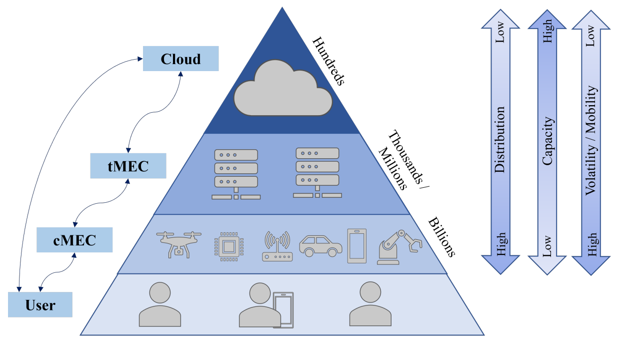

Figure 2: The 4-layer architecture: User-cMEC-tMEC-Cloud

A. From a 3-Layer to a 4-Layer Architectural Model

The cMEC departs from ETSI MEC framework and presents characteristics tailored and specific to constrained devices:

Lightweight Functionalities: The cMEC can be deployed as a full-fledged MEC system (i.e., including all its functional elements), but the limited resources available in the devices might impose the support of only a subset of the MEC functionalities. As an example, the MEO comprises resource-demanding functions as it is responsible for deciding in which host applications will be deployed. This might exceed the capabilities of the end-user devices, not being considered a primary function particularly in environments where cMEC and tMEC collaborate (as tMEC might provide that orchestration instead). Thus, it will be opt-out in most constrained situations, unless a lightweight operation can be provided.

A Layered Approach: As tMEC relies on the cloud for computational offloading, content fetching, user authentication, and context, cMEC relies on the tMEC for the same purposes. Such a layered approach should depend on an interconnection relation between the cMEC and tMEC, disregarding the implementation of federation concepts that imply explicit business agreements and rely on orchestrators. In fact, according to the study on inter-MEC system connection and federation [13], MEO is considered the key enabling element for many workflows, but cMEC may not support it. Notwithstanding, a given cMEC can decide on sharing different resources with distinct tMECs, using its orchestrating capabilities, or even peer cMECs.

Dependency from a tMEC System: Whenever the cMEC does not implement a specific MEC function, it needs to rely on the upper-layer tMEC system to offer the missing functionalities. Novel workflows, MEC application development guidelines, and specific interconnection mechanisms must then be implemented to compensate for the absence of functions.

End-User Device Co-location and Awareness: The cMEC system can be co-located in the same end-user device as the MEC application or it can run in a constraint device in its close proximity. The end-user device can take part of the cMEC integration as follows:

1) *cMEC-aware*: end-user device and cMEC are in the same local network or their identity is known to each other (e.g., the cMEC runs on that end-user device). The end-user device can inspect the cMEC systems available and request the instantiation of a MEC application, which in turn triggers the interconnection of the cMEC to a tMEC.
2) *cMEC-unaware*: end-user device is not aware of a nearby cMEC and therefore requests the instantiation of a MEC application towards the tMEC. The tMEC, knowing there is a cMEC deployment near the user, decides to instantiate the application on an interconnected cMEC.

OSS: The OSS is a service provider tool operated at the MEC level and shall not necessarily be linked to a subordinate local cMEC for application on-boarding and instantiation. These actions, traditionally performed by a network manager operating on the MEC through the OSS, may need to be initiated by the end user (e.g., requesting a particular application for their house or car), and handled by the cloud and the tMEC remote OSS and MEO, employing alternative workflows supporting a new set of cross-system MEC interfaces. That means interfaces *Mx2* and *Mm8* in Figure 1 should be enhanced to allow users to trigger new instantiations.

B. Architectural scheme for cMEC and tMEC interconnection

Given the aforementioned points, Figure 3 details the architectural scheme to interconnect the cMEC with the tMEC, without the MEO being present in the cMEC system. The cross-system reference points *inter-Mm2* and *inter-Mm3* are mainly introduced for the cMEC-tMEC interconnection setup. *Mx2* reference point is extended to allow users to trigger the lifecycle management (e.g., instantiation, deletion, or update) of MEC applications in a cMEC or even a tMEC. Thus, the *inter-Mx2* interface, which connects the cMEC app proxy to that of the tMEC, can guarantee a certain level of concurrence between the cross-systems applications (i.e., those applications distributed across several layers), and allow any request to be propagated from cMEC up to the tMEC. Finally, *Mp1* reference point, which connects the MEC applications and services and their platform within each other, should be extended as a *inter-Mp1* reference point for service consumption and app-to-app communications between different systems.

C. High-level cMEC workflows

The integration of cMEC with a tMEC requires additional workflows. In the following, three key operations are described: *(i)* discovery and interconnection; *(ii)* application on-boarding and instantiation; and *(iii)* service availability and consumption.

1) Discovery and Interconnection: The cMEC discovery by the tMEC or by other cMEC systems is a necessary step to their interconnection. It consists of either *(i)* making a tMEC system aware of a the cMEC; or *(ii)* discovering peer cMECs. Moreover, the cMEC does not support orchestration (i.e., it does not comprise a MEO element). At the same time, cMEC can be co-located in an end-user device. The challenge for an end-user device to discover a nearby standalone cMEC is left outside of the scope of this work, since multiple protocols (not directly related to ETSI MEC) can serve this purpose.

The message workflow for a cMEC to advertise itself to a tMEC is presented within the box titled *Discovery and Interconnection* of Figure 4. The cMEC reaches out to the

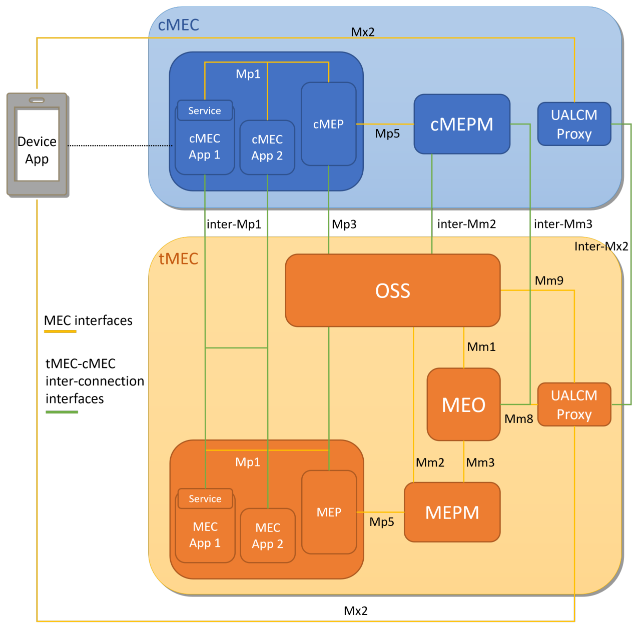

Figure 3: Architectural scheme of cMEC together with tMEC

UALCMP of the tMEC it wants to integrate with, by issuing a *Request for Integration* message (step 1), through which the cMEC advertises the interfaces and the computational capabilities (i.e., computing, storage and network resources, MEC services, etc.) to be shared (step 2). The OSS can then update the catalogue of interconnections with this new information (step 3), as cMECs rely on tMEC MEO for coordination. Afterwards, the cMEC can proceed to send the agreed interface addresses (step 4).

After signalling between the cross-system interfaces (OSS contacting cMEPM through *inter-Mm2* and MEO contacting cMEP through *inter-Mm3*) to check interconnectivity (step 5), the process is finalized when the interconnection is activated in the OSS (step 6). An activation step is necessary as the cMEC can move away from the tMEC during the procedure. Finally, the MEO module adds the cMEC to its host list (step 7), so that, if granted permissions, the cMEC host can be selected by the orchestrator for application on-boarding and instantiation.

2) Application On-boarding and Instantiation: In standard ETSI MEC, the package on-boarding request for an application is initiated by the operator interacting with the OSS. Then, the actual application instantiation is subject to the MEO's decision, which normally evaluates the application requirements and performs host selection accordingly. As previously mentioned, a cMEC host connected to a tMEC, becomes a host of the tMEC system, so that it can be automatically selected by the MEO for application deployments when needed. As illustrated in the box titled *Application On-boarding and Instantiation* of Figure 4, the cMEC Device App contacts the tMEC UALCMP, which solicits the OSS to grant the on-boarding permissions. The same on-boarding request would then reach the MEO (step 1). At this point, the MEO would have, according to the current standard, to perform host selection. The current specification does not define how host selection is realized in practise. In such case, the cMEC could request the MEO to select the desired cMEC, and not an arbitrary host of the tMEC system selected by the MEO's algorithm. The actual package on-boarding and app instantiation processes are later triggered by the MEO in the cMEPM through the *inter-Mm3* reference point (step 2).

3) Service Availability and Consumption: A MEC application, whether deployed in a cMEC or a tMEC, might also request a MEC service not locally available. As the *Service Availability and Consumption* box of Figure 4 represents, this can be tackled by issuing a service request to the OSS (step 1), followed by a lookup in the catalogue of interconnections (step 2). The lookup goal is to identify if a service is available in a cMEC or tMEC, which would then communicate the availability details (step 3) to the MEC application. If the requesting MEC (cMEC or tMEC) is not interconnected to the target MEC, the interconnection is invoked by the MEO or the OSS. The service consumption between the MEC application and the remote service can then occur via the *inter-Mp1* interface (step 4). Alternatively, a dedicated service management proxy can be introduced in every cMEC to manage service availability. However, it prevents the cMEC to benefit from remote services belonging to cMEC systems not directly interconnected: proxies must be known to the cMECs in advance. A last option can rely on sending queries about service availability directly towards the MEO, which then queries each of the cMEPMs and provides an answer based on the information stored in their cMEP's *service registries*.

V. cMEC Advantages

cMEC paves the way to novel opportunities of deploying tailored and optimized applications across the cloud-to-thing continuum, but it also imposes new challenges to be tackled.

A. *Why extend MEC to contrained end-devices?*

- **Virtualization and orchestrating capabilities**: Microservice-containerized architectures are becoming predominant for embedded and constrained device applications. Having a MEC-compliant system on board of such devices can support the management of many different concurrent and distributed applications.
- **Services**: Developers can design more efficient and flexible applications. cMEC applications can be deployed as services with great flexibility.
- **Application lifecycle management**: The lifecycle management of applications in mobile/constrained devices becomes automated and flexible thanks to the functional elements encompassed in the MEC framework.

B. *Which added value will cMEC adoption bring?*

- **Lower latency**: The edge computing paradigm is built on the assumption that execution of heavy computational tasks should be offloaded. However, cMEC is founded on the fact that more gain would be achieved if the computational capacity were further spread among end-user devices, even if less powerful than the edge. This can benefit applications with stringent latency requirements and reduction in back-haul bandwidth utilization.

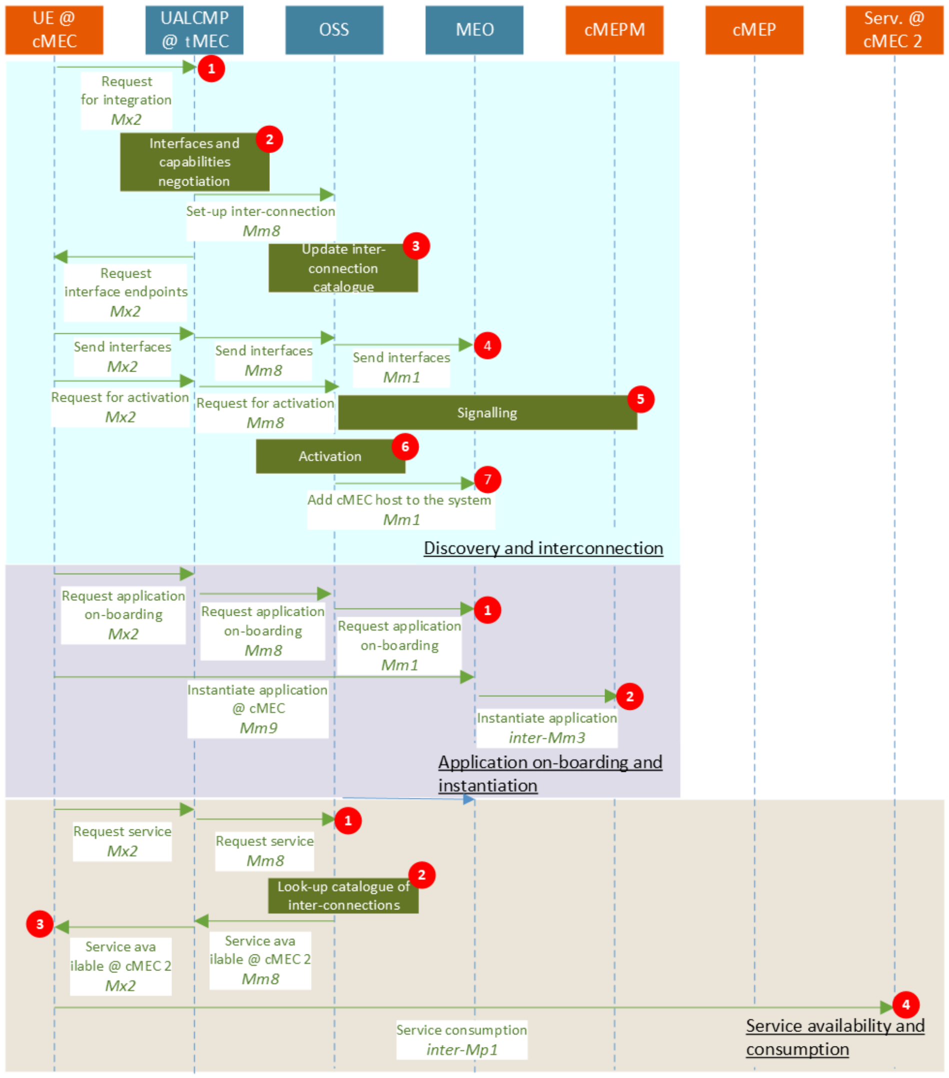
UE @ cMEC
UALCMP @ tMEC
OSS
MEO
cMEPM
cMEP
Serv. @ cMEC 2
1
Request for integration Mx2
2
Interfaces and capabilities negotiation
Set-up inter-connection Mm8
3
Update inter-connection catalogue
Request interface endpoints Mx2
Send interfaces Mx2
Send interfaces Mm8
Send interfaces Mm1
4
Request for activation Mx2
Request for activation Mm8
Signalling
5
Activation
6
Add cMEC host to the system Mm1
7
Discovery and interconnection
Request application on-boarding Mx2
Request application on-boarding Mm8
Request application on-boarding Mm1
1
Instantiate application @ cMEC Mm9
Instantiate application inter-Mm3
2
Application on-boarding and instantiation
Request service Mx2
Request service Mm8
1
Look-up catalogue of inter-connections
2
3
Service available @ cMEC 2 Mx2
Service available @ cMEC 2 Mm8
Service consumption inter-Mp1
4
Service availability and consumption

Figure 4: High-level cMEC workflows

- **Better bandwidth utilization**: tMECs will be overloaded when massive offloading of computational tasks to the edge occur, as envisioned by future applications. cMEC enables the pervasiveness of available resources in the end-user domain to seamlessly share their resources with the edge, not only distributing the computation load but also the bandwidth utilization across network segments.
- **Enhanced reliability and resilience**: Partitioning the intelligence of the network and locating parts of it on cMEC devices close to its consumer, drastically reduces losses or application disruption since local functions can be run even without connection to network infrastructure.
- **Increased security and privacy**: For applications handling sensitive information (e.g., eHealth), security and privacy is of paramount importance. cMEC tackles this requirement at its root by enabling MEC applications to run on end-user devices where sensitive information is generated. However, if devices are owned by different stakeholders, additional mechanisms should be implemented to grant enhanced security.

C. *Which challenges are still ahead?*

- **Dynamic and distributed infrastructure:** Since cMEC can handle mobile and/or battery-powered end-devices, devices are likely to join or leave the computing infrastructure or migrate to a different location. Such occurrence will continuously change the topology of the infrastructure and its computational capabilities in a given area. In doing so, the operation of applications running therein, or even the entire E2E application, will get disrupted. The integration of cMEC and tMEC must envision mitigation mechanisms, either by requiring a new set of connectivity requirements to be defined, or by enforcing migration or fallback mechanisms.
- **Heterogeneity of devices:** cMEC requires handling heterogeneity of end-devices with distinct computing, storage, and networking capabilities, not only making it more complex to manage the infrastructure but also to orchestrate E2E applications. In order to reduce such complexity, the cMEC should apply an abstraction layer when interacting with tMEC.

- **Distributed orchestration**: The architectural design of cMEC facilitates flat-hierarchy deployments. As the cMEC does not include an orchestrator, further studies should be performed on, for instance, auction-based federation solutions [14], in which no business agreements are required between peers, which could serve as a dynamic solution for cMEC to join a larger tMEC domain.
- **Ownership of end-devices:** End-devices managed by cMEC and attached to existing tMEC systems are considered third-party nodes that do not belong to the MEC provider. Therefore, the MEC provider is limited in terms of management and control procedures that can be used, making the fulfilment of the applications' requirements more complex. The request for a cMEC to be part of a tMEC system might require dynamic agreements (e.g., by means of resources federation mechanisms) so that the entire system can become more flexible and react faster to changes. Still, such agreements must also enforce monitoring and auditing capabilities so that a break in any E2E application SLAs can be identified and accountable.
- **Security and trust:** cMEC requires applications to run on end-devices which trustfulness cannot be guaranteed. Thus, the runtime environment for cMEC must provide a certain level of isolation and encapsulation in order to reduce the surface attack. Moreover, security requirements must be considered by the MEO whenever deciding on the orchestration of E2E applications. If end-devices are self-managed by users, they can be considered safe to their own applications or services.

VI. Conclusion

In this article, we presented a novel MEC variant for mobile and constrained devices named cMEC, envisioned as a holistic solution to enable MEC capabilities down to end-devices. Although many works already exist in the literature about distributed edge computing, most focus on specific scenarios (e.g., optimizing task offloading in MEC and cloud) and none provides an integrated solution aligned with industry requirements. This work presents an architectural solution devoted to accomplish them, along with an analysis of its main benefits and challenges ahead. As future work, a proof-of-concept of cMEC will be implemented and evaluated for a quantitative evaluation and the verification of its added value.

VII. Acknowledgement

This work received support from the EU's Horizon 2020 research and innovation programme under grant agreement ID no. 101070177 (ICOS), and European Union's Horizon Europe research and innovation programme under grant agreement No 101095759 (Hexa-X-II). It was also funded by grants from Comunidad de Madrid through project MistLETOE-CM (CM/JIN/2021-006), by project ONENESS (PID2020-116361RA-I00) of the Spanish Ministry of Science and Innovation, by the "Ayudas para la Recualificación del Sistema Universitario Español 2021-2023" program of Universidad de Alcala, and by the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D 6G-EDGEDT.

Biographies

Elisa Rojas (M.Sc.'2009, Ph.D.'2013) is an Assistant Professor at Universidad de Alcalá (UAH). She is an ambassador of the ONF and her current research areas include SDN, NFV, 5G networks, routing algorithms, IoT, data center networks.

Carlos Guimarães (M.Sc.'2011, Ph.D.'2019) is a Senior Technologist at ZettaScale Technology (France) where he develops data-centric networking solutions. His current research interests are computer networks and telecommunications.

Antonio de la Oliva (M.Sc.'2004, Ph.D.'2008) is an Associate Professor at Universidad Carlos III Madrid (UC3M). He is an active contributor to IEEE 802 where he has served as Vice-Chair of IEEE 802.21b and Technical Editor of IEEE 802.21d. He has published more than 30 papers on different networking areas.

Carlos J. Bernardos (M.Sc.'2003, Ph.D.'2006) is an Associate Professor at Universidad Carlos III Madrid (UC3M). His current research interests are network virtualization and wireless networks. He is an active contributor to the IETF.

Robert Gazda Robert (Bob) Gazda is a Senior Director in InterDigital's Wireless Networking Lab. Bob is an accomplished engineering professional and technologist with over 20 years of industry experience in wireless telecommunications, networking, and embedded systems. At InterDigital, Bob leads research and innovation focused on 5G/6G Distributed and Converged Computing and Communications Architectures.

References

[1] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "MEC in 5G Networks," *ETSI white paper*, vol. 28, pp. 1–28, 2018.

[2] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, C. Mahmoudi *et al.*, "Fog Computing Conceptual Model," 2018.

[3] M. Babar, M. S. Khan, F. Ali, M. Imran, and M. Shoaib, "Cloudlet Computing: Recent Advances, Taxonomy, and Challenges," *IEEE Access*, vol. 9, pp. 29 609–29 622, 2021.

[4] D. Sabella, A. Reznik, K. R. Nayak, D. Lopez, F. Li, U. Kleber, A. Leadbeater, K. Maloor, S. B. Mary Baskaran, L. Cominardi, C. Costa, F. Granelli, V. Gazis, F. Ennesser, and X. Gu, "MEC Security: Status of Standard Supports and Future Evolutions," *ETSI white paper*, vol. 46, pp. 1–26, 2021.

[5] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data Security and Privacy-preserving in Edge Computing paradigm: Survey and Open Issues," *IEEE access*, vol. 6, pp. 18 209–18 237, 2018.

[6] *DGR/MEC-0036ConstrainedDevice*, ETSI Std. v3.0.4 Draft, 2021.

[7] *IoT Edge Challenges and Functions*, IETF Std. draft-irtf-t2trg-iot-edge-08, 2023.

[8] *GS MEC 003: Multi-access Edge Computing (MEC); Framework and Reference Architecture*, ETSI Std. v2.2.1, 2020.

[9] J. Islam, T. Kumar, I. Kovacevic, and E. Harjula, "Resource-Aware Dynamic Service Deployment for Local IoT Edge Computing: Healthcare Use Case," *IEEE Access*, vol. 9, pp. 115 868–115 884, 2021.

[10] A. M. Cavalcante, M. V. Marquezini, L. Mendes, and C. S. Moreno, "5G for Remote Areas: Challenges, Opportunities and Business Modeling for Brazil," *IEEE Access*, vol. 9, pp. 10 829–10 843, 2021.

[11] V. Sanchez-Aguero, I. Vidal, F. Valera, B. Nogales, L. L. Mendes, W. Damascena Dias, and A. Carvalho Ferreira, "Deploying an NFV-based Experimentation Scenario for 5G Solutions in Underserved Areas," *Sensors*, vol. 21, no. 5, p. 1897, 2021.

[12] *GS MEC 002: Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements*, ETSI Std. v2.1.1, 2018.

[13] *GR MEC 035: Multi-access Edge Computing (MEC); Study on Inter-MEC systems and MEC-Cloud Systems Coordination*, ETSI Std. v3.1.1, 2021.

[14] K. Antevski and C. J. Bernardos, "Federation of 5G Services Using Distributed Ledger Technologies," *Internet Technology Letters*, vol. 3, no. 6, p. e193, 2020.