2017 IEEE 5th International Conference on Future Internet of Things and Cloud

Converging Mobile Edge Computing, Fog Computing, and IoT Quality Requirements

Paolo Bellavista, Luca Foschini, Domenico Scotece
Computer Science and Engineering Dept. (DISI)
University of Bologna
Viale del Risorgimento, 2 – 40136 Bologna, Italy
{paolo.bellavista, luca.foschini, domenico.scotece}@unibo.it

***Abstract*— This position paper proposes a novel and integrated architectural model for the design of new 5G-enabled supports, capable of synergically leveraging Mobile Edge Computing (MEC) and fog computing capabilities together. In particular, we claim the relevance of dynamically distributing monitoring and control intelligence close to sensor/actuator localities in order to reduce latency in the control loop and to enable some forms of at least partial decentralized autonomous control even in absence of (temporary) cloud computing availability. This scenario significantly benefits from the possibility of having functions that are dynamically migrated from the global cloud to the local 5G-enhanced edges (and possibly vice versa), in order to best fit the characteristics of the deployment environment and of the supported Internet of Things (IoT) applications at provisioning time. Among the others, the paper details and discusses the technical challenges associated with i) the quality-constrained exploitation of container-based virtualized resources at edge nodes and ii) the quality-constrained integration of IoT gateways. The reported use cases help to practically understand the benefits of the proposed integrated architecture and shed light on most relevant and open related technical challenges.**

Keywords—fog computing, 5G, Mobile Edge Computing, Cloudlet, OpenFog, Smart Gateways, Virtual Resources.

I. Introduction

The recent advances in telecommunication technologies, the wide availability of powerful and always-connected smart devices, and the maturity of cloud computing, together with the cloudification of core telco services, are paving the way to the novel 5G scenario. Focusing on convergence and integration aspects, 5G should enable not only the full convergence of heterogeneous wireless technologies, but also, and most important, it should seamlessly support the delivery of a wide spectrum of different applications with highly challenging and very diverse requirements, such as in terms of latency, bandwidth, management, and so forth. In this area, two classes of proposals, namely, the standardization effort under the name of Mobile Edge Computing (MEC) and the platforms/solutions that start to be available under the name of fog computing, have recently emerged as important enablers [1, 2].

MEC is an architectural model and specification proposal (i.e., by European Telecommunications Standards Institute - ETSI) that aims at evolving the traditional two-layers cloud-device integration model, where mobile nodes directly communicate with a central cloud through the Internet, with the introduction of a third intermediate middleware layer that executes at so-called network edges. This promotes a new three-layer device-edge-cloud hierarchical architecture, which is recognized as very promising for several application domains [1]. In fact, the new MEC model allows moving and hosting computing/storage resources at network edges close to the targeted mobile devices, thus overcoming the typical limitations of direct cloud-device interactions, such as high uncertainty of available resources, limited bandwidth, unreliability of the wireless network trunk, and rapid deployment needs.

In parallel, with a stronger emphasis on the (sensing) devices and communication-enabled things, hence stemming from the Internet of Things (IoT) scenario, Cisco has initially proposed the fog computing (or simply fog for the sake of briefness) paradigm, and we have recently witnessed some related standardization efforts inside the Open Fog Consortium [3]. The proposed fog model shares with MEC the idea of interposing an intermediate layer, deployed at the edge of the network, between final devices and the central cloud. This layer can be organized in a very flexible way by composing the resources and services offered by all the local devices, which can usually interact thanks to the intermediation of a local proxy, often called Smart Gateway (SG) in this context, and via direct point-to-point ad-hoc IoT interconnections.

MEC and fog focus on (and permit to increase) the quality and performance of several cloud-assisted device services, but we claim that these models still face, each of them separately, some non-negligible incompleteness and weaknesses. Starting with MEC, the number of employed edges is generally limited, since edges introduce additional costs of operation for supported services, such as deployment, maintenance, and configuration costs for telco operators. Moreover, MEC typically works in infrastructure mode, being unable to easily leverage the resources available in surrounding devices at runtime: once MEC edges are deployed, they are rarely and hardly re-deployed (high cost of re-configurations) in other positions and this might be highly inefficient, e.g., when service load conditions significantly change during provisioning, such as during specific time slots, maybe with daily, weekly, or yearly patterns. Focusing on fog, instead, although its more decentralized architecture (at least from a control/management perspective) makes it more flexible, at the same time it complicates its management and the possibility to leverage the monitored context (e.g., resource usage and availability) typically available in infrastructure-oriented MEC telco environments. In addition, fog use cases are tailored mainly for resource-poor devices and sensing scenarios, and so SGs are typically unable to host heavy computations, such as in the case of a video surveillance service that monitors an environment with smartphone cameras capturing photos/videos of the surroundings and that exploits face recognition to trace suspicious users' movements.

To overcome such limitations, we claim the need for new solutions able to bring the best of the two MEC and fog approaches by integrating them into a unique and fully-converged 5G architecture. Notwithstanding recent advances in both MEC and fog, to the best of our knowledge, only a very limited number of seminal works, such as [4] and [5], has explored the mutual advantages in the joint synergic exploitation of these two classes of solutions. However, these few works in the literature are mostly focused on a specific goal and do not provide any in-depth analysis of the pros/cons of integrating MEC and fog, as well as of the main related integration challenges. Moreover, even if it starts to be envisioned, at the current stage, a general architectural model able to truly ease the integration of the existing MEC/fog functions is still largely missing; contributions covering this gap can relevantly help in paving the way to the development of new classes of services that exploit the full (and still largely unexplored) potential of intertwining MEC and fog supports.

In this perspective, the paper proposes a new architectural model based on the introduction of a fully-converged 5G-Enabled Edge (5GEE), as a core element deployed at the intermediate layer in the proximity of the mobile node. Our 5GEE support enables the combination and joint exploitation of MEC and fog capabilities by contributing with several core elements of technical novelty. First of all, after providing some needed background about MEC and fog standards and related integration literature, we propose a *thorough and in-depth analysis of all the main MEC and SG functions and capabilities*. Second, we present the main design guidelines of our new 5GEE architectural model that aims to overcome MEC and fog limitations by enabling *the activation and dynamic management/(re-)configuration of 5GEE entities according* to service-specific needs in the served device localities. Third, we detail *a possible implementation blueprint of the proposed support* with the aim of both leveraging resources contributed by all locally available mobile devices and enabling/facilitating the definition of new services through the composition of existing functions. To neatly separate new services from existing functions, while granting resource isolation, 5GEE exploits lightweight cloud and container technologies, such as OpenStack and Docker, that simplify the deployment and fasten the (de-)/activation of support functions and services; moreover, 5GEE employs MANagement and Orchestration (MANO) support based on OpenBaton to take over all needed management issues [6, 7]. Fourth, *we show our 5GEE at work in two significant case studies* that we are currently realizing as part of our ongoing work on Mobile Crowd-Sensing (MCS) in the context of the ParticipAct MCS living lab, i.e., a wide-scale experiment that involved about 170 students of the University of Bologna for more than two years [8].

The remainder of the paper is structured as follows. The next section gives some needed background material and overviews the most important literature in the field. Section 3 describes MEC and fog functions, as well as reports the main challenges to realize a fully-integrated 5G edge infrastructure. Section 4 introduces the architecture of our 5GEE middleware and presents its inner implementation. Finally, Section 5 shows our use cases and, based on them, highlights most open technical challenges in the field, while some conclusions and directions of future work end the paper.

II. Background & Related Work

A. Mobile Edge Computing (MEC)

MEC is considered an emerging technology and an important component of future generation 5G networks. The main idea of MEC is to place storage and computation resources at the network edge, in the proximity of users. MEC can be seen as a relatively small datacenter running at the edge of a mobile network and performing tasks, such as hosting services on behalf of mobile nodes, that could not be achieved with the support of traditional network infrastructure. In the literature, we have two main "flavors" of MEC:

- the European Telecommunications Standards Institute (ETSI)-oriented one. ETSI published a first well recognized white paper where it emphasizes how MEC is characterized by low latency, proximity, high bandwidth, real-time, radio network requirements, and location awareness. In addition to the already mentioned standardization efforts and specifications, ETSI provides some examples of use cases that would benefit from the adoption of MEC solutions, for example, augmented reality, video analytics, gaming, IoT applications, etc. [1].
- The cloudlet-based Edge Computing one. According to Satyanarayanan who first proposed them, "Cloudlets are a decentralized and widely dispersed Internet infrastructure whose compute cycles and storage resources can be leveraged by nearby mobile devices" [9]. This paradigm enabled mobile devices to make use of the resources of a cloudlet in order to offload resource-intensive applications. In fact, one of the main limitations of cloud computing is latency, as well as low responsiveness in interactions that involve user devices and remote cloud support, thus hurting usability and quality. Cloudlet-based Edge Computing can help to alleviate this issue by using nearby resource-rich cloudlet accessible via one-hop wireless communication. Another main aspect of cloudlet is virtualization. In fact, services deployed as virtual machines can be easily migrated and Cloudlet environments are used to support service handoff between two cloudlets by using virtual machine overlays.

B. Fog Computing

Fog computing starts from the primary idea of extending cloud computing and services to the edge of the network [2]. Similar to MEC, fog provides data, storage, computing, and application services to end-users, but with stronger emphasis on virtualization techniques on the one hand, and on smart city and IoT scenarios on the other hand.

The OpenFog Consortium was started by ARM, Cisco, Dell, Intel, Microsoft, and Princeton University in November 2015. Through the OpenFog Reference Architecture (OpenFog RA) [3], the consortium is intended to help developers to implement and deploy fog-based solutions. The OpenFog RA describes a generic fog platform that is designed to be applicable to any scenario or application, from transportation and agriculture to smart cities and smart buildings, from healthcare to virtual sensor networks, by providing business value for IoT applications that require low latency, real-time decision making, and other constrained quality indicators. The OpenFog RA is based on a set of core principles, called pillars: these pillars represent the key attributes that a system needs to embody and include security, scalability, openness, autonomy, agility, hierarchy, and programmability. For our purposes, the most interesting pillar is hierarchy, as better detailed in the following sections: depending on the scalability and nature of the scenario, the hierarchy may be a network of smart and connected partitioned systems arranged in either physical or logical layers, or it may collapse into a single physical system.

C. Related Work about Fog and MEC integration

Together with the above main standardization efforts, in the last years there are also a very few seminal works aimed at exploring the idea of integrating Fog and MEC paradigms, i.e., Mobile Edge Fog Computing and Foud presented below.

Mobile Edge Fog Computing (MEFC) is a MEC technology that provides cloud computing capabilities within the access network, close to the mobile subscribers [4]. The MEFC architecture is based on relay gateways to enable the fog trunk. Gateways form a MEC server network with Device-to-Device (D2D) communications and routing capabilities. The main idea is to enrich the traditional MEC platform by including two new modules, i.e., request and service handlers. The request handler module manages service requests from smart devices, while the service handler module dispatches requests to servers or to neighbor D2D Relay Gateways. Shubhranshu et al. have split up service requests and service handlers in different steps. First of all, the end device sends a service request to the nearest D2D relay gateway. The relay gateway parses the request to get the service information: if the service exists locally, the relay gateway performs that service; otherwise, the request is forwarded to another gateway. Finally, the service handler module returns the result to the device. The main limitation of this proposal is that it mainly focuses on routing communications aspects and does not adequately address the (several) open technical challenges and issues associated with integration management.

Foud is a fog-based and cloud-based architecture for Vehicle-to-Grid (V2G) networks in 5G [5]. The Foud architecture consists of two entities: permanent cloud and temporary fog. Cloud entity performs conventional services such as storage, computing, and network resources. The temporary fog, instead, implements the following functions that run at local on-board modules: Local AGgregator (LAG), Certification/Registration Authority (CA/RA), and Control Centers (CC). Supporting different computing functions and integrating 5G communications, Foud could enable the end-user authentication in the V2G system to authorize access to all available stationary and mobile computing resources. Similarly to MEFC, also Foud organizes the involved architecture components into three layers: the user layer, the network layer, and the service layer. However, it focuses mainly on security aspects and is specialized for the support of applications in the vehicular cloud domain.

III. MEC AND FOG: FUNCTIONS AND ARCHITECTURES

The proposed 5GEE architecture aims at converging MEC and fog computing paradigms while maintaining quality awareness and orientation. With the goal of identifying possible similarities and mutual benefits offered by these two architectures, this section focuses and presents the main architectural models and functions of MEC and fog computing.

The architectures of MEC and fog computing share many similarities. First of all, as shown in Figure 1, they adopt the same distributed architecture skeleton that consists of a *central cloud*, a *middleware* (middle) *layer*, typically deployed at the edge of the network at the user locality, and *end devices* that might connect either directly through mobile heterogeneous networks, as it is usual for the fog, or through a wireless access points, such as for cellular mobile nodes connected through a radio access network.

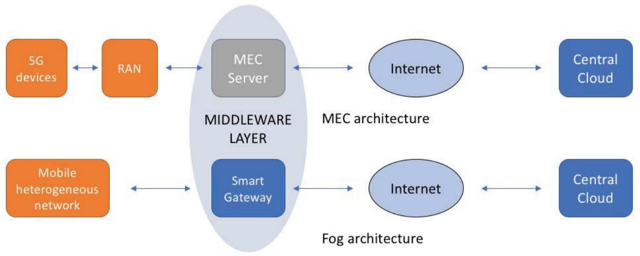

Figure 1. MEC and fog architectures.

Focusing on the crucial *middleware layer*, for both architectures, it provides a set of functions close to the edge. Those features are sometimes similar, other times different and complementary, and can be seen as enablers for the development of new 5G service scenarios. Starting with MEC, at the middleware layer we can find MEC servers that are usually implemented by telco providers to enrich their network infrastructure with new services. These servers are typically connected directly at the base station and include telco-oriented facilities. Based on the existing literature (i.e., [1, 9]), we have identified the following main MEC features (we identify each of them with an incremental id MFi, and for each of them we report a brief description of its main pros/cons):

MF1. *Execution of resource-intensive applications.* This function enables the execution of telco multimedia and mobile applications close to the edge to grant ultra-low latencies below 1ms (one of the core 5G requirements). A main limitation is that it usually does not support the sharing of resources across different nodes in the same locality.

MF2. *Context data services*. This function, typical of telco infrastructures, allows the MEC node to retrieve context information about the Radio Access Network (RAN) it is logically linked to, such as cell load, user location, and allocated bandwidth in a very efficient way. This function is highly beneficial for the development of new classes of context-aware services.

MF3. *Pre-processing of data*. This can enable, for example, the real-time video analytics scenario detailed later as a relevant use case, where video data is processed by MEC servers and the application can detect and notify only specific events to the central cloud. This can help to guarantee required privacy and to avoid useless transmissions of huge amount of data across the network via locality-based proximity processing.

MF4. *Caching of multimedia contents.* This function enables contents caching, i.e., one of the basic building blocks to minimize round trip time and to maximize throughput for better quality, for example in multimedia content streaming. In the currently available proposals, this is typically realized by a single MEC node; it is envisioned that it could benefit from leveraging also storage resources of other MEC nodes and mobile devices in the locality.

MF5. *Resource management.* This facility enables the reservation of slices of resources along the whole end-to-end service path including the MEC node. To be efficient, that typically requires also to predict incoming loads and observed patterns with a relatively good approximation. Resource management is actually adopted only for the data plane; to grant high quality levels, an interesting extension that has just started to be investigated in the related literature is about the introduction of resource slicing also at the management plane.

MF6. *Mobile offloading*. This facility has been originally proposed to operate cloudlets by enabling the remote execution of services (or pieces of service) as mobile node companion applications at MEC nodes. How to automate the decision of which part(s) of the application to offload according to the actual context and the characteristics of the application is still an important research direction. In addition, novel forms of offloading, considering also the opportunity to dynamically migrate functions from MEC nodes to the global cloud start to be investigated, for even greater flexibility at provisioning time.

Instead, the fog architecture is more oriented to industrial SGs, typically equipped with more modest resources. In this case, the middleware layer aims at providing connectivity to the underlying Mobile Heterogeneous Networks (MHNs) and to execute services. The fog architecture is usually used for helping heterogeneous devices, such as sensors, actuators, and smartphones to overcome cloud computing limitations. In order to guarantee/support differentiated quality levels, we have identified some SG features:

FF1. *Mobile Heterogeneous Network*. This facility enables the automatic building of network topologies with heterogeneous devices that exploit (possibly multiple) heterogeneous forms of wireless connectivity. In addition, SG may allocate them network resources in order to support some forms of quality management over the provided connections.

FF2. *D2D communication support*. This support function allows either communicate in D2D modality or facilitate and make more efficient the D2D-based interactions of devices in the 5GEE node locality. For instance, SG may work as a cluster head in a peer-to-peer collaborative network of devices that are directly under its local control.

FF3. *Context data services*. This function is complementary to the context data service function in MEC. In fact, MEC context data service focuses on the context information available at the infrastructure side; instead, in fog environments, this function helps to maintain and to know who is in the network, but also to characterize the sensed (physical) surrounding environment.

FF4. *Storage and aggregation of data*. This facility allows to store data collected from heterogeneous sensors or devices. Considered the requirements on reliability and durability of data in a given locality, after an agreed period, SG can synchronize the whole (or only the significant part of the) locally aggregated data to the cloud.

FF5. *Discovery/Registry of services*. In fog environments, in order to support the underlying IoT and D2D models that are typically highly decentralized, it is necessary to include and offer service discovery/registration functions at the SG level, not at the logically centralized cloud as typically offered in 5G telco infrastructures. The final goal is always to keep track of all devices and services available in the surrounding, where the surroundings are generally defined in terms of a single locality (multiple adjacent federated localities can be anyway considered as well as a suitable scope in some application domains).

FF6. *Execution of resource-intensive applications*. Similarly to the MEC server, the fog layer is also useful to support the execution of applications with low latency and high energy consumption requirements. However, in fog, the support solution is typically tailored for a specific class of sensing applications (rather than multimedia service-oriented as in MEC).

FF7. *Service Handoff*. This facility allows the transparent service rebinding of end user devices as they move from one (old) SG to another (new) one, possibly while maintaining continuity of service provisioning. Let us note that this function is not available in MEC nodes because 5G/LTE handoff management is completely on behalf of other inner infrastructure components, thus becoming transparent for the pure MEC functionality.

FF8. *Mobile offloading*. This support function in fog has the goal of granting low latency and short reaction time; moreover, due to the poorer SG resources, the decision about the splitting of the application between the mobile node and the SG is typically rather static and proactively taken at application deployment time. At the same time, mobile offloading in fog is typically intertwined with service handoff due to the more dynamic and decentralized nature of fog environments.

Moreover, at the basis of our 5GEE architecture, we have the idea of properly splitting support functionalities in order to make them more easily and efficiently usable by final user applications; this is a relatively new idea in MEC–fog computing integrated scenarios. In particular, we propose to put a service layer on top of all proposed support functionalities; this also means that we need to dynamically manage such services in order to link them with the used and composed functionalities at the lower layer. Some works and researches have just started to go into this direction, such as OMEC (Open Mobile Edge Cloud) [10], an open cloud platform, developed by AT&T, that uses some end-user clients and located at the "mobile edge" to carry out about functionalities on the edge.

IV. The 5GEE Architecture: Model and Design

A. *Novel 5G – Enabled Integration*

Figure 2 presents the general architecture of our proposal. We aim to merge the MEC and fog architectures into a single infrastructure with the combined support features. Our architecture consists of multiple 5GEE nodes, which expose and allow to combine MEC and fog functionalities.

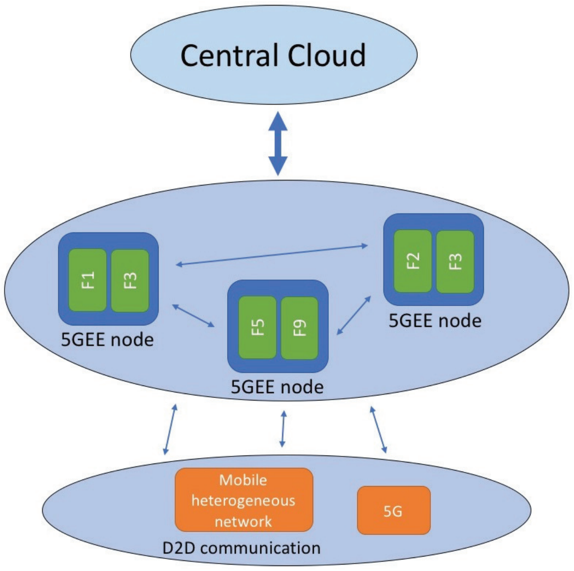

Figure 2. General architecture of our proposed 5GEE integration.

As described above, some MEC/fog functions are similar in existing proposals. In our work, we decide to smartly combine functionalities by minimizing overlapping features and by exploiting synergies. In our architecture, we include the following 5GEE features explained in Table 1.

Table 1. 5GEE functions and motivations.

Function	Motivation
MF1 and FF6: **F1 - Execution of resource-intensive applications**	Both functionalities help devices to execute resource-intensive application. Despite MEC limitations our architecture is able to run third-party applications.
MF2 and FF3: **F2 – Context data service**	MEC has network-based information about context, while fog solutions tend to have better information closer to edge. We combine these aspects in order to obtain an All-Context-aware support service.
MF3 and MF4: **F3 – Pre-processing and caching of multimedia contents**	MEC platforms are more multimedia-oriented than in fog solutions. Our 5GEE proposal combines all functionalities from the two paradigms in order to help devices to achieve better quality, especially for multimedia services.
MF5: **F4 – Resource management**	This is a central functionality for enabling efficient handoffs, in particular in scenarios of service continuity. Proper management of resource slicing is a needed enabler for allowing runtime service migration between 5GEE nodes.
MF6 and FF8: **F5 – Service offloading**	We combine together these two different functionalities about service offloading. The result is that the derived service offloading feature can move processing either to the 5GEE layer and/or to the global logically centralized cloud.
FF1: **F6 – Mobile Heterogeneous Network**	This feature helps our architecture to communicate with all set of dynamically discovered devices, which may exploit different wireless connectivity and discovery protocols.
FF2: **F7 – D2D communication support**	D2D communication is recognized as one of the emerging technology of central interest for increasing the scalability of 5G architectures. In our solution, the integration of D2D mechanisms and protocols is also the key towards locality identification and detection, as well as for enabling localized communications with no load over the MEC/fog-to-cloud trunk.
FF4: **F8 – Data storage and aggregation**	See the FF4 description above.
FF5: **F9 – Discovery / Registry of service**	The addition of discovery/registry facilities for available resources and services in a locality is of central relevance, even if not typically covered by MEC/fog solutions at this stage of their implementation evolution. Of course, this kind of support goes into the direction of leveraging locality-based interactions that are

	autonomous with regards to the central cloud. Integration of existing and heterogeneous discovery protocols is envisioned as appropriate to facilitate the path towards adoption.
FF7: **F10 - Handoff**	This is a crucial functionality for our architecture, given our specific interest in the support of mobile IoT applications. In order to guarantee continuity of service, the envisioned feature has to exploit predictive mechanisms (also based on profiling), proactive management of involved resources, and live but lightweight migration.

B. Management & Deployment Issues

By focusing on the management and deployment of our integrated architecture features, we foresee that services and network functionalities will be mainly implemented as software components executing on standard operating systems by leveraging the latest achievements in these fields. Network Function Virtualization (NFV) and Software Defined Networking (SDN) are basic enablers for the new scenario by simplifying the way network resources and functionalities are deployed and controlled across distributed locations. NFV [11] transforms Network Functions into Virtual Network Functions. Otherwise, SDN [12] simplifies the control plane of network elements by extracting it from the physical devices, and providing abstracted APIs for on demand control of the network. Moreover, in order to manage services and functionalities, we need a service orchestrator to take over the deployment complexity across the whole infrastructure. Many recent research activities are in this direction: in fact, it starts to be recognized that having an orchestrator allows architectures to overcome deploy challenges. For instance, in our architecture, we plan to use open source MANO (Management and Orchestration [6]). MANO is the ETSI-defined framework for the management and orchestration of all resources. This includes computing, networking, storage, and virtual machine (VM) resources. The main focus of MANO is to allow flexible on-boarding and sidestep the chaos that can be associated with rapid spin up of network components. MANO is composed by three main functional blocks: i) NFV Orchestrator, responsible for on-boarding of new network services (NS) and virtual network function (NFV) packages; ii) NFV Manager, oversees lifecycle management of NFV instances; iii) Virtualized Infrastructure Manager (VIM), controls and manages the NFV compute, storage, and network resources. Furthermore, we plan to use a particular solution based on MANO framework called OpenBaton; OpenBaton extends the existing standard specification of ETSI MANO in order to be able to properly manage also MEC applications and to use container deployment tools [7].

C. A Possible Implementation Blueprint

Here, we describe how we plan to implement our architecture. A possible implementation is designed according to a three-layer architecture, based on the extension of emerging MEC 5G - Fog technologies, by integrating proactive and/or reactive container migration. The three considered layers are:

- Mobile devices layer: Our mobile device layer consists of all the endpoints that need to perform high-resource demanding executions of mobile services and do not have enough capabilities to do that. For instance, this includes heavy image and video analysis/processing that can be performed uniquely with computationally intensive techniques that require resources beyond the capability of mobile devices, at least taking into consideration possible application-specific requirements on response time. Our solution fits a very wide spectrum of heterogeneous mobile devices, with the only constraint to run Android OS (in the future we plan to extend the approach to other relevant OS, such as iOS). Generally, mobile devices often do not have enough capabilities to satisfy strict requirements on response time, in particular if considering their possible immersion in hostile environments; therefore, it is a must that they have to delegate most analysis functions to the 5GEE layer.

- 5GEE layer: Our 5GEE layer consist of two primary components: i) the standard OpenStack platform [13], one of the current most promising and complete open-source project to realize the middleware layer, that allows moving computational resources near the edge of the network, by providing and orchestrating containers near mobile devices; and ii) a service orchestrator based on OpenBaton that provides containers deployment of services and functionalities. The 5GEE layer run an OpenStack instance, while services and functionalities are implemented as containers in execution on it. We propose this option because containers can relevantly reduce resource computation on the edge, because container-oriented solutions start to be supported by OpenBaton (also thanks to our joint collaboration and contribution), and containers are more easily movable across our 5GEE infrastructure. In particular, we have selected Docker as the container solution for our implementation. Our 5GEE nodes can be provisioned in either a proactive or a reactive way. Proactive provisioning allows to minimize and automate in an efficient way virtualized function migration, by pre-loading the needed functions in advance on the target 5GEE node that, presumably, will be the next visited by the served user; this is managed before receiving explicit migration requests, thus limiting the costs in terms of unavailability and performance during the procedures of mobile device handoffs and associated container data volume. The central cloud layer (third layer below) triggers virtualized function migration. On the contrary, reactive migration is triggered when a mobile device requests explicitly to move a virtualized function.

- Central cloud layer: The cloud layer is used to assist the 5GEE intermediate middleware; in the current ongoing implementation, this assistance is mainly focused on proactive analysis about users' movements. In addition, the cloud layer is used during the initial service setup operations and interacts with a service orchestrator in order to load the needed services and functionalities over the interested subset of 5GEE nodes, by need at service provisioning time.

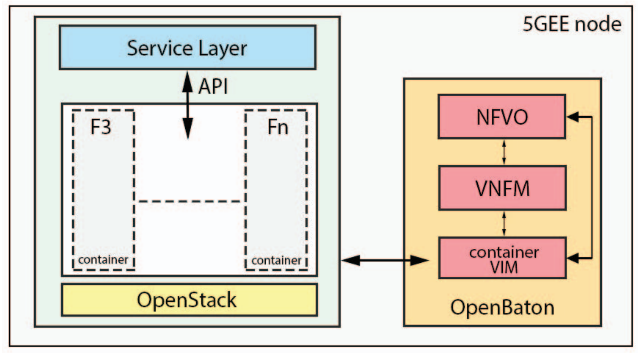

Figure 3. Implementation blueprint of our 5GEE architecture.

V. 5GEE-ENABLED USE CASES AND DISCUSSION

In this section, we practically show that the previously described features may be useful in two main use cases, and then we report a discussion of open technical challenges and most promising research directions. The two use cases both relate to Mobile Crowd Sensing (MCS).

A. Use Cases

Use case 1: Alice participates to an MCS project (UNIBO has its own crowdsensing project called ParticipAct [8]). The increasing popularity of smartphones, already equipped with multiple sensors from GPS to microphone and camera, paired with the inherent mobility of their owners, enables the ability to acquire local knowledge from the individual's surrounding environment through mobile device sensing properties or even from the individual itself. Participants collect passive and active data that are processed and shared with other participants and with the central cloud. Back to Alice, assume that the connectivity in her city is provided by a new 5GEE node. Normally, the 5GEE node is set to provide telco functionalities such as LTE/5G connectivity and a few additional network support functionalities. In a particular day, when a new MCS campaign starts, the available functionalities over 5GEE nodes in given localities have to be updated in order to support the campaign. This is done by an orchestrator, such as MANO, that dynamically loads the needed functionalities as overlays on top of the already installed basis (i.e., base image plus previous possible overlays). Which support features does Alice need? Analyzing the scenario, Alice has to use her smartphone to collect data, process them, and then upload their aggregated summary to the central cloud. So, she would benefit from F8 (store and aggregate data functionality) as well as from F2 (context data service functionality) in order to perform context-aware campaign tasks. These are just the basic functionalities that an MCS campaign needs. But we may also need an F7 feature (D2D communication support) for collaborative crowdsensing, or F10 to effectively support device mobility. Those support features help to overcome many cloud computing problems and helps Alice to safeguard power consuming of her smartphone and achieve a better user-perceived end-to-end final quality. Figure 4 depicts how the proposed 5GEE architecture works in a crowdsensing context.

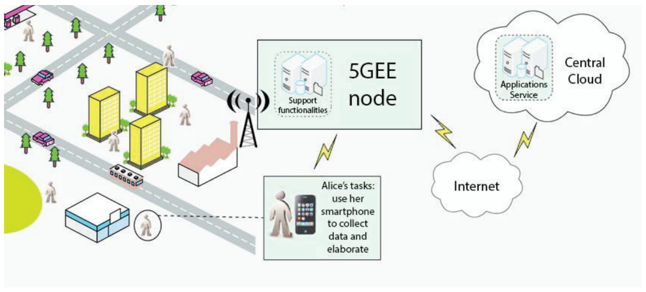

Figure 4. 5GEE in an MCS monitoring scenario.

However, in a classic Smart City scenario composed by sensors and actuators, we need to maintain a given quality level, for example to ensure low latency response time for sensor configuration operations and/or for control commands over actuators. The support of quality requirements is with no doubt still an open technical challenge in the field; however, solutions based on container-oriented virtualization can help to reduce latency time: for instance, some few seminal works have started to exploit Docker containers in such IoT scenarios [14, 15].

Use case 2: Instead, let us suppose that, in order to complete his MCS task, Bob has to share video streaming to other people to show what happened in his zone (see Figure 5). In the last past years, mobile video streaming traffic has increased considerably and represents a relevant bottleneck for mobile traffic data. To overcome this challenge, edge caching has been identified as a natural solution. Videos may be cached in the 5GEE node, so that demands from users can be accommodated easily without transmission from remote resources, e.g., on the global cloud. This approach contributes to reduce central cloud usage and content access delay.

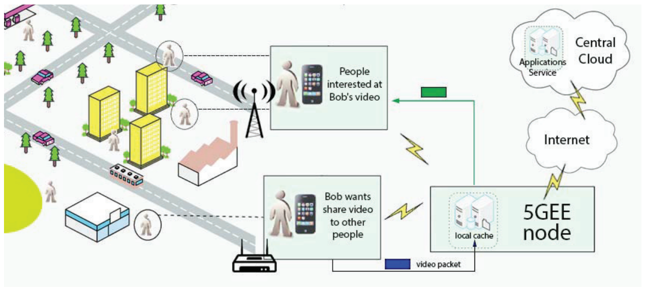

Figure 5. 5GEE in an MCS content sharing scenario.

Focusing on content caching and delivery techniques, several approaches for wireless networks have been presented in the past [16], but they are out of the scope of this paper. Back to Bob's problem, he wants to share his multimedia content with a neighbor node in his locality. As already mentioned, the connectivity in the city is provided by a new 5GEE node that normally has basic telco support features. In this case, which functionalities does Bob need for his active

MCS participation? We may probably benefit from i) F3 (pre-processing and caching of multimedia contents) in order to store video contents in a local cache and ii) F1 (execution of resource-intensive applications) to enable the execution of possibly heavy codecs that may be necessary to adapt video contents for a set of device renderers currently present in the locality. Figure 4 shows how our 5GEE architecture works in this kind of application scenario.

B. *Open Challenges and Research Directions*

As discussed above, smart and efficient component/service orchestrators are crucial enablers in such as context; their availability for converging MEC/fog computing is still a very open and technically challenging point. In particular, traditional service orchestrator approaches that have been applied to cloud computing (which abound in the related literature) are demonstrating to be unsuitable for the scenarios envisioned in this position paper. In fact, low latency, high mobility, high scalability, and real-time execution over decentralized localities call for new kind of orchestration supports, capable of supporting dynamic and decentralized autonomy of localities. Therefore, we claim the need for a new model of service orchestrator, i.e., Hybrid Environment Services Orchestrators (HESOs), that will be capable of ensuring the resilience and trustworthiness expected for open, large-scale, and dynamic services in the fog-MEC layer. New HESOs will be responsible for the composition of services available in fog-oriented (e.g., sensing, data aggregation, storage) and MEC-oriented environments (e.g., multimedia streaming and caching).

In our work of 5GEE architecture implementation, we intend to use container-oriented virtualization solutions as a service deployment. Here, the primary technical challenge is the support to interoperability, reliability, time-constraints, and security, which are considered crucial in smart industry application scenarios, where control sensors and actuators dynamically discover each other and coordinate to work together in a locality.

VI. CONCLUSIONS AND ONGOING WORK

In this position paper, we have presented 5GEE, a new architecture model to ease the provisioning and to extend the coverage of traditional edge computing approaches by bringing together the best of MEC and fog research areas. The cornerstone of our proposal lies in the ability to dynamically orchestrate all functions and needed resources at 5GEE nodes without assuming any predefined configuration. Rather, 5GEE leverages latest achievements in the two main areas of cloud computing and containerization, while exploiting MANO to automate and fasten the tailoring of 5GEE node installations according to the specific requirements of final users and currently supported applications. In addition, we are currently working on our ParticipAct living lab to exploit the benefits and potential of the 5GEE integrated architecture into it.

Those encouraging results are pushing us to further investigate and refine our 5GEE model and we are currently exploring various related areas. On the one hand, we are working to enable the self-adaptable fine tuning of our 5GEE middleware to the different dynamics and variations of the city pulse, for instance to the different behaviors that might present along the year, such as working vs. vacation periods, and the week, such as working days vs. weekends. On the other hand, we are investigating innovative techniques in order to reduce the latency of downloading Docker overlays on 5GEE nodes i) via parallelization of I/O and configuration operations and ii) via approximated heuristics to predict the need for specific middleware features over a locality.

ACKNOWLEDGMENT

This research was supported by the Sacher project (no. J32I16000120009) funded by the POR-FESR 2014-20 program through CIRI.

REFERENCES

[1] White Paper: ETSI's Mobile Edge Computing initiative explained https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf.

[2] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog Computing and Its Role in the Internet of Things", Proc. of the first edition of the workshop on *Mobile Cloud Computing (MCC 2012)*, ISBN: 978- 1-4503-1519-7, doi:10.1145/2342509.2342513, ACM, New York, NY, USA, August, 2012, pp. 13-16.

[3] OpenFog Consortium, 2017, http://www.openfogconsortium.org.

[4] S. Shubhranshu, C. Yen-Chang, T: Yi-Hsing, Y. Jen-Shun, "Mobile Edge Fog Computing in 5G Era: Architecture and Implementation", Proc. of *International Computer Symposium (ICS)*, 2016.

[5] M. Tao, K. Ota, M. Dong, "*Foud: Integrating Fog and Cloud for 5G-Enabled V2G Networks*," Proc. of *IEEE Network* vol. 31, no.2, March/April 2017.

[6] ETSI OSM Community White Paper: Open Source MANO: https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseTWO-FINAL.PDF.

[7] G. Carella, M. Pauls, T. Magedanz, M. Cilloni, P. Bellavista, L. Foschini, "Prototyping NFV-based Multi-access Edge Computing in 5G ready Networks with Open Baton" To be pubblised in Proc. of *3rd IEEE Conference on Network Softwarization (IEEE NetSoft 2017).*

[8] G. Cardone, A. Corradi, L. Foschini, R. Ianniello, "ParticipAct: A Large-Scale Crowdsensing Platform", Proc. of *IEEE Transactions on Emerging Topics in Computing* vol. 4, no. 1, Jan.-March 2016, pp. 21–32.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," Proc. of *IEEE journal on Pervasive Computing*, vol.8, no.4, pp.14-23, Oct.-Dec. 2009

[10] C. Buyukkoc, "Open Mobile Edge Cloud", Workshop of *IEEE SDN Initiative Pre-Industrial Committee*, Nov. 2015.

[11] White Paper: ETSI's NFV https://portal.etsi.org/NFV/NFV_White_Paper_5G.pdf.

[12] White Paper: ONF SDN: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[13] OpenStack, Available online at: https://www.openstack.org/.

[14] Docker, Available online at: https://www.docker.com/.

[15] Ruchika, "Evaluation of Docker for IoT Application", Proc. of *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 4 no. 6, June 2016, pp. 624–62.

[16] E. Bastug, M. Bennis, and M. Debbah, "Living on the Edge: The Role of Proactive Caching in 5G Wireless Networks," Proc. of *IEEE Commun. Mag.*, vol. 52, no. 8, Aug. 2014, pp. 82–89.