IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON MOBILE EDGE COMPUTING

Received December 1, 2017, accepted January 17, 2018, date of publication January 30, 2018, date of current version March 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2799707

Edge Computing Architecture for Mobile Crowdsensing

MARTINA MARJANOVIĆ, (Student Member, IEEE), ALEKSANDAR ANTONIĆ, (Student Member, IEEE), AND IVANA PODNAR ŽARKO, (Member, IEEE)

Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: Martina Marjanović (martina.marjanovic@fer.hr)

This work was supported in part by the Croatian Science Foundation (Human-centric Communications in Smart Networks) under Project 8065, and in part by the H2020 symbIoTe Project, through the European Union's Horizon 2020 Research and Innovation Programme under Grant 688156.

ABSTRACT Mobile crowdsensing (MCS) is a human-driven Internet of Things service empowering citizens to observe the phenomena of individual, community, or even societal value by sharing sensor data about their environment while on the move. Typical MCS service implementations utilize cloud-based centralized architectures, which consume a lot of computational resources and generate significant network traffic, both in mobile networks and toward cloud-based MCS services. Mobile edge computing (MEC) is a natural choice to distribute MCS solutions by moving computation to network edge, since an MEC-based architecture enables significant performance improvements due to the partitioning of problem space based on location, where real-time data processing and aggregation is performed close to data sources. This in turn reduces the associated traffic in mobile core and will facilitate MCS deployments of massive scale. This paper proposes an edge computing architecture adequate for massive scale MCS services by placing key MCS features within the reference MEC architecture. In addition to improved performance, the proposed architecture decreases privacy threats and permits citizens to control the flow of contributed sensor data. It is adequate for both data analytics and real-time MCS scenarios, in line with the 5G vision to integrate a huge number of devices and enable innovative applications requiring low network latency. Our analysis of service overhead introduced by distributed architecture and service reconfiguration at network edge performed on real user traces shows that this overhead is controllable and small compared with the aforementioned benefits. When enhanced by interoperability concepts, the proposed architecture creates an environment for the establishment of an MCS marketplace for bartering and trading of both raw sensor data and aggregated/processed information.

INDEX TERMS Mobile crowdsensing, mobile edge computing, MCS functional architecture, MEC reference architecture.

I. INTRODUCTION

MOBILE crowdsensing (MCS) is a human-driven activity which leverages the pervasiveness of wireless connectivity and various mobile devices with built-in sensing capabilities as well as the inherent user mobility to create dense and dynamic data sets characterizing our environments. In other words, MCS benefits from a large user base (i.e., ''the power of the crowd'') and its potential to become a rich source of information about various phenomena in our environment [1]. A typical MCS scenario involves users carrying their smartphones with applications running in the background to continuously collect sensor readings, either from built-in sensors or wearables. Such data acquisition activity requires minimal user involvement and is named *opportunistic sensing* in literature, in contrast to *participatory sensing* which requires active user involvement to create sensor readings [2]. The generated sensor data is preprocessed on mobile devices and subsequently sent to cloud servers running an MCS service. The service can potentially receive huge data volumes from a large user base and generate significant load on mobile network. The value of MCS data is twofold: 1) *offline usage*: huge data sets serve as input for big data analytics to uncover hidden information about sensed phenomena in areas which are otherwise difficult to cover with dense spatio-temporal measurements, and 2) *real-time usage*: continuous processing of incoming data streams

creates context-aware notifications and alerts, both for citizens and public administration, to be sent from the cloud to user mobile devices.

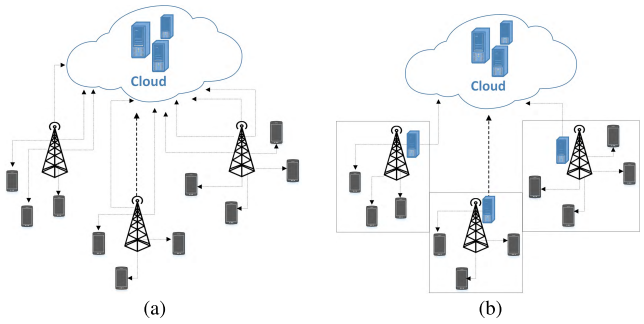

FIGURE 1. Mobile crowdsensing deployments. (a) Cloud-based MCS architecture. (b) MCS architecture powered by MEC infrastructure.

Typical MCS service implementations are cloud-based and centralized [3]–[5], as depicted in Figure 1(a). This architecture consumes a lot of resources since many concurrent connections to back-end cloud servers are needed for near real-time processing and storage of incoming data sets. Moreover, cloud services perform device management functions: They coordinate the sensing tasks on many user devices and keep track of device context to choose the best data sources for defined sensing tasks. Both device management and real-time data processing require significant computational resources in case of large-scale MCS deployments. In addition, user movements and frequent context changes can quickly make information obsolete, which requires efficient real-time processing of raw data to produce context-based information followed by notification delivery with low propagation latency in real-time usage scenarios. Large-scale and centralized MCS introduces the following problems:

- generates significant load on mobile network radio and backhaul,
- creates an increased traffic to cloud servers running MCS services,
- it is computationally expensive, especially for real-time usage scenarios, due to a large number of devices participating in MCS tasks with frequently changing context,
- increases the latency of data and information propagation, which is critical for real-time usage scenarios, and
- represents a threat to user privacy since all user traces are collected in a centralized manner.

Mobile Edge Computing (MEC)[1] is designed to enable third parties to run their services and applications at the edge of mobile networks, and is promoted as an enabler of advanced IoT services of massive scale, such as MCS, that would not be technically or economically feasible before the launch of 5G-like networks [6]. MEC moves computation in the proximity of mobile devices by introducing a new intermediate layer responsible for data filtering, aggregation, processing and storage [7]: In the context of MCS it is used to process MCS data between mobile devices and cloud services, as shown in Figure 1(b). Therefore, in addition to preprocessing of raw sensor data on mobile devices, edge resources can be used for processing/aggregation of data streams contributed by a subset of users involved in MCS tasks and can even support real-time usage scenarios autonomously without the need to contact cloud services.

The main benefits of MEC in the context of MCS are the following:

- parallelization and partitioning of problem space based on location, where MEC servers naturally take the responsibility for controlling the sensing process on mobile devices located within their deployment area and manage MCS tasks within the same area,
- computation offloading from both mobile devices and cloud servers, since certain data processing tasks can be performed at the edge,
- reduced computational complexity of centralized cloud services, which in turn increases the complexity of service orchestration,
- decreased latency when disseminating notifications and alerts to mobile devices in case of real-time MCS usage scenarios, and
- reduced privacy threats since sensitive data is partitioned and distributed across MEC servers.

MEC reference architecture perfectly matches the requirements of massive-scale MCS solutions since the generated sensor data streams are always handled by the closest MEC service, in line with submitted MCS tasks relevant to the covered location area. Although MEC offers limited computational and storage resources at the edge, this does not represent a big disadvantage since a single MEC service is serving a limited geographic area. These advantages have already been identified in the early works [8], [9] which introduce an edge/fog layer responsible for edge analytics and local data storage. Another relevant approach is presented in [10]: It proposes an IoT enabled MCS framework based on the oneM2M standard architecture to be deployed at any computing platform including cloud, edge or even M2M gateways.

In this paper we present a hierarchical MCS model adequate for edge computing environments that allows users to control the contributed sensor data by means of the *announce/subscribe/publish* communication mechanism, which also manages the propagation of MCS tasks to devices capable to address those tasks. Furthermore, we analyze the features characterizing MCS services to propose a layered functional architecture adequate for MEC environments. The architecture identifies a minimal set of features to be placed at the edge computing resources in order to satisfy the requirements of future massive-scale MCS deployments. Those features are put into the context of the MEC reference architecture proposed by ETSI [11].

[1]ETSI recently renamed its corresponding Industry Specification Group to Multi-access Edge Computing.

A distributed architecture powered by MEC introduces potentially high overhead due to the need to orchestrate MCS services at network edge and reconfigure them over time. This is the result of moving patterns of the users participating in MCS tasks. Thus, in this paper we investigate the aforementioned overhead by using a real data set of user traces collected in South Korea [12]. Our analysis shows that a relatively small number of MCS services needs to be deployed at network edge if their reconfiguration period is reasonably small (15 to 60 minutes). This shows that the proposed architecture introduces small and controllable overhead and is indeed adequate to be deployed in a MEC environment.

The proposed solution creates an environment for a future MCS marketplace where both crowd-sensed and aggregated/processed data is used as commodity across various domains. Such future marketplace requires the deployment of interoperable MCS services, as those built by the following research projects that devise interoperability solutions for IoT environments: symbIoTe[2] and BIG-IoT.[3] In this paper we assume that MCS services across different domains are interoperable, i.e., they expose unified interfaces, use open communication protocols and standardized information models, but do not provide further design details regarding interoperability as it is a complex problem per se. An interested reader can find further information in [13] and [14].

The remainder of this article is organized as follows. First, in Section II we showcase an example use case depicting the usage of existing and future MCS services to motivate the need for massive-scale and interoperable solutions. We provide an overview of the MCS paradigm and corresponding communication model in Section III. Section IV identifies the characteristic features of MCS services and incorporates them in a functional MCS architecture which we position within the MEC reference architecture in Section IV-C. An evaluation of the proposed solution is included in Section V. We discuss challenges, open issues, and opportunities which the MEC technology brings to MCS in Section VI. Furthermore, we briefly survey related work addressing hierarchical fog and mobile edge computing solutions in Section VII, and conclude the paper in Section VIII.

II. SMART NEIGHBORHOOD USE CASE

To motivate further discussion, we present a Smart Neighborhood use case enabling citizen collaboration for better quality of life in urban districts. Let us consider a scenario in which a family moves to a new neighborhood and wants to actively engage in community life.

John is a 35-year old businessman who has recently been promoted. As required by his new position, John and his family need to move to another city and are looking for an adequate apartment. Since they are big fans of nature and sports, while they also like theater and cultural events, it is vital to find an adequate neighborhood with lots of parks and a vivid cultural life. Before renting an apartment, John and his wife Jane are checking candidate districts which have active citizen communities that collaborate through MCS services. Citizens actively engage in MCS tasks by contributing, e.g., environmental data and ratings of cultural events through their Smart Neighborhood MCS marketplace. In turn they can create MCS tasks on-the-fly, e.g., to identify unpolluted jogging routes or to find the closest available bicycle which is shared by the community. Thus, John and Jane decide to use the marketplace to get to know their future neighborhood.

It is vital for Jane that the neighborhood is quiet at night, while the air quality is in general good so the family can spend a lot of time outdoors. John also wants to check the traffic density and availability of parking places. In addition, they both want to check the frequency of nearby cultural events and impressions shared by their future neighbors. Therefore, John and Jane browse the marketplace to find out that parking and traffic MCS tasks are actively used in the new neighborhood, so John joins the service and accepts to contribute his location while commuting. In return he will be able to check traffic density and receive statistics about available parking spots in the new neighborhood. Jane subscribes to receive updates about cultural events, such as poetry nights and plays in a nearby theater. Since John finds out that the MCS marketplace does not host a service tracking noise and air quality, he decides to create a new request for checking those parameters. The marketplace accepts a new MCS task and discovers citizens with adequate devices to monitor noise and air quality. When a sufficient number of citizens accepts the new monitoring task, a new MCS service for environmental monitoring is deployed in the new neighborhood. The community is engaged in a new task to collect environmental parameters also serving as a data source for other citizens. Citizens are motivated to participate in MCS tasks because they benefit from community-generated data of their interest. In addition they also enjoy in special benefits from a local board and sometimes receive discounts at a local green market for their community engagement. John and Jane start to receive updates and statistics about their new neighborhood and decide it is appropriate for their family. They have in the mean time collected a lot of sensor data for their current local community and shared this valuable information with its residents.

In the context of MCS, John and Jane are data prosumers: Every time when they express interest in traffic conditions or cultural events, they create/join a *task* in the Smart Neighborhood MCS marketplace, and become task *requesters* for their new neighborhood. This interaction is depicted in Figure 2 where John and Jane use the MCS marketplace to define and deploy tasks and consume data contributed by other citizens to the marketplace. Additionally, they serve as *data sources* for their current local community, thus taking the role of *workers* collecting sensor data for tasks defined in their old neighborhood. This is an example

[2]https://www.symbiote-h2020.eu/
[3]http://big-iot.eu/

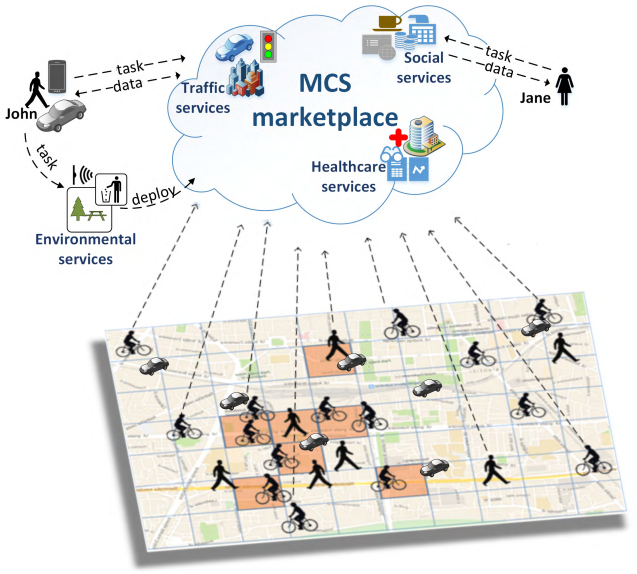

FIGURE 2. Smart Neighborhood use case enabled by MCS marketplace.

how MCS data becomes a commodity for sharing within a marketplace, but other usage examples are also possible.

It is quite straightforward to imagine a large number of similar scenarios where bartering of citizen-generated data can help communities to grow and engage in social activities. Specific MCS tasks can for example target various vulnerable groups, from asthma patients to elderly people. In this example the technology is used to motivate a positive human engagement and community building, which leads to improved quality of life in big cities.

III. MOBILE CROWDSENSING

An MCS marketplace involves *requesters* and *workers*, who interact through an MCS service. A communication flow starts from a requester creating a new task T which is sent to the MCS service responsible for finding at least one eligible worker willing to participate in data acquisition for this task. Each task is specified by associated requirements (location area, needed sensors, incentives, etc.) which have to be fulfilled within a certain time period before the task expires. If there is a positive match between workers' capabilities (i.e., type of available sensors) and task requirements, the MCS service disseminates the task to all potential workers who can accept it and start the sensing process. The produced data is delivered to the MCS service, which in turn notifies the task requester to optionally express the level of satisfaction with the quality of collected data which may directly affect the reputation of a worker responding to the task.

An MCS service has to control the data production process since mobile devices typically support only limited filtering and aggregation mechanisms and often deliver all raw readings to the cloud. Furthermore, due to user mobility patterns, potentially large amounts of redundant data can be generated in certain areas, while other areas may suffer from lack of available data sources, which complicates both the data acquisition and device management process, as further investigated in [15]. In traditional MCS architectures, all existing tasks are maintained by a centralized MCS cloud service which is responsible for the data acquisition management. Since such service has to keep track of all active tasks and processing of incoming data sets, its real-time performance can become rather inefficient, precluding the real-time usage of MCS services.

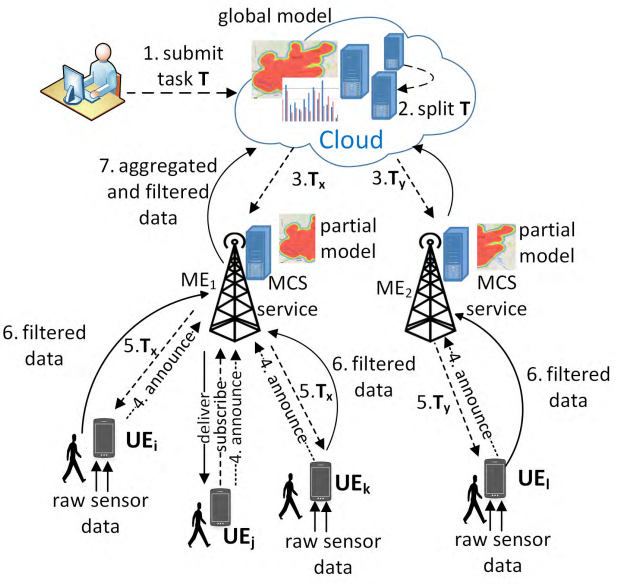

FIGURE 3. Communication flow in an MCS ecosystem deployed on the MEC infrastructure.

The MEC paradigm has the power to reduce the load of MCS data processing and service execution by distributing MCS tasks to multiple Mobile Edge (ME) servers deployed at network edge. ME servers become responsible for a subset of tasks and workers in their close vicinity, as shown in Figure 3. When a new task T (e.g., for monitoring air quality in a specific region) is submitted to the system (1), the cloud server splits T in two subtasks T_x and T_y (2), based on required location areas, and forwards them to the corresponding ME servers (3). An MCS service running on the ME server is aware of all potential workers in its area and their capabilities, since workers *announce* (4) the type of data they can and are willing to measure with their user equipment (UE) when entering the area under the responsibility of an MCS service. Thus, only tasks which can be potentially executed by workers are forwarded to their UEs, which prevents device overload, in accordance with the publish/subscribe messaging paradigm that proves to be suitable for MCS ecosystems, as we have already shown in [3]. For example, in Figure 3 the MCS service deployed on ME_1 sends task T_x only to workers i and k with equipment UE_i and UE_k (5). Workers start to produce raw sensor data, which is first filtered on their mobile devices and subsequently delivered to the MCS service running on ME_1 (6). Since ME servers have substantial computational power compared to

mobile devices, they can perform data aggregation and processing of all data produced by workers under their responsibility. Thus, an MCS service builds a *partial data model* for its area and sends the aggregated model to the cloud (7). The MCS cloud server subsequently creates a *global data model* based on partial models received from multiple ME servers and performs different analytics to extract knowledge about the sensed phenomena for a larger area. Additionally, it can also correlate different sensor data. In case of real-time MCS usage, users who express interest in the data produced in their close vicinity can receive alerts with low propagation delay from their edge MCS service. For example, user *j* (in Figure 3) sends a *subscribe* message to the MCS service on ME_1 server to express his/her interest in the air quality for the area in which he/she currently resides. If the MCS service posses data regarding air quality, it *delivers* real-time notifications to the user *j*, as long as the user is interested in the data. Note that an event *subscribe* can happen at any time, while *deliver* occurs when an MCS service deployed on the ME server finds a positive match between user subscription and published data.

A wide range of MCS applications have already been used across different domains, from environmental and transportation monitoring, to healthcare and social services. Hereafter we provide a brief overview of MCS domains and identify the most important benefits which MEC infrastructure brings to such services.

A. ENVIRONMENTAL MONITORING

MCS applications are used in combination with wearable sensors to monitor environmental conditions (air quality, noise detection, waste management, water pollution) over large geographical areas [3], [16]. The goal is to collect dense spatio-temporal measurements, which is hard to achieve with traditional equipment or static meteorological stations. The MCS architecture supported by MEC facilitates coordination between multiple devices within the same location area and reduces the communication latency so that correlated measurements can be collected in order to perform device calibration and cooperation on-the-fly for collocated devices.

B. TRAFFIC MONITORING

Since MCS applications do not require special infrastructure and potentially involve a large number of citizens, they can be used to estimate traffic conditions [17]. Typically, they are used to detect road congestions and damages, monitor traffic conditions, find available parking spots, identify citizen mobility patterns, etc. The MEC technology offers support for localized aggregation and filtering of redundant data which is important for traffic services where numerous users are concurrently publishing large amounts of potentially redundant information.

C. HEALTHCARE

The proliferation of wearables has created potential for novel applications addressing healthcare and wellbeing monitoring. Applications are person-centric and presume active involvement of users in the sensing process. However, their usage on community wide level during longer time periods can encourage citizens to engage more in physical activity and help experts to recognize specific patterns between different symptoms and living habits [2]. A major benefit of MEC deployments is related to privacy issues since user data is distributed across multiple ME servers.

D. SOCIAL SERVICES

This domain involves different MCS applications which rely on user's subjective opinion and personally sensed information, from social recommendations and activity suggestions, to disaster management and crime prevention [18]. Typically, such services require quick and location-aware information exchange which can be enabled through broadcast mechanisms supported by the MEC technology.

IV. MCS ARCHITECTURE ENABLED BY MEC

By leveraging edge processing resources for the design of future MCS services, we can identify a paradigm shift affecting the placement of MCS-related features across the new hierarchical and distributed architecture, where certain features are now deployed across multiple MEC sites, while being coordinated from the cloud. In this section we identify the characteristic features of MCS services which serve as input for the design of a four-layered functional MCS architecture enabled by the MEC technology. Furthermore, we put the features placed at the edge layer in the context of the MEC reference architecture defined by ETSI [11].

A. MCS CHARACTERISTIC FEATURES

To model a functional architecture suitable for MCS, while exploiting the full potential of MEC technology, we identify a characteristic set of MCS features focusing on these that benefit from functionalities natively provided by the MEC technology.

Data (pre)processing refers to the processing performed close to sensor data production place, mostly to reduce the load on edge and application servers as well as to save smartphone resources [19]. An MCS service performs basic preprocessing of raw data on a smartphone before transmitting data further to the cloud, e.g., time-series processing techniques can be used to reduce the size of the data that is communicated. An ME server can aggregate received data streams before sending a final result to the cloud for a long-term storage. Such incremental and hierarchical processing is adequate to reduce network traffic and save energy on user devices, thus increasing MCS service performance.

Sampling mode distinguishes *continuous sampling* which enables monitoring of a value during its lifetime, and *triggered sampling* which indicates the occurrence of a significant event. Since the ME technology can infer user availability based on network traffic, this information can be used to control the data acquisition process on UE (i.e., continuous sampling can be triggered in the background when

appropriate, while the user is occupied with another activity). Of course, it is vital to have user consent for such scenarios.

Sensing scope describes outreach of an MCS service, grouping it in three categories: personal, group and community [2]. The personal scope is focused on individuals, the group scope involves more individuals who share a common goal, while the community scope indicates that a topic is of interest to a wide-scale population. Both group and community dissemination scope, which is location-aware, are benefiting from the MEC technology, since an ME server possesses knowledge about users grouped within the same geographical location.

Device discovery is one of the mandatory non-functional requirements because an MCS service needs to efficiently match a task with the best workers. A network provider running an ME host has the knowledge about available UEs which are candidates to fulfill an active MCS task and can choose the most appropriate workers based on their location, capabilities and reputation.

Mobility support relates to handling of a large user base and frequent location changes. The functionality is vital to detect patterns of user movement, with emphasis on identifying dense and sparsely populated areas, so that an MCS service can devise measures to efficiently handle both cases [20].

Energy management is responsible to minimize energy consumption by optimizing the set of UEs that are needed for MCS tasks, while others are on stand-by until their involvement is required. Smart energy management ensures that a user is more willing to be involved in MCS tasks, while the final goal is to increase task accomplishments and to satisfy the required sensing quality levels. Since an ME host is responsible only for managing a subset of devices, it can perform optimizations on a smaller scale with better results compared to the cloud approach which optimizes energy consumption on a global scale.

Data analytics and storage is the functionality responsible for handling off-line data processing, and thus provides deeper insight in the data collected by users. Primarily, it is used to create aggregates of historical data and to perform demanding processing operations, such as statistical analysis, data interpolation and cross-domain data correlation. It is expected that this functionality is performed on cloud servers.

User (and sensor node) reputation provides a measurement of trustworthiness related to a contributed data set [21] which is vital for the quality of contributed data, since the service is driven by data collected from sources that are not known *a priori*. There is no single method to calculate reputation because each service adapts the reputation mechanism to its particular needs [22]. In MEC environments, novel distributed mechanisms for calculating trustworthiness, which should not jeopardize user privacy, are needed.

Real-time processing is essential for MCS services, both in terms of real-time data processing and information dissemination to interested parties. Support for such high reactivity is expected because of a highly dynamic environment, which involves numerous users frequently changing their context, while the service needs to send only up-to-date information [19]. The MEC infrastructure can fulfill ultra-low latency requirements (less than 1 ms) since it enables hosting of MCS services at the network edge and therefore supports data processing close to the users. It provides the shortest path between the application and MCS service to significantly reduce the information retrieval time [6].

Communication protocol can be used to distinguish two groups of devices based on their communication capabilities. Resource constrained devices without IP-connectivity use one of (wireless) close proximity protocols, such as Bluetooth or ZigBee, to communicate with a smartphone which serves as a gateway [23]. Devices with IP-connectivity use one of protocols suited for IoT and MCS environments, such as CoAP or MQTT. The MEC can act as a gateway between devices which interact through diverse communication technologies serving as an aggregation point towards the cloud layer [7].

Security and privacy is a key requirement since users contribute their location data or data enriched with sensitive contextual information, and thus users would like to ensure that sensitive data is not subjected to misuse [24]. Approaches to address this requirement span from anonymization to end-to-end data protection along the communication path. MEC technology can potentially improve security and privacy protection of end users, since there is no need to forward user-sensitive data in its original form to the cloud.

In addition to the previously listed features, MCS services also possess domain-specific characteristics which relate to the type of sensing nodes, user involvement (opportunistic or participatory), incentives motivating users for active involvement, and quality of service measuring overall user satisfaction with an MCS service. Since the previously mentioned features do not have significant influence on MCS architecture, they are not further discussed in detail.

B. FUNCTIONAL MCS ARCHITECTURE

Hereafter we present the four-layered functional architecture for MCS derived from the key MEC properties and previously identified MCS features, as shown in Figure 4:

User Equipment comprises both wearable sensors and smartphones with different sensing capabilities, as well as human-driven sensing, where incentive mechanisms have great impact on participant's involvement in MCS tasks. Initial data processing is typically performed on a smartphone to reduce bandwidth and energy consumption before sensor data is transmitted to an edge server.

Edge Computing layer is located in close vicinity of users, between physical sensing devices and the cloud. It is responsible for the management of workers in its location area, data gathering from active workers, processing and filtering of sensor data, as well as data aggregation. Available information about UEs on ME servers can be reused by MCS service to infer user context, detect new available workers,

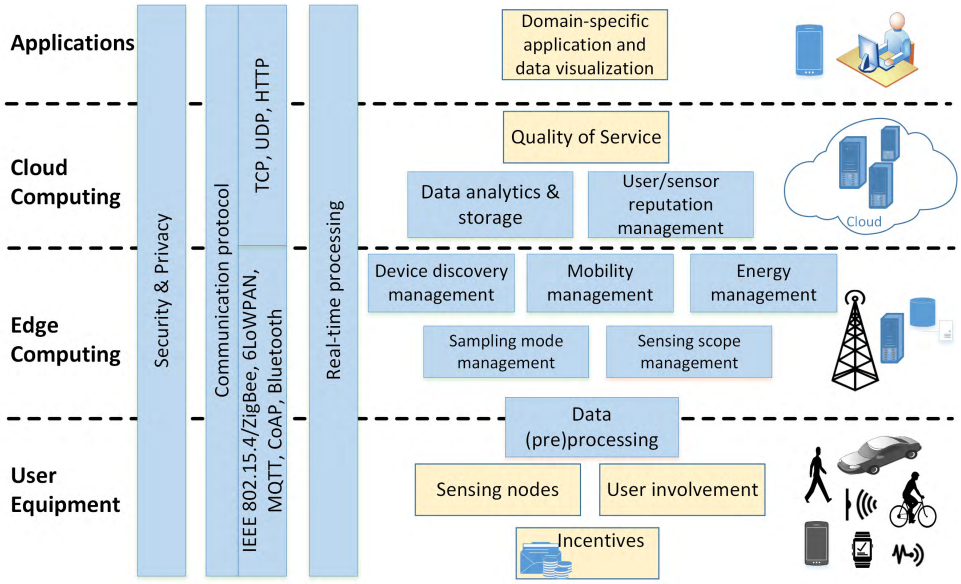

FIGURE 4. The four-layered functional architecture for MCS.

monitor user mobility, and to push notifications created at the edge to the UE with low latency.

Cloud Computing layer offers resources for complex data analysis and long-term storage. Furthermore, this layer enables interactions between multiple MCS services, including their collaboration and data exchange.

Applications are developed on top of the cloud services and provided to requesters specifying MCS tasks who want to analyze results of data analysis performed on top of collected data sets. They are domain-specific and may include different interfaces which enable data visualization and knowledge sharing among users, both via web and mobile applications with real-time user notifications.

Security and privacy, support for diverse communication protocols and real-time processing are generic features needed across all architectural layers to ensure data protection during the execution of an MCS service, and to enable interactions and data dissemination between multiple functional components.

C. ADAPTING THE MEC REFERENCE ARCHITECTURE TO MCS SERVICES

In this subsection we position MCS functional components within the MEC reference architecture (RA). More specifically, we focus only on the functionalities that we have identified as part of the Edge Computing layer in Section IV-B, and place them in the MEC RA. Other functionalities are placed either above MEC infrastructure in the cloud layer responsible for complex data analysis and long-term storage, or below MEC infrastructure in UE layer which contains different data sources, and are not further discussed in detail.

MEC RA identifies the functional entities at the host and system level, and defines reference points enabling communication between those entities. Different MCS services can be deployed on an ME host as ME applications, in accordance with MEC RA (see Figure 5). The ME host runs the ME platform and the Virtualization Infrastructure to provide computing, storage, and network resources for the ME applications. The ME platform maintains valuable network information for MCS. In particular, the MCS service interacts with the ME platform over the Mp1 reference point to consume network operator data useful for managing the distribution of MCS tasks and worker involvement (e.g., location service, bandwidth manager service, etc.). Also, the ME platform can be used to direct user traffic to a specific ME application (i.e., MCS service) deployed on a corresponding ME host since it supports configuration of the local domain name system (DNS) proxy/server. The Mp3 reference point enables direct communication between two parallel deployments of MCS services in different ME hosts, which can be utilized for direct communication between neighboring ME hosts. The interaction flow in MEC RA is as follows: a new MCS task is submitted by a requester to the system via the customer-facing service (CFS) portal. It is delivered to the Operations Support System (OSS), the highest level management system responsible for the instantiation or termination of ME applications. The OSS forwards this task to the ME Orchestrator (MEO), a central function in the ME system which has visibility over the resources and capabilities of the entire ME network. Since MEO keeps track of all deployed ME hosts, their available services and resources, as well as already instantiated applications and network topology, it can split the received task in several subtasks, based on the location area, and assign each subtask to a corresponding ME host. Also, it can instantiate a new MCS service if needed. In parallel, an MCS application which runs on a UE can use

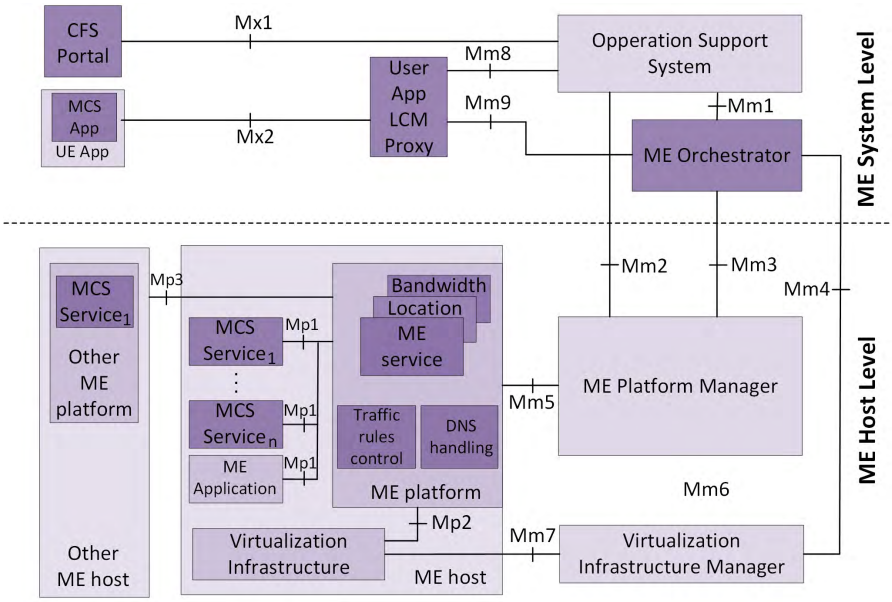

FIGURE 5. Mapping between MCS and MEC reference architecture.

Mx2 reference point to communicate directly with an MCS service deployed on the ME host responsible for the UE in a particular area (e.g., the UE can announce its capabilities, setup the service and submit sensed data). The user application lifecycle management proxy (User App LCM Proxy) will authorize the received request and forward it to the OSS and ME Orchestrator for further processing.

Other components of the MEC architecture, as shown in Figure 5, such as the ME Platform Manager and Virtualization Infrastructure Manager, are not directly influenced by the MCS service deployment. They coordinate and manage normal work operation of the ME host, i.e., they handle the management of specific functionalities on a particular ME host and all applications running on it. In the context of MCS, they are responsible for orchestrating the MCS services on ME hosts in accordance with defined tasks and available workers.

V. EVALUATION

Hereafter we present the evaluation of the proposed MEC architecture for MCS to investigate the overhead incurred due to the need to orchestrate MCS services at network edge. This overhead depends highly on the movement pattern of users involved in MCS tasks, and thus a realistic data set is needed to investigate the distribution of MCS services and the need for their reconfiguration in a MEC environment. For this purpose we are using the data set collected in South Korea from March 2011 to September 2012 [12], where 85 participants actively contributed their location information by using a smartphone application for tagging places. The application was used 79 days on average by a single user. Altogether, users have tagged 13,500 distinct places with information obtained from cellular network providers, smartphone GPS sensor, wireless module, microphone and camera. The data set was originally used for autonomous place detection, and thus we needed to process it to match our need of simulating movements of a large user base which is adequate for MCS deployments.

The data set was first filtered to remove entries where user check-ins are not associated to exact location or timestamp. The filtered data set contains 151,649 user check-ins from 67 unique users. Next, we split the filtered set based on a user identifier and date because the number of unique users is too small to investigate a realistic MCS scenario. Thus, we have created multiple virtual user traces for a single day from the trace of a single real user. Each slice (i.e., user-day slice is a set with user check-ins during a single day) represented a movement pattern of a virtual user during one day. The total of 7,724 virtual users and associated user-day traces is used further on in our evaluation. Since the data set does not contain user location information with high sampling frequency as expected in MCS, we created additional check-ins with a sampling frequency of 1 minute by interpolating expected user location between two consecutive check-ins. This creates a realistic MCS data trace with the total of 4.7 million user-location entries (on average 600 location entries per user).

The analysis of the created MCS data set reveals that user behavior in urban areas highly depends on the time slots in which the analysis is performed and geographical grouping of users in cells which represent a single location context. For geographical grouping, i.e., determining unique cells over wide urban areas, we used the military grid reference system (MGRS) [25]. MGRS is a reference system for locating points on Earth. It supports precision in the range from 1 meter to 100 kilometers and is suitable to be used both in urban and rural areas. We use cells of 10 square kilometers in our

evaluation, where a single MGRS cell is associated to one MEC instance, i.e., we assume that a single ME host is deployed per each MGRS cell covering several cellular antennas that can be densely distributed in urban environments. This is in accordance with a proposed deployment of MEC servers presented in [7]. Note that from now on we use the term MEC cell to refer to a MGRS area with a deployed ME host. The second important parameter influencing the deployment of MCS services in corresponding MEC cells is the time period in which user behavior is observed. Long time periods are not suitable for MCS services since the distribution of users in urban areas is volatile during a single day. In particular, different periods within a single day exhibit different characteristics (e.g., users in the morning travel to work in business districts of a city, while they are dispersed in the afternoon in the suburbs or city center). We decided to analyze the behavior of MCS service in the MEC environment during the course of a day by varying the observation time slot from 15 minutes to 24 hours. We assume that all time slots are independent and an MCS service deployed on a ME host is active during the entire slot if there is at least one user contributing MCS data during this period within a MEC cell. Further insights about user distribution in an urban area during the course of a day can be found in our previous work [15], where the same data set is used for the evaluation of different MCS collection techniques assuming a cloud-based architecture.

TABLE 1. Number of active MEC cells depending on the duration of time slot.

time slot duration	min	max	avg
15 min	70	280	160
30 min	89	404	224
60 min	124	586	323
120 min	210	789	471
240 min	398	1013	698
480 min	628	1220	1008
720 min	967	1530	1249
1440 min	1738	1738	1738

First, we analyze the number of MEC cells which are active in parallel to collect user MCS data while varying the time slot length. More specifically, we counted how many MEC cells require an active MCS service in the observed time slot because at least one user is contributing data in the area covered by an ME host. Table 1 shows the number of active MEC cells, where *min* indicates minimal, *max* maximal, and *avg* average number of MEC cells which execute an MCS service with respect to the length of time slot. We can observe that altogether users have visited 1738 different MEC cells during a 24 hours period (i.e., time slot of 1440 minutes), but better insight of simultaneous activity of MEC cells is provided in numbers corresponding to the 15 minute time slot. In this scenario, the number of active MEC cells requiring an MCS service deployment is significantly reduced, since less than 10% of all MEC cells on average has active users contributing MCS data while for a 60 minute time slot it grows to 18.6%

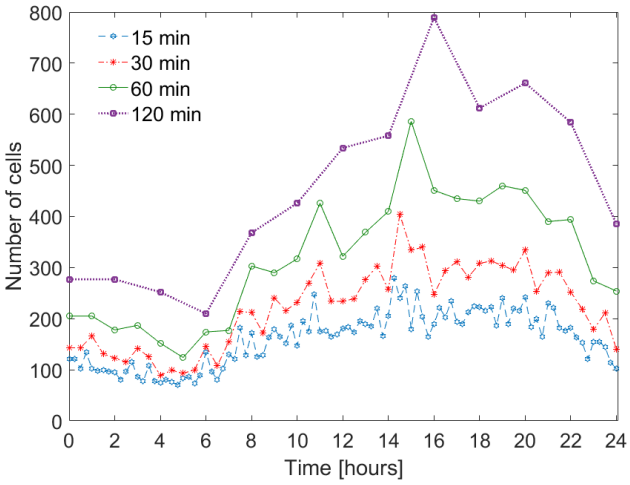

FIGURE 6. Number of active MEC cells during a single day for different time slots.

on average. This shows that a relatively small number of MEC cells need to have an active MCS service if services are reconfigured on hourly basis.

Figure 6 analyzes the data presented in Table 1 over the entire day. It shows the number of active MEC cells during a single day depending on the time slot duration. We can see that, as expected, the number of active cells grows for longer time slots since mobility of users is increased (i.e., users travel longer distances) and they can cross over multiple MEC cells in which an MCS service instance remains active during the whole time slot even when there are no users generating data for the particular service. For example, when an MCS service is reconfigured in intervals of 120 minutes, the number of active MEC cells during peek hours (between 2 p.m. and 4 p.m.) is almost 800, while for the 15 minute intervals, the number of MEC cells which are active in parallel never exceeds 300. This indicates that the activity period of an MCS service should be rather short, but each reconfiguration of MCS services is also costly and thus should be carefully selected.

Next, to analyze the need for reconfiguration of MCS services, we investigate how long a specific MEC cell is active during a single day. In particular, for each MEC cell we trace whether it was visited by at least one user in every time slot during a day as shown in Figure 7, where violet markers present the most visited cells (i.e., MEC cells where an MCS service was active more than 90% of all time slots), while blue markers present the least visited cells (i.e., cells where an MCS service is active less than 10% of time). We can see that only a small number of MEC cells contains active users during the whole day, while significant amount of cells is active less than one third of a day meaning that it is possible to achieve savings by activating an MCS service in less *popular* MEC cells only in those time slots when they are indeed visited by users (i.e., not to keep an MCS service active for time periods when the MEC cell is empty).

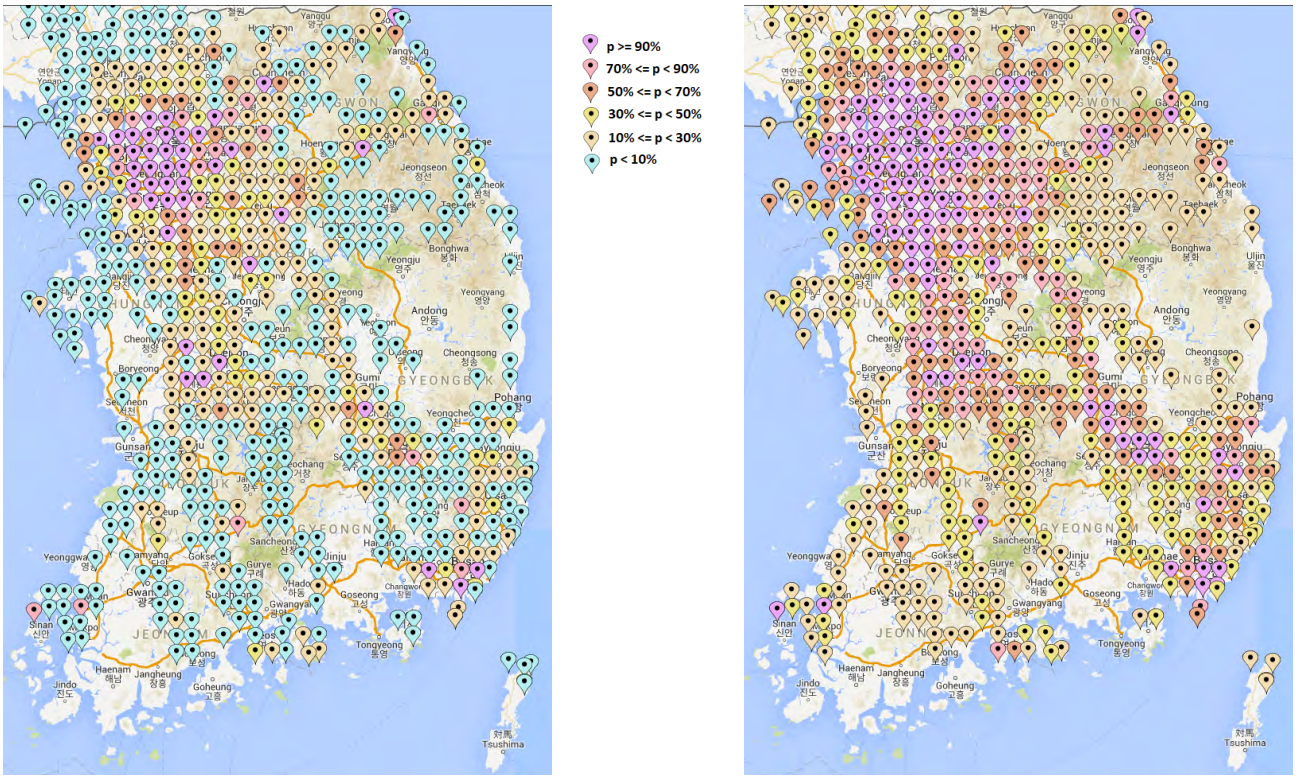

FIGURE 7. Percentage of time in which a MEC cell was visited by users during a 24 hours period. (a) 15 minute time slots. (b) 120 minute time slots.

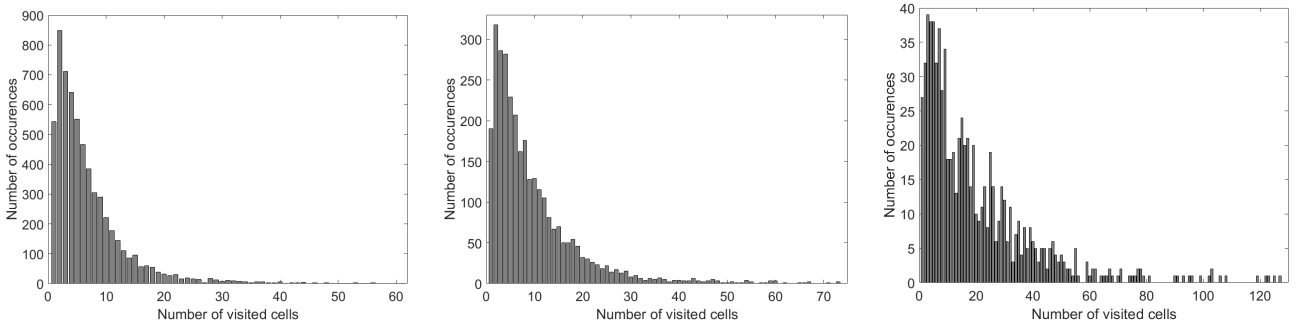

FIGURE 8. Mobility of users across MEC cells. (a) 15 minute time slot. (b) 30 minute time slot. (c) 120 minute time slot.

Also, when we compare the activity of MEC cells during a single day with an MCS service binded to 15 and 120 minute time slots (i.e., service execution life-cycle is re-assessed in fixed time intervals of 15 or 120 minutes, checking if there are still active users in the MEC cell), we can see that significant savings can be achieved if service execution is bound to shorter time intervals. More specifically, if we observe cells that were active less than 10% of time when an MCS service was bound to 15 minute time slots (depicted by blue markers in Figure 7(a)), we can see that their activity period is significantly increased (up to 50% of time) when the same service is observed in 120 minute time slots (depicted by beige and yellow markers in Figure 7(b)). Similarly, when we compare cells that were active more than 90% of time, once again we can see that more resources can be saved when an MCS service is active in 15 minute slots indicating that there is no need to keep all MEC cells active during a day, but rather to start an MCS service in a MEC cell when it is indeed visited by users.

Finally, we analyze the impact of user mobility on the proposed MCS deployment. Figure 8 presents the distribution of the number of visited MEC cells during a single time slot, where x-axis shows the number of different visited cells, and y-axis shows the number of users. We can see that if an

MCS service is active in 15 minute intervals (see Figure 8(a)), the majority of users remain in the same MEC cell (i.e., they do not change their original MEC cell) or they move only to neighboring cells, while with longer service activity periods the number of visited cells in one time slot is significantly increased (see, e.g., Figure 8(b)). Such cell changes cause additional reconfigurations within corresponding MCS services since an ME host has to keep track of user context and transfer his/her state (e.g., user subscriptions) to the neighboring ME node for every cell change (i.e., when a user enters a new MEC cell). For longer time slots (e.g, 120 minutes) there can be more that 100 changes during one time slot, as shown in Figure 8(c) which indicates that it is not profitable to keep track of users all the time, rather that users report to the service when they actually want to contribute the data of interest. Note that the *price* of user's check-in and check-out to the MCS service is higher compare to the *cost* of an MEC cell change, due to the additional overhead of fetching the user-specific data from the central storage in the cloud, but it can be profitable in case of a very high number of visited cells. Additionally, the MEC infrastructure can support user context changes (primarily location updates) by reducing the overhead that would be present in the traditional cloud MCS service deployment by direct data exchange between ME host instances. Also, this graph goes in favor of shorter activity periods for an MCS service instance.

VI. DISCUSSION AND OPEN ISSUES

Edge computing is an appropriate technology for the next-generation MCS services that can perform heterogeneous and large-scale MCS tasks, where users have the control over contributed data, while also being adequately rewarded for their efforts. To utilize the full potential of MCS services on top of MEC, the following open questions and challenges need to be addressed.

Interoperability between MCS deployments covering different domains and phenomena is vital to reuse the same user base for various MCS tasks. In traditional MCS ecosystems, collaboration is possible only between back-end cloud systems, without sharing the same user base, while MEC technology enables collaboration on a ME host level, where MCS services can share the same user base. However, to enable cooperation and data sharing between multiple MCS services deployed on ME hosts coordinated from the cloud, all MCS services should use unified interfaces, open communication protocols and standardized information models. More specifically, interoperability between MCS deployments relates to two aspects of service coexistence in MCS environments enabled by the MEC technology. The first aspect enables the sharing of user-generated data between MCS services running in MEC environment. Semantic interoperability which enables various MCS solutions to understand the data being shared is vital here, with a potential to use the concepts of data bartering between different solutions that in this context have new incentives to share their user base. The second aspect relates to data sharing between a network/ME host operator and an MCS service. Since the network operator maintains valuable data characterizing potential workers (e.g., user activity, location, etc.), an MCS service would greatly benefit from such information, but needs to properly identify users and understand the data provided by the mobile network.

Enriched user context would represent a user with a single profile collecting all user-related information from both the network operator and MCS service domain. Such user profiles would provide a deeper understanding of user behavior for the benefit of all involved parties. The user is involved in MCS tasks with minimum overhead, a network operator acquires better understanding of user needs and can provide personalized customer services, while an MCS service can optimize the management of MCS tasks and consumption of required resources.

Privacy and security is a delicate issue both for MCS and MEC because both environments are inextricably connected to user context. The MEC technology offers positive benefits to the security of MCS data, mostly due to distributed storage and its aggregation at the edge so that a single breach would not expose all user data. In addition, such distributed operation is beneficial from the privacy perspective, but is also challenged by the previous open issue requiring a single user profile. An open and yet vital issue is to ensure that users have the control over their data with protection of sensitive data at all service levels, so that contextual user data are not misused or used by third-parties without user consent.

Orchestration of MCS services needs to be enabled by the MEC virtualization infrastructure which will initiate MCS services on adequate edge resources to answer the needs of MCS tasks and take into account available workers. Furthermore, it needs to regulate an MCS service life-cycle, setting the responsibilities regarding service execution which provides location for data storage and defines rules for data ownership and access rights.

Task scheduling and optimization between users connected to the same edge server strongly depends on their geographical locations, mobility patterns and reputation levels as well as requirements of active MCS tasks. Different algorithms can be deployed on ME servers to minimize energy consumption of mobile devices, cost and execution time, while satisfying sensing coverage and ensuring QoS. For example, in [15], we have proposed an energy-aware MCS system which obviates redundant sensor activity by continuously selecting the k-best sensors for a predefined sensing task within a cell. Furthermore, a geographical distribution of MCS tasks and the types of MCS services deployed on ME servers will also have influence on the performance of the proposed system. It is possible to achieve significant energy savings and reduce latency because tasks are processed at an ME server and there is no need to send all data to a back-end cloud.

To fully exploit the potential of this new MCS environment, and in light of the sharing economies, we envision an open MCS marketplace which matches various sensing tasks on-the-fly with workers willing to contribute sensor data, while

also being provided by adequate incentives for the tasks that do not disrupt their daily activities [26]. However, it requires adequate addressing of the aforementioned challenges to create the potential for new business models involving various stakeholders. The marketplace could be operated by a network provider, an entity that is trusted by all involved parties. Such a marketplace and novel hierarchical and interoperable MCS environment would offer opportunities for the development of novel MCS applications by SMEs and startups which are not solely dependent on their own user-base.

VII. RELATED WORK

As already stated, MCS platforms are typically based on a centralized architecture. For example, Sensarena [4] is a crowdsensing platform relying on mobile users who communicate through a centralized server. CAROMM [5] defines an MCS framework which enables end-users to upload different sensing data to the cloud. Our solution CUPUS [3], a CloUd-based PUblish/Subscribe middleware for the Internet of Things, enables the management of mobile sensor resources within the cloud, supports filtering and aggregation of sensor data on mobile devices prior to its transmission into the cloud, and can push information of interest from the cloud to user devices in near real-time. However, recently the need for a distributed MCS architecture has been recognized. One of the early works presented in [8] is MECA (Mobile Edge Capture and Analytics), a middleware for data collection from mobile devices. The MECA architecture introduced an edge layer located at the network edge (e.g., base stations in cellular networks) which receives requirements from the back-end servers, manages the data collection among a subset of local devices, and runs edge analytics for primitive data processing. Similarly, Sahni *et al.* [27] have proposed a new computing paradigm, named Edge Mesh, which distributes the decision-making tasks among edge devices within the network instead of sending all data to a centralized server. Another example, RedEdge, is a novel big data processing solution which enables processing of big data streams near the data source in mobile edge cloud computing environments [28]. In this paper we go beyond state-of-the art since we propose a generic model for MCS suitable for edge computing environments. More precisely, we identify the characteristic features of MCS services, and propose a layered functional architecture which we position within the MEC reference architecture. Our hierarchical deployment is adequate for both data analytics, and efficient real-time usage at the edge of mobile network since MEC reference architecture ideally matches the requirements of large and scalable MCS services.

Another relevant approach suitable for IoT services reusing resources at network edge is presented in [29]. The authors propose Fog Computing, a hierarchical and distributed platform which contains compute, storage, and network resources, while enabling new services and applications at the network edge. Similarly, Jayaraman *et al.* [9] propose CARDAP, a context aware distributed mobile data analytics platform for MCS applications, which contains a fog layer responsible for local computing and data storage. Luan *et al.* [30] present a three-tier Mobile-Fog-Cloud architecture which deploys highly virtualized computing and communication facilities at the proximity of mobile users, while Tang *et al.* [31] propose a 4-layer hierarchical distributed Fog Computing architecture which enables parallel data processing at the edge of the network and is therefore suitable for deployment of smart city services. Similarly, we also define a 4-layer generic functional architecture for MCS services suitable for edge computing environments, where the top layer is for application purposes. Also, in this paper we provide insights to achieve interoperable MCS services deployed at ME hosts, while other papers focus on offering a single ecosystem for MCS services.

VIII. CONCLUSION

Mobile crowdsensing is a human-driven paradigm empowered by ordinary citizens who contribute and share sensor data by means of mobile devices and wearables. It is a true example of a sharing economy where generated sensor data represents a shared resource, while scalable and interoperable technical solutions are needed to create the next-generation marketplace for MCS involving a huge number of users and various stakeholders.

This article presents a reference architecture for hierarchical and large-scale deployments of MCS services which assumes the usage of edge computing resources to decentralize MCS services and improve their performance. Mobile Edge Computing brings computation and storage to the edge of mobile network providing MCS services in close proximity of users. The goal of such architecture is to simplify service execution and increase the quality of service, primarily by reducing the latency and complexity of data processing. Our analysis of the service overhead introduced by the distributed architecture which requires reconfiguration of edge MCS services shows that this overhead is controllable and small, especially for shorter reconfiguration periods of MCS services at network edge. Thus, it represents a promising approach for enabling the next-generation MCS marketplace of massive scale.

REFERENCES

[1] B. Guo, C. Chen, D. Zhang, Z. Yu, and A. Chin, "Mobile crowd sensing and computing: When participatory sensing meets participatory social media," *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 131–137, Feb. 2016.

[2] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 140–150, Sep. 2010.

[3] A. Antonić, M. Marjanović, K. Pripužić, and I. P. Žarko, "A mobile crowd sensing ecosystem enabled by CUPUS: Cloud-based publish/subscribe middleware for the Internet of Things," *Future Generat. Comput. Syst.*, vol. 56, pp. 607–622, Mar. 2016.

[4] R. Ben Messaoud, Z. Rejiba, and Y. Ghamri-Doudane, "An energy-aware end-to-end Crowdsensing platform: Sensarena," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2016, pp. 284–285.

[5] W. Sherchan, P. P. Jayaraman, S. Krishnaswamy, A. Zaslavsky, S. Loke, and A. Sinha, "Using on-the-move mining for mobile crowdsensing," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, Jul. 2012, pp. 115–124.

[6] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
[7] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.
[8] F. Ye, R. Ganti, R. Dimaghani, K. Grueneberg, and S. Calo, "MECA: Mobile edge capture and analysis middleware for social sensing applications," in *Proc. ACM 21st Int. Conf. World Wide Web*, New York, NY, USA, 2012, pp. 699–702
[9] P. P. Jayaraman, J. B. Gomes, H. L. Nguyen, Z. S. Abdallah, S. Krishnaswamy, and A. Zaslavsky, "Scalable energy-efficient distributed data analytics for crowdsensing applications in mobile environments," *IEEE Trans. Comput. Soc. Syst.*, vol. 2, no. 3, pp. 109–123, Sep. 2015.
[10] S. K. Datta, R. P. F. da Costa, C. Bonnet, and J. Härri, "oneM2M architecture based IoT framework for mobile crowd sensing in smart cities," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2016, pp. 168–173.
[11] *Mobile Edge Computing (MEC); Framework and Reference Architecture*, document ETSI GS MEC 003 V1.1.1 (2016-03), 2016.
[12] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, "Understanding the coverage and scalability of place-centric crowdsensing," in *Proc. ACM Int. Joint Conf. Pervas. Ubiquitous Comput.*, 2013, pp. 3–12.
[13] I. P. Žarko *et al.*, "Towards an IoT framework for semantic and organizational interoperability," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2017, pp. 1–6.
[14] S. Schmid *et al.*, "An architecture for interoperable IoT ecosystems," in *Interoperability and Open-Source Solutions for the Internet of Things*. Cham, Switzerland: Springer, 2017, pp. 39–55.
[15] M. Marjanović, L. Skorin-Kapov, K. Pripužić, A. Antonić, and I. P. Žarko, "Energy-aware and quality-driven sensor management for green mobile crowd sensing," *J. Netw. Comput. Appl.*, vol. 59, pp. 95–108, Jan. 2016.
[16] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 2011.
[17] F. Kalim, J. P. Jeong, and M. U. Ilyas, "CRATER: A crowd sensing application to estimate road conditions," *IEEE Access*, vol. 4, pp. 8317–8326, 2016.
[18] B. Guo, Z. Yu, D. Zhang, and X. Zhou, "Cross-community sensing and mining," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 144–152, Aug. 2014.
[19] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
[20] G. M. Lee, P. Jung-Soo, N. Kong, N. Crespi, and I. Chong, *The Internet of Things—Concept and Problem Statement*, document draft-lee-iot-problem-statement-05, Working Draft, IETF Secretariat, Internet-Draft, Jul. 2012.
[21] A. Srinivasan, J. Teitelbaum, J. Wu, M. Cardei, and H. Liang, "Reputation-and-trust-based systems for ad hoc networks," in *Algorithms and Protocols for Wireless and Mobile Ad Hoc Networks*, A. Boukerch, Ed. Hoboken, NJ, USA: Wiley, 2008, doi: 10.1002/9780470396384.ch13.
[22] M. Pouryazdan, B. Kantarci, T. Soyata, L. Foschini, and H. Song, "Quantifying user reputation scores, data trustworthiness, and user incentives in mobile crowd-sensing," *IEEE Access*, vol. 5, pp. 1382–1397, 2017.
[23] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," *Comput. Commun.*, vol. 54, pp. 1–31, Dec. 2014.
[24] J. S. Kumar and D. R. Patel, "A survey on Internet of Things: Security and privacy issues," *Int. J. Comput. Appl.*, vol. 90, no. 11, pp. 20–26, Mar. 2014.
[25] T. D'Roza and G. Bilchev, "An overview of location-based services," *BT Technol. J.*, vol. 21, no. 1, pp. 20–27, 2003.
[26] T. Luo, S. S. Kanhere, J. Huang, S. K. Das, and F. Wu, "Sustainable incentives for mobile crowdsensing: Auctions, lotteries, and trust and reputation systems," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 68–74, Mar. 2017.
[27] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge mesh: A new paradigm to enable distributed intelligence in Internet of Things," *IEEE Access*, vol. 5, pp. 16441–16458, 2017.
[28] M. H. ur Rehman, P. P. Jayaraman, S. ur Malik, A. U. R. Khan, and M. M. Gaber, "Rededge: A novel architecture for big data processing in mobile edge computing environments," *J. Sensor Actuator Netw.*, vol. 6, no. 3, p. 17, 2017.
[29] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Cham, Switzerland: Springer, 2014, pp. 169–186.
[30] T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun. (Feb. 2015). "Fog computing: Focusing on mobile users at the edge." [Online]. Available: https://arxiv.org/abs/1502.01815
[31] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proc. ACM ASE BigData Soc. Inf. (ASE BD&SI)*, New York, NY, USA, 2015, pp. 28:1–28:6.

MARTINA MARJANOVIĆ received the M.Sc. degree in information and communication technology from the University of Zagreb in 2013, where she is currently pursuing the Ph.D. degree with the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb. She was a Guest Student with the Vienna University of Technology in 2012. She is currently a Research Associate with the Department of Telecommunications, Faculty of Electrical Engineering and Computing, University of Zagreb. Her interests include mobile crowdsensing architectures and applications, contextual services, and quality of service in the Internet of Things.

ALEKSANDAR ANTONIĆ is currently pursuing the Ph.D. degree with the Faculty of Electrical Engineering and Computing, University of Zagreb. He was a Research Intern with the Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland, in 2011. He has experience of participating in projects founded by the EU Framework Programme, Croatian Ministry of Science, Education and Sports and industry. He is currently a Research Associate with the Faculty of Electrical Engineering and Computing, University of Zagreb. His interests include large-scale distributed systems, real-time data stream processing, information retrieval, and recommender systems.

IVANA PODNAR ŽARKO was a Research Associate with the Distributed Systems Group, Technical University of Vienna, from 2000 to 2001, and a Post-Doctoral Researcher with the Distributed Information Systems Laboratory, Swiss Federal Institute of Technology, Lausanne, from 2005 to 2006. She is currently an Associate Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, and the Head of the Internet of Things (IoT) Laboratory. She is also the Technical Manager of the H2020 Project symbIoTe: Symbiosis of smart objects across IoT environments. Her main research interests include large-scale distributed systems, IoT, and big data processing. She has co-authored over 60 scientific journal and conference papers in these domains. She has served as a Program Committee Member for a number of international conferences and workshops (e.g., IEEE Globecom, CCNC, and ISCC). She was the Chapter Chair of the IEEE Communications Society, Croatia Chapter (2011–2014).