Optimization strategies for the selection of mobile edges in hybrid crowdsensing architectures

Optimization Strategies for the Selection of Mobile Edges in Hybrid CrowdSensing Architectures

Dimitri Belli[a], Stefano Chessa[a,c], Antonio Corradi[b], Luca Foschini[b], Michele Girolami[c]

[a]*Department of Computer Science University of Pisa, Italy, email: name.surname@di.unipi.it*
bDipartimento di Informatica: Scienza e Ingegneria University of Bologna, Bologna, Italy, email: name.surname@unibo.it
[c]*Istituto di Scienza e Tecnologie dell'Informazione, National Council of Research, Pisa, Italy, email: name.surname@isti.cnr.it*

Abstract

Communication infrastructures are rapidly evolving to support 5G enabling lower latency, high reliability, and scalability of the network and of the service provisioning. An important element of the 5G vision is Multi-access Edge Computing (MEC), that leverages the availability of powerful and low-cost middle boxes, i.e., MEC nodes, statically deployed at suitable edges of the network to extend the centralized cloud backbone. At the same time, after almost a decade of research, Mobile CrowdSensing (MCS) has established the technology able to collect sensing data on the environment by using personal devices, usually smartphones, as powerful sensing-and-communication platforms. Even though, mutual benefits due to the integration of MEC and Mobile CrowdSensing (MCS) are still largely unexplored. In this paper, we address and analyze the potential of the synergic use of MCS and MEC by thoroughly assessing various strategies for the selection of both traditional Fixed MEC (FMEC) edges as well as human-enabled Mobile MEC (M2EC) edges to support the collection of mobile CrowdSensing data. Collected results quantitatively show the effectiveness of the proposed optimization strategies in elastically scaling the load at edge nodes according to runtime provisioning needs.

Keywords: Mobile CrowdSensing; multi-access edge computing; clustering; sensor data collection.

1. Introduction

Mobile CrowdSensing (MCS) [1] is a technological paradigm that provides an effective solution for the capillary collection of large amounts of information over large crowded areas, which makes it particularly attractive as enabling technology for the smart cities. The reason for its success and acceptance relies in the exploitation of the sensing ability of personal communication devices (especially smartphones, but also wearable devices) that are carried out by people continuously, thus providing to a MCS platform the ability to collect data from every part of a city. Furthermore, the wide range of sensors currently embedded in wearable and smartphones combined with the ability of fusing heterogeneous data together, enables an even wide range of sensing applications. The research on MCS in the last years focused on many aspects, ranging from the optimization of the task assignment [2], even taking into account the energy budget of the personal devices and the energy cost of the assigned tasks [3, 4] to the strategies for the involvement of volunteers in MCS data collection campaigns [5].

In a MCS platform personal devices are, however, only the outer (and visible) line of the architecture. Behind this line, the MCS relies on the Internet and powerful servers in the cloud to aggregate, store, and process data coming from potentially billions of devices worldwide. In order to make this architecture more efficient, this infrastructure is often enriched with an additional level of edge servers, hereafter called Fixed Multi-access Edge Computing nodes (FMECs), which provides an intermediate level of data filtering, aggregation, analysis and storage functionalities operating on a localized base between the devices and the cloud. This infrastructure is commonly known as Multi-access Edge Computing (MEC) [6].

The synergy between a MEC architecture and a MCS platform has been subject of intensive research in the last years, motivated by the need to achieve a better integration, with the purpose of reducing the deployment and maintenance costs. This research effort was particularly fruitful for those MCS applications that aim at very large data collection over a long-lasting time periods for off-line and big data analytics. Examples of such applications concern the analysis of people/vehicle mobility [7] or of the trends of pollution [8] in a smart city for example. In these applications, the latency requirements typical of several communication networks can be almost relaxed to improve the efficiency in terms of deployment and maintenance costs of the MCS platform. A first approach adopted to reduce the latency is based on

opportunistic communications between FMECs and mobile devices, based on short-range network interfaces (e.g Wi-Fi Direct or LTE Direct). With this approach, rather than transmitting every sensed data directly to the cloud through a broadband interface, the device logs the sensed data and, when it gets close enough to a FMEC edge, the device transmits all data back to the FMEC [9]. This approach can be further combined with the introduction of Mobile Multi-access Edge Computing nodes (M2ECs) [10]. Such nodes are general-purpose mobile devices that are selected by the MCS platform to act as mobile edges for a given time period. During this period, M2ECs play the role of regular FMECs but, differently from them, they can move and use their short-range interfaces to collect data in an opportunistic way from other devices in the nearby. Since M2ECs are also personal devices they move with their owner, hence while the spatial coverage of FMEC is fixed, the spatial coverage of M2ECs varies over time. Mobility, in turn, is generally related to the sociality of the M2ECs owners. A second advantage of M2ECs is that they do not have any deployment and maintenance costs, since such costs are usually guaranteed implicitly by their owners.

In this work we follow this trend of research. Specifically, we first assess the use of different strategies for the deployment of FMECs based on the mobility of the MCS users. Then, we present an assessment of different strategies for the selection of the M2ECs. The strategies we propose are based on some features of the sociality of the MCS users. Finally, we assess a hybrid architecture that combines both FMEC and M2EC. Note that this work extends our preliminary work [11], by further analyzing the performance of the selection strategies of Fixed and Mobile MECs. In particular, we explore the performance of the selection strategies by considering the temporal dimension.

We performed the assessment of all these selection strategies by means of emulation tool tested over a large dataset of real users resulting from the ParticipAct project [12]. Our experimental results show that simple heuristics for the deployment of the FMEC (such as grid-based), which aim to cover space are, in fact, outperformed by clustering algorithms as DBSCAN. Furthermore, in hybrid architectures combining FMEC and M2EC it is seen how M2EC can efficiently replace the FMEC, as they can achieve similar results even with a limited number of them. The rest of the paper presents a survey of relevant works in Section 2, the emulation methodology in Section 3, and the results of the emulations with FMEC alone, of M2EC alone and in combination in Section 4. Section 5 draws the conclusions.

2. Background and Related Work

This section presents the main characteristics and limitations of the MEC and of the MCS paradigms. We review some studies focused on the sociality of users enabling the selection strategies discussed in Section 4. We conclude this section with a review of those works that, similarly to our approach, explore the combined use of MEC and MCS to bring all the benefits of HEC [13] (Human-driven Edge Computing) model, by specifically addressing the M2EC selection problem in the new hybrid scenario.

The MEC [14] paradigm proposes a new standard that enriches radio access networks via the introduction of edge nodes with computational and storage at the access network borders. The MEC inherently supports peripheral nodes by reducing latency for mobile users and optimizing mobile backhaul utilization through the interposition of the edge layer. Such decentralized cloud technology makes the MEC one of the cornerstones of upcoming 5G systems [15]. In fact, MEC eases the convergence between telecommunication and information technology (IT) services [16, 17, 18]. MEC node resources are virtualized and made available to third parties, typically via easy-to-use Application Programming Interfaces (APIs). Basic MEC implementations are made up of individual platforms that provide local services without considering user mobility. More recent MEC implementations consider aspects like traffic [19, 20], mobility [21], and accountability. Moreover, they support heterogeneous types of (wireless) networking interfaces and they ease the execution of parts of the computation at the edge (i.e., offloading), both in indoor and outdoor scenarios [22, 23, 24, 25]. Concerning the use of MEC with the Internet of Things (IoT) applications, recent studies focused on the development of platforms that ensure management and interoperability between large-scale devices without loss of bandwidth or increased latency [26]. With a plethora of devices interacting with intermediate MEC nodes, computation offloading techniques became valuable methods to save energy, battery lifetime, and computational costs. We limit to mention a comprehensive study on these techniques over 5G heterogeneous networks [27]. Other major use cases of the MEC paradigm include optimization techniques for distributed content discovery and delivery [28, 29], caching [30], big data storage and computation [31], and services for smart cities such as localization [32], emergency [33], and public safety [34, 35]. We further survey some MEC applications designed to build a MEC end-to-end emulator to evaluate the possibility of bridging the functionalities of an existing data center or mobile network emulators [36]. Moreover, the MEC paradigm is also applied to the possibility of preventing in vulnerable road users through the use of communication capabilities of the current LTE networks as discussed in [37]. Another interesting research line convers the energy footprint of a MEC-based architecture and its implication on the optimal deployment [39].

The MCS paradigm enables the massive sensing in a mobile scenario [40], by exploiting the active participation of the crowd. MCS exploits the high diffusion of mobile devices with sensing capabilities as well as the user mobility. The combination of such two factors gives rise to design efficient sensing applications based on a collaborative approach. Since those devices are carried by humans, the knowledge of social attitude of people becomes a crucial point. In this context, a sensing task (also referred to as task) is type of information that can be collected through a mobile device.

Examples of tasks include taking a photo, recording an audio track, sensing the environment or collecting personalized feedbacks from the users. Strategies for task assignment include the study of recruitment areas considering the population density at different times of the day, and the use activity as the willingness in performing tasks and the time spent in city hotspots [41]. Careful selection strategies give the opportunity to minimize costs and maximize aspects such as the accuracy in the assignment and the execution of tasks. In this perspective, recruitment policies based on the sociability of individuals can typically grant good results [42]. However, the management of all involved technical and human resources is still an open issue, especially for large-scale sensing scenarios. First, MCS campaigns are characterized by devices with heterogeneous computational capacities and energy resources and battery drain is the main concern for mobile users. Second, although user recruitment policies leveraging profiles and current battery level of devices for task assignments have been devised there is still room for improvement [43, 44]. Third, apart cost-saving energy solutions, there is also the need to limit bandwidth wastage. Some works propose local data mining at mobile devices via production of intermediate results, showing a real energy and bandwidth saving in data transmission to remote servers [45]. Fourth, in addition to recruitment issues, there is also the problem of how to stimulate the participation of users to keep them active, such as using incentive mechanisms for MCS campaign users [46, 47, 48]. Similarly, in [38] authors propose a simulation environment framework emulating real active participants' behavior of a MCS system, enabling analysis of the effects that rewards and participation bring to the overall MCS platform performance. Finally, linked to above management issue there is also the data security problem, including data storage and communication issues, as well as privacy and data anonymization issues [49, 50]. This is another vast area and for an extensive analysis we refer interested readers to [51].

To address above challenging management issues, the MCS research community in recent years has started to investigate the use of hybrid solutions able to support the synergistic use of MCS techniques combined with MEC solutions, like the one proposed here. For instance, in the context of smart cities [42], some seminal research activities propose to include models for optimizing content sharing by leveraging the sociality and mobility of the MCS nodes [52]. Similarly, [53] and [54] explore the use of MEC-based architectures for massive scale MCS services and privacy preserving citizens' data control flow. However, to the best of our knowledge only a very limited number of works proposed social-driven edge computing architectures based on incentives and centrality measures to reduce installation and maintenance costs for MEC intermediate layer nodes [55]. In this context, one of the main challenges is the selection of M2ECs acting at the edge of the cloud, namely collecting data from others and uploading data to the backend [56, 57, 58]. An efficient selection of the devices can result with high spatial and temporal coverage (as discussed in [59, 60]). Spatial coverage considers the number of areas of a region that can be monitored with a certain accuracy, such as the districts of a city. Temporal coverage, instead, refers to the time required in order to collect data from all the regions. In [61], authors propose a selection strategy based on the dimension of the region to cover. Authors consider deterministic node mobility and propose several heuristics for the selection of devices, in order to minimize an objective function. Other recent works [62, 63, 64] consider incentive mechanisms for the users in order to increase the probability of collecting data from specific locations.

3. Hybrid CrowdSensing Framework

This section introduces our CrowdSensing framework that reproduces realistic for the task assignment and for the data collection. We first report the reference architecture used for the whole MCS process management. Then, we introduce the ParticipAct dataset, and some mobility features characterizing the dataset. Finally, we detail the experimental settings used to configure our experiments.

3.1. A 3-Tired Reference Architecture for MCS

In the remainder of this paper we refer to a reference MCS platform architecture that comprises three tiers: users, devices and a cloud server. Users are voluntarily that accept to join the MCS data collection campaign. They, typically, use their devices to collect data by means of a custom mobile application acting as bridge between the user and the cloud server. The cloud server (also referred to as backend) requests task to the user's devices. A task is an activity assigned from the backend to a device and it generally refers to a request of data collection. Each task is associated with a TTL (Time to Leave), corresponding to the time within the task can be completed by the user's device. Examples of MCS tasks that we consider are: collecting multi-media contents from a geographic region, collecting feedbacks from the users about an event or monitoring environmental parameters such as the noise intensity or e.g. measuring the WiFi signals in district. Such examples of task do not require a real-time answer from a device; differently, the users of the MCS measurement campaign have a sufficient time period to complete their task. It is important to remark that users are volunteers without any obligation to complete a task, therefore it is important to extend the TTL to avoid uncompleted task or drop-off of users.

The MCS mobile application collects data autonomously through the sensors or with the active participation of the users. In turn, the mobile application can forward the collected data to the backend for storage for off-line analysis. We foresee two ways of communication between the device and the backend:

- via broadband communication (e.g. 4G LTE or 5G) to directly connect the mobile device to the backend;
- via the use of edges (both fixed or mobile) that are deployed in the sensing region. In the last case, the MCS platform exploits the capabilities of a mobile edge computing architecture.

Our hybrid architecture combines FMECs with M2ECs. M2ECs are selected among the personal devices carried by MCS users, and they collect opportunistically all the data produced by any other MCS devices that come in the range of their short-range radio interfaces. M2EC can be implemented in client applications by relying on modern virtualization and containerization technologies, for which we refer interested readers to [51].

While the broadband communications are long-range and subject to costs, the communications between the user's devices and the MEC (fixed or mobile) rely on the use of short-range communications that are generally free of charge. Note also that the selection of M2ECs among all the MCS devices is a critical point that can affect the overall performance of the hybrid architecture we propose.

3.2. The ParticipAct Dataset

Our experiments rely on the ParticipAct CrowdSensing project [12] carried out by approximately 170 students from the University of Bologna, Italy. Students were equipped with an Android smartphone provisioned with the ParticipAct mobile app able to collect information about the user's location every approximately 2.5 minutes through the Google Location APIs. In particular, such APIs can access the device's position based on the location of the Wi-Fi access point, the GPS or by considering the position of the base station to which a device is connected with. As a result, the accuracy of the user's position varies with the type of technology used for the localization purpose.

The dataset collects not only the user's location but also feedbacks from users and media content generated by volunteers. ParticipAct's data cover a period of 18 months, from December 2013 to February 2015. For our purposes, we considered a period of one month, from March 1st to March 31st, 2014. The time frame is based by considering the number of active users and the existence of significant variations of the users, so that to reproduce realistic conditions of a MCS measurement campaign. The mobility of the users joining ParticipAct is unrestricted: users live in town or sub-urban areas; some of them commute daily by train, while others walk or move by bike. For the sake of exemplification, we report in Figure 1 the geographical extension of ParticipAct dataset. Most crowded areas are in the Emilia-Romagna region, where Bologna is the capital; however, the involved people roam also outside the region.

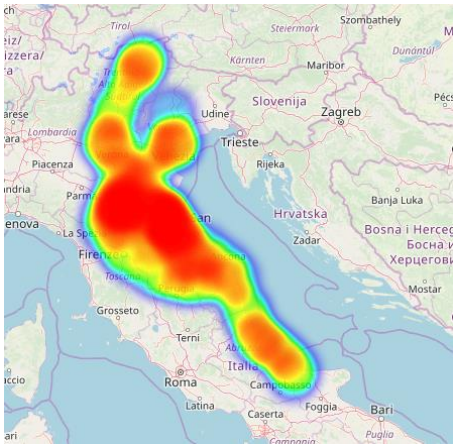


Fig. 1 Geographic extension of the ParticipAct dataset.

We further analyze the accuracy of the user's location detected with the Google Location APIs. The accuracy varies remarkably as shown by the Cumulative Distribution Function (CDF) shown in Figure 2. The figure shows the probability of having accuracy values in the range: 3.9 meters to 9959 meters. Values of accuracy in the order of 103 or higher are due to the use of cellular base stations as location source. In particular, the user is localized at the position of the base station to which its device is connected with. This condition represents our worst case, and such values can be filtered out. As a general trend, we measure the 25th percentile of 27 meters and the 50th percentile of 39 meters which represent admissible values for the purpose of this work.

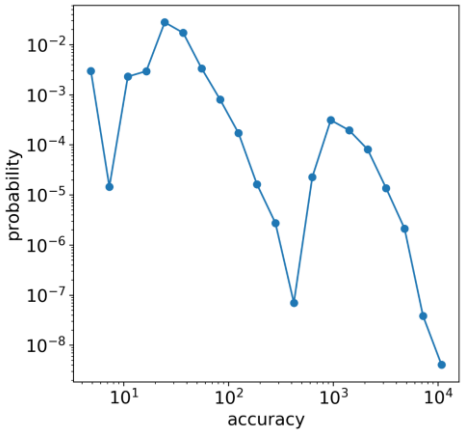


Fig. 2 CDF of the accuracy value.

3.3. Distribution of Users with the ParticipAct Dataset

We now further analyze how users in ParticipAct distribute in the region during the observation period. As discussed in Section 3.2, we restrict our analysis to March 2014 during which users move according to their daily scheduling. Most of the ParticipAct 'users are students from the University of Bologna, therefore their mobility is influenced by lessons, examination periods and weekly breaks. We first analyze the number of active users during the observation period. A user is defined active if its device reports the position at least once during a day. Some users do not always report their position, this can be caused by several factors such as: internet connection unavailability, the user explicitly disables the networking capabilities, the user deliberately switches off the mobile device etc. Figure 3 reports a heatmap showing the amount of active users aggregated for each week's day and during the 6 weeks covering March 2014, from week 9 to week 14.

We observe that during weeks 11 to 13 a lower number of users are active, with respect to the other weeks. As average, we measure 78 active users, ranging from a minimum of 40 to a maximum of 116 users. The reduction of the number of active users, is also confirmed by analysis of the locations. In particular, the measure the amount of visits in each location hourly, as reported in Figure 4. It is not possible to identify the exact motivation of such reduction, but the daily variations

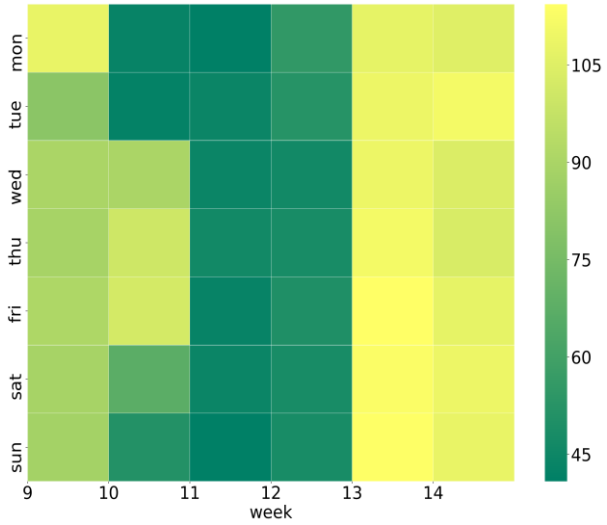


Fig. 3 Active users in March 2014.

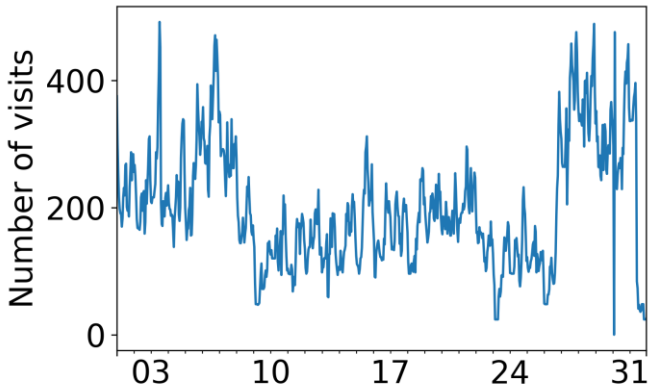


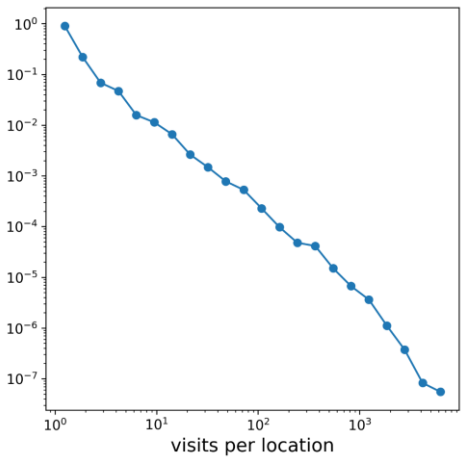
Fig. 4 Number of visits in March 2014.

increase the reality of our experimental dataset at realistic conditions, this represents one the criteria we adopted to select the observation period.

We now study the number of visits in each location as described in [65, 66]. For the shake of simplicity, we restrict such analysis to the Bologna city center, so that to better highlight the differences. In particular, we measure the CDF of the number of visits in every location, as reported in Figure 5. As shown in the left-side of Figure 5, the CDF follows a power-low distribution, according to which the probability of having locations with a high number of visits reduces with the number of visits. The right-side of Figure 5, shows the geographic distribution of the top-k locations, namely the first k = 22 locations with at least 1k visits each. We finally show how locations vary during a day. To this purpose, we identify 4 temporal intervals: 07:00 – 09:00, 09:00– 18.00, 18:00-20:00, 20:00 – 22.00 and we aggregate data across such intervals during the observation period of March 2014. Figure 6 shows a heatmap of the locations visited for each of the 4 intervals. From the figure it is clear that, according to the time interval, different regions of the Bologna city activate more clearly with respect to others. The rhythm of such mobility is highly influenced by several factors such as the type of users involved in the ParticipAct, their age and their habits as well as the way users commute from and to their private residences. We consider that a synthetic model can hardly capture such heterogeneity of the user's mobility. Differently, we support the need of real-world dataset providing evidence of the natural way people move in an urban area. Moreover, the use of a mobility model can surely support an initial design of the MEC selection strategy (see Section 4), but not for a realistic assessment of its performance. In fact, situations such as the one described in Figure 4 (reduction of the number of active users) can be hardly reproduced with a pure simulation setting that, generally, does not alter the amount of active users during a test.

3.4. Configuration Settings

We evaluate the performance of the hybrid architecture introduced in Section 3.1 by means of emulation tools. To this purpose, we developed a python-based MCS emulator that reproduces the collection of information from the crowd. The simulator can be configured by setting the number of FMEC and M2EC and the selection strategy for the positioning of FMECs as well as the M2ECs. The current implementation supports DBSCAN, grid and random strategies for the



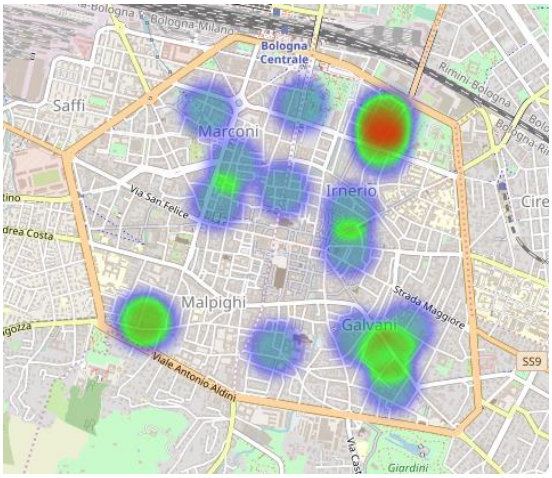


Fig. 5 CDF of the number of vists and the top-*k* locations.

TABLE I. CONFIGURATION SETTINGS.

Property	Value
Number of participants	170
Observation period	March 1st – 31th 2014
Number of requests	5 *k*
Request time interval	9.00 AM – 8.00 PM
TTL	7.5 days
Co-location distance	100 *m*
FMEC strategies	DBSCAN, GRID, RANDOM
M^2EC strategies	Betweenness, Eigenvector centrality

positioning of the FMEC (see Section 4.1), and two graph-based selection strategies for M2ECs based on the betweenness centrality the eigenvector centrality (see Section 4.2).

Our tool implements the communication model for the generation of tasks from the backend and the collection of the data provided by devices. Each task is also associated with a timestamp and to a time to live (TTL). The timestamp represents the time (in the emulation) at which the device produces a new data (and thus the communication request is also originated), while the TTL is the expiration time for the corresponding communication request. If at a given time the device lays in the transmission range of a FMEC or of a M2EC, then our emulation tool assumes that the data is transmitted successfully and it records the time elapsed from the generation to the transmission (the communication latency) and the FMEC or the M2EC that received the data. If the request expires, then we consider that data generated by a device are transmitted back to the cloud by using a broadband communication. Tasks are generated by the backend and they are randomly assigned to the user's devices except to the ones elected as M2ECs. We simulate the generation of a bunch of 5k tasks and the assigning users can complete the tasks during a time period (TTL). If the task requires to collect data from a specific region of the city, the stationary behavior of the assigning users helps to collect incremental data along the time. More specifically, we consider that the routinary behavior of a user supports an incremental collection of data from the region of interest for a task. Moreover, it is worth to notice that some tasks require the users' intervention and, consequently, they can be completed more slowly. For instance, taking a photo of a remote region requires the availability of the user to deviate from his/her routinely path.

For the purpose of this work, we are interested in measuring the performance of our architecture in satisfying the requests generated from the crowd to the backend by means of short-range communications. In particular, we assume that devices can interact directly in ad-hoc mode without the existence of a network interface. We consider WiFi Direct as a viable solution for such peer-to-peer communications since it can be used both indoor and outdoor by exploiting the WiFi network interface. It is worth to notice, that the upcoming LTE Direct and 5G network interfaces can be further be used for this short-range communications [67]. The interaction between devices are obtained from the co-location traces extracted from the mobility trace of the ParticipAct dataset. More specifically, co-locations are obtained by detecting those a pair of users in proximity at the same time. Therefore, a pair of device can interact with a short-range network interface if the pair lays on the same place at the same time for a time interval. Device are co-located within a distance up to 100 meters, an acceptable distance for wireless outdoor communications. We assume that tasks are generated only during daily hours, from 9.00 AM to 8 PM. Such assumption follows the analysis of the responsiveness of users in the ParticipAct which is higher during the daily hours rather than the nightly hours. Moreover, tasks are generated during the first three weeks of March 2014, so that to enable the emulator to run for completion by the end of March. Table 1 summarizes all the configuration settings we described.

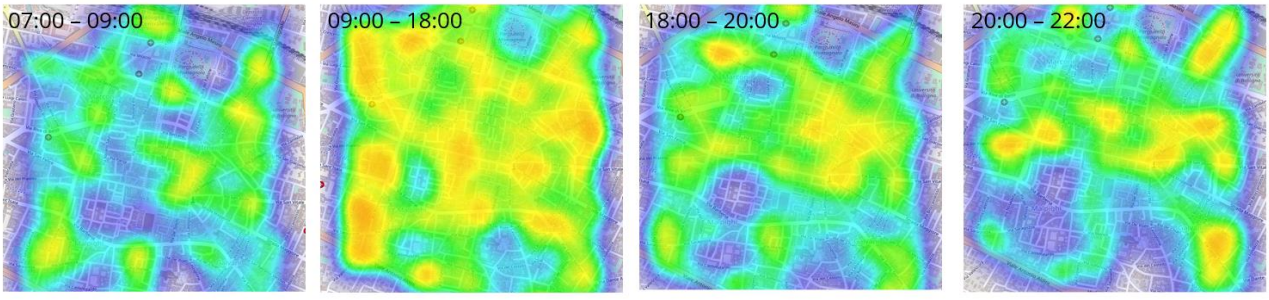


Fig. 6 Heatmap of the locations visited during the different time intervals.

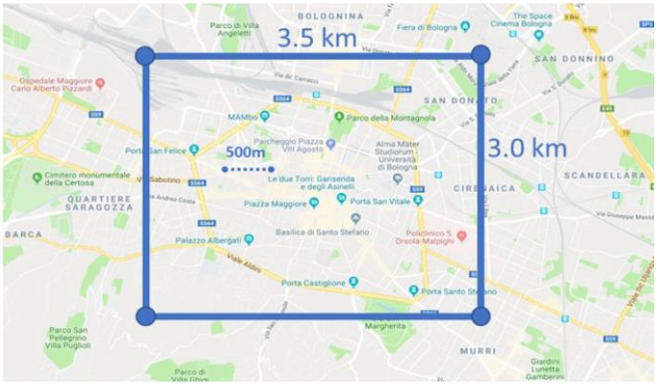


Fig. 7. Bounding box the sensing region.

4. Strategies for the Selection of Edge Nodes

We now evaluate the performance of a MCS architecture based on MEC selected according to several strategies. Our analysis first focusses on an architecture composed only by FMECs, which are selected according to three strategies (Section 4.1). We then analyze the performance of a MCS architecture composed only by M2EC (Section 4.2) and, finally, we study the performance of the architecture by combining FMECs and M2ECs together (Section 4.3).

The goal of the selection strategies described in the following sub-sections, is to define an efficient MCS architecture able to gather and to disseminate information from and to the crowd. To this purpose, we measure two metrics that well reflect such goal, namely latency and the number of satisfied requests. The latency measures the effectives of the MCS architecture in contacting edge nodes (both FMEC and M2EC). In particular, it measures the time interval between the task generation and the time at which such task is assigned to a FMEC or M2EC. The number of satisfied requests measures the amount of information successfully retrieved from nodes.

4.1. FMEC Selection Strategies

We evaluate the performance of a MCS architecture only based on FMECs. To this purpose, we consider 3 strategies selecting the FMECs:

- spatial clustering;
- grid deployment;
- random selection.

This section first describes each of the 3 strategies, and secondly it provides a detailed analysis of the performance that we obtain.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [59] is the spatial-based clustering strategy that we adopt. It is an unsupervised learning approach to identify groups of points close in the space. In our case, the points are the GPS locations in WGS84 format of the ParticipAct's users. DBSCAN detects clusters of points according

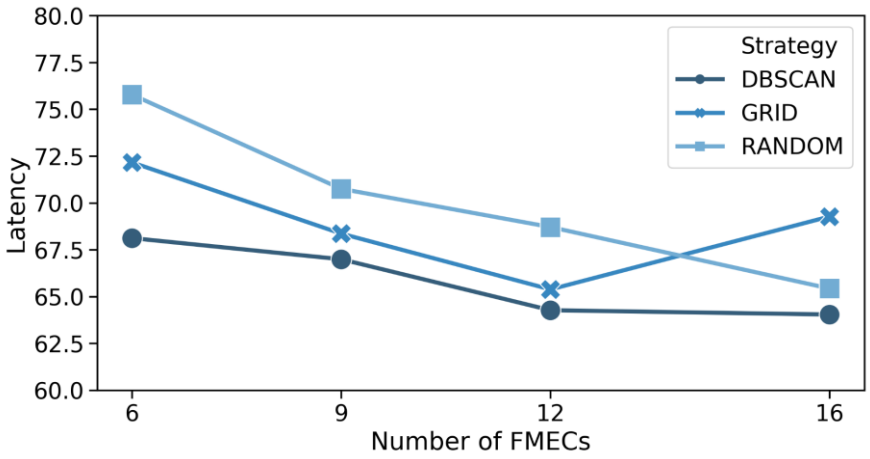


Fig. 8. Average latency with 6, 9, 12 and 16 FMEC and 4 FMEC selection strategies.

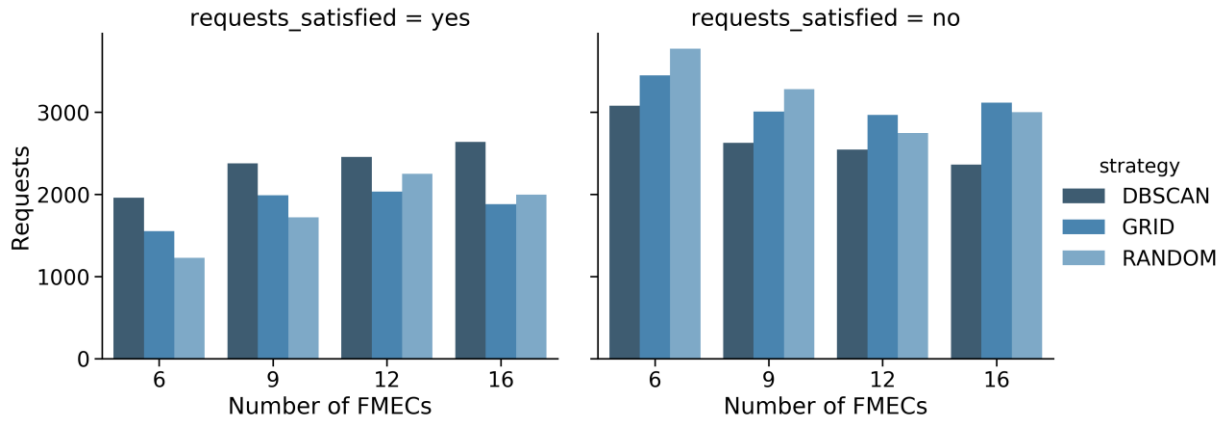


Fig. 9. Average number of satisfied and non-satisfied requests with 6, 9, 12 and 16 FMECs.

to a distance measure and a distance threshold ε: points close to each other according to the distance measure and the threshold are clustered together. DBSCAN assigns to each cluster a label. More specifically, we consider a subset of the ParticipAct dataset of 3-weeks duration, and we use it as input for DBSCAN. The clusters detected by DBSCAN are finally ranked according to the number of distinct users joining the cluster, so that to rank clusters based the users. For every cluster, we compute the centroid of the cluster and, finally we assume the possibility of deploying a FMEC in each of the centroids. We configure DBSCAN with the haversine distance and a minimum number of samples set to 10 points with ε = 50 meters.

We also consider two other strategies based on a grid and a random distribution of the FMECs. The grid strategy selects a set of FMECs deployed according to a regular rectangle grid, while the random strategy arbitrary deploys FMECs over the selected region. For both of the strategies, we set a minimum distance between two FMEC to 500 meters. It is worth to notice that, we restrict the geographical region to the Bologna city center (see Section 3.2), where the majority of user's locations are actually recorded. Figure 7 shows the bounding box of the geographical region that we considered.

We now present the results by studying the average latency and number of satisfied requests by varying the number of FMECs. We vary the number of FMEC in the range 6, 9, 12 and 16, as shown in Figure 8. As expected, the higher the number of FMEC, the lower the latency. In particular, we observe that DBSCAN provides the lower values of latency. The

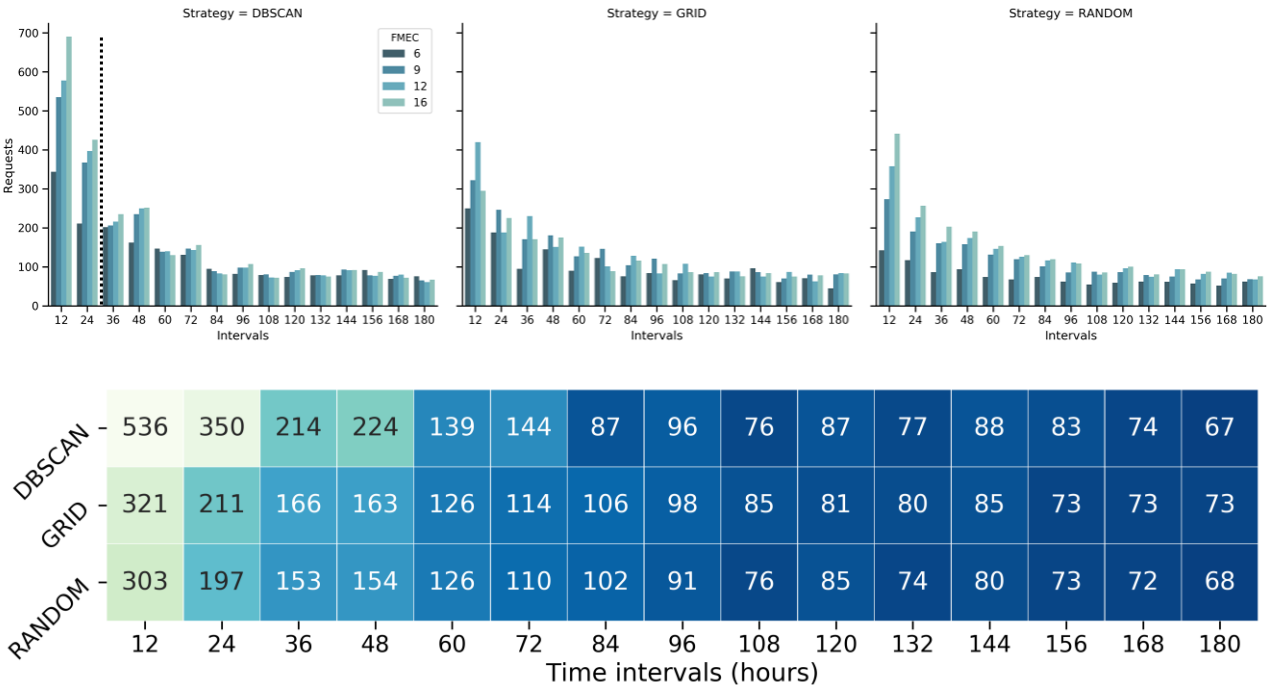


Fig. 10. Average number of satisifed requests with each of the 4 strategies.

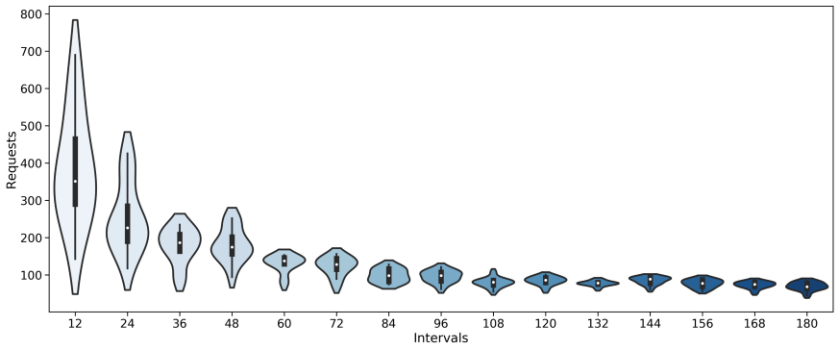


Fig. 11. Distribution of the number of satisifed requests according to the time intervals.

latency with DBSCAN is bound between 68 hours with 6 FMEC and it decreases down to 64 hours with 16 FMEC. The grid and random strategies show a similar trend of latency. Grid is bound between 72 hours with 6 FMECs and 69 hours with 12 FMECs, while the random strategy is bound between 75 and 65 hours.

The results concerning the number of satisfied and non-satisfied requests are presented in Figure 9. The figure shows for each of the 3 strategies, the total number of satisfied and non-satisfied requests. DBSCAN always maximize the number of satisfied requests. In particular, out of the 5*k* requests, DBSCAN satisfies 40%, 43%, 45% and 48% with 6, 9, 12 and 16 FMECs respectively. The grid and random strategies are far from the optimal values. In particular, the grid strategy satisfies 31%, 39%, 40% and 37% with 6, 9, 12 and 16 FMECs respectively. The random strategy provides the lower bound in terms of number of satisfied requests, with an average of 23% less than DBSCAN. We also analyse the number of satisfied requests as a function of the time. Figure 10 shows on the x-axis the time (in hours) required in order to satisfy a request. As described in Section 3, we consider that a request is pending (not satisfied) up to 7.5 days (180 hours), after which it is considered non-satisfied. All the three strategies exhibit the same pattern, the more the time progresses the lower the number of requests are satisfied. As a general observation, the requests are satisfied more probably during the first 12 – 24 hours. DBSCAN is a meaningful example, it is able to satisfies 690 requests with 16 FMECs within the first 12 hours, after which we observe a significant reduction of the requests satisfied after 24 hours (as reported with a dashed line in Figure 10). More specifically, within the first 24 hours, DBSCAN solves 28%, 37%, 39% and 42% of the total requests with 6, 9, 12 and 16 FMECs. After the 24 hours' interval, the number of satisfied requests stabilizes until the limit of 180 hours. Differently, the grid and the random strategies show a more gradual decrease of the number requests satisfied. Both of the strategies tend to satisfy the requests more likely within the first 12 hours, but after such threshold the curves decrease more smoothly. The heatmap in Figure 10, shows a visual representation of the average number of satisfied requests with 6, 9, 12 and 16 FMECs as a function of the interval and of the strategy. It well reflects the considerations made so far, more precisely the lower the interval, the higher the requests satisfied. Finally, we study the probability density of the requests satisfied in each of the time intervals by combining together the results of the 3 strategies. The results are shown in Figure 11 as a set of violin plots. Each violin provides a kernel density estimation of the underlying distribution. Inside each violin, we report a box plot with the median (white dot), 25th, 75th percentile and min and max values. According to the results presented in Figure 11, requests are more likely to be solved within the first 12 to 24 hours, with a mean of approximately 386 and 252 requests satisfied respectively.

4.2. M2EC Selection Strategies

We now study a MCS architecture composed only by M2EC. To this purpose, we adopt the M2EC selection strategies presented in [52]. The strategy relies on two steps. Firstly, it detects communities of users' devices. More specifically, communities are detected by analysing the user's locations during the time. Secondly, the strategy selects a representative

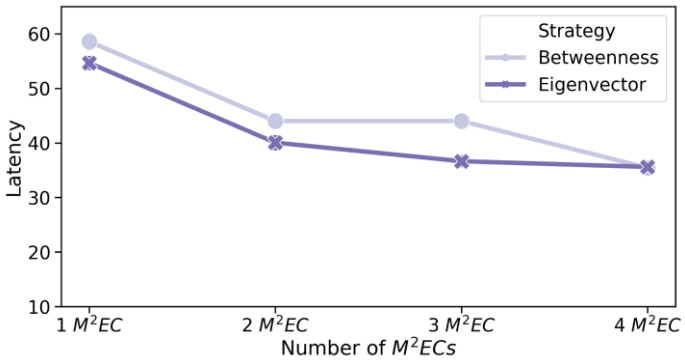


Fig. 12. Average Latency with M^{2}EC selection strategies.

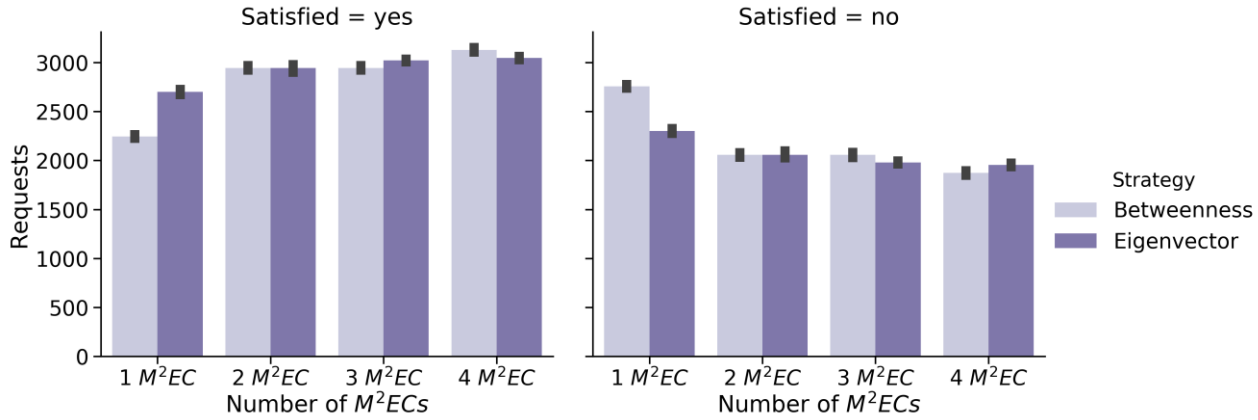


Fig. 13. Number of satisfied and non-satisfied requests with 1 – 4 M^{2}EC.

device for each of the communities identified, such device plays the role of M2EC. The M2EC acts as bridge between the community itself and the devices that the M2EC will encounter, its goal is to collect data from the community's members.

The strategy analyses the list of communities detected and the number of M2EC, and as a result, it outputs the list of devices elected as M2EC. It follows that the number of M2EC selected can never exceed the number of communities. The selection of the M2EC for each community is achieved by measuring the centrality of the community's members. In [52] we refer to two meaningful centrality scores that can be computed for each of the devices in a community, namely betweenness and eigenvector centrality. The betweenness metric measures, for each device, the number of shortest paths crossing a device, the higher the number, the higher the centrality score. In graph theory such measure represents the interaction degree that a node has with other nodes of its network. Differently, the eigenvector centrality measures the influence that a device exerts within the network. The score assigned to a node depends essentially on the score assigned to each node of its neighbourhood. The more the connections of neighbouring nodes, the higher the score of the node itself.

We analyze the latency by varying the number of M^{2}EC as shown in Figure 12. We consider a bunch of 5*k* requests and a TTL of 7.5 days, as done with the metrics measured for FMECs. We restrict our analysis to the top 4 communities detected and therefore we select a number of M^{2}EC ranging from 1 to 4. Such restriction depends on the nature of the dataset considered. Indeed, by running the community detection algorithm with a time resolution of 30 days, we observe that users of ParticipAct tend to cluster to a maximum of 10 different communities. In turn, we kept only the most significant ones by pruning small and unstable communities. As a result, we consider the first 4 communities. The results presented in this section are obtained by executing 10 runs of the same tests so that to obtain a certain degree on statistical confidence. The graph in Figure 12 shows the latency with both of the strategies, betweenness and eigenvector. The curves

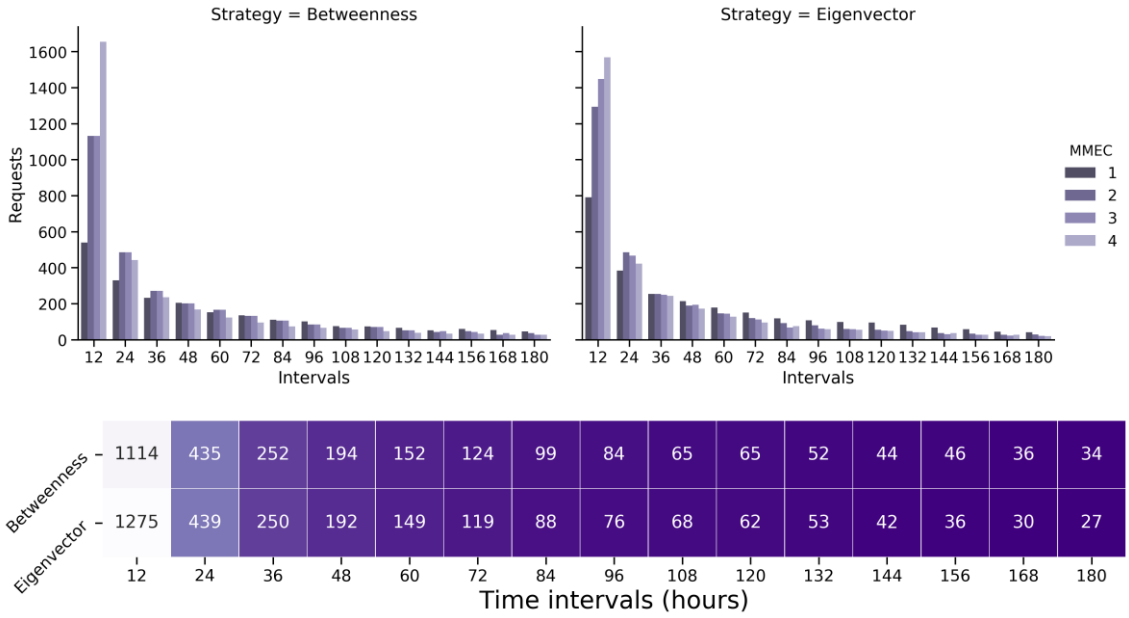


Fig. 14. Average number of satisifed requests according to the time intervals.

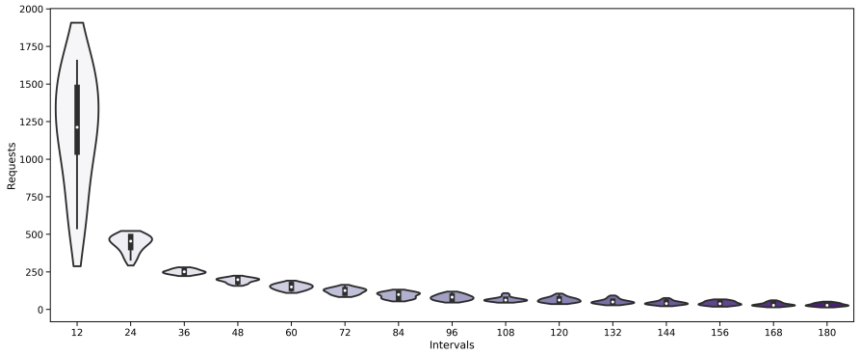


Fig. 15. Distribution of the number of satisifed requests according to the time intervals.

show similar trends, the higher the number of M2EC, the lower the latency. Eigenvector outperforms betweenness up to 3 M2EC, after which both of the strategies are comparable. By adopting the eigenvector algorithm, the latency is always bound between 54 hours with 1 M2EC and 35 hours with 4 M2EC, while the latency for the betweenness algorithm is bound between 58 and 35 hours. For what concerns the number of satisfied and non-satisfied requests, the two algorithms slightly differ, the results are shown in Figure 13. In particular, we observe that both of the strategies obtain similar values. The eigenvector strategy outperforms the betweenness with 1 M2EC with an average of 2700 and 2243 requests satisfied respectively. We analyze the number of satisfied requests for both of the strategies as a function of the time interval. Figure 14 shows on the x-axis the time (in hours) required in order to satisfy a request. As for the FMECs, the highest number of requests is satisfied within the first 24 hours: the eigenvector strategy satisfies 23%, 23%, 38% and 39% of the total requests 1, 2, 3 and 4 M^2EC, while the betweenness strategy satisfies 17%, 32%, 32% and 41% of the total requests. Similarly, to the FMECs analysis, the number of satisfied requests remains almost stable until the limit of 180 hours after the 24-hours intervals. Such behaviour can be also observed with the heatmap in Figure 14. The heatmap shows that the after 24 hours, the colour remains unaltered both for betweenness and eigenvector strategies. We finally show the distribution of the number of satisfied requests as probability density, as shown in Figure 15. As expected, requests are more likely satisfied within 12-hour interval, with a median of approximately 1200 requests (white dot of the first violin). We observe that the performance obtained with M^2EC always outperform the results obtained with FMECs (Figures 8 and Figure 12). More precisely, the optimal FMEC strategy (DBSCAN) obtains values of latency higher than that the two M^2EC strategies. In particular, DBSCAN provides a latency of 64 hours with 16 FMECs, while eigenvector and betweenness 54 and 58 hours respectively by using 1 only M^2EC instead of 16 FMECs. Similar considerations apply also for the total number of satisfied requests. In this case, the 16 FMECs selected with DBSCAN are able to satisfy up to 48% of the total requests, while the 4 M^2ECs selected with eigenvector and betweenness satisfy up to 60% and 62% respectively.

4.3. A Hybrid MEC Architecture

We finally analyse the performance of an architecture composed both of FMECs and M2ECs, we refer to such architecture as Hybrid MEC Architecture. Such architecture provides a considerable level of flexibility, on one hand it is composed by stationary edges deployed on strategic locations, while on the other hand it is enriched by mobile edges able to collect and disseminate data.

We consider FMECs selected according to the DBSCAN strategy and M2ECs selected according to the betweenness strategy. In particular, we show the results by considering 6 FMECs and by varying the number of M2ECs in the range 1 – 4. All the results are obtained by considering a bunch of 5k requests (see Section 3.4).

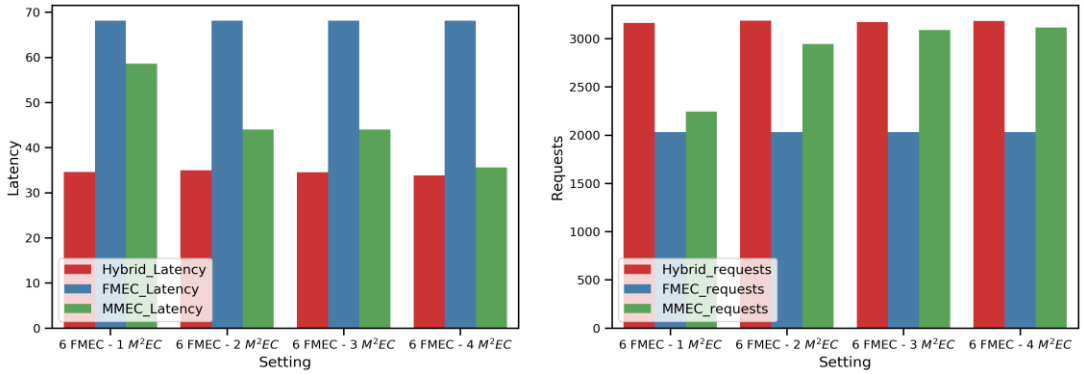


Fig. 16. Latency and number of satisfied requests with 6 FMECs and 1 – 4 M^2ECs.

The results concerning the latency and the number of satisfied requests are shown in Figure 16. The latency obtained with the hybrid architecture is the lowest among the three solutions. In particular, the hybrid architecture provides values ranging from 34 hours with 6 FMEC and 1 M2EC down to 33 hours with 6 FMEC and 4 M2EC. The M2EC strategy obtains values of latency lower than that of the FMEC strategy which provides the lowest performance in terms of the latency. Similar considerations apply also for the number of satisfied requests. In this case the hybrid architecture allows to satisfy approximately 63% of the requests generated, while the M2EC satisfies an average of 56% of the requests and the FMEC strategy satisfies 40% of the requests. The performance reported in Figure 16 lead us to consider the hybrid architecture as an efficient solution for the scenario considered.
We further analyze the contribution of the singles FMECs and M^{2}ECs for a hybrid architecture. In particular, we measure out of the 5k requests how many of them satisfied only by FMECs and by M^{2}ECs. The results are shown in Figure 17. The outer circle reports the results with 6 FMEC and 4 M^{2}EC, while the inner circle the results with 6 FMECs and 1 M^{2}ECs. Moreover, it is worth to notice that the proportion of requests not satisfied by FMECs or by M^{2}ECs (the green ribbon) remains stable along with the different settings. Differently, the requests satisfied by the M^{2}ECs (blue ribbon) increase with the number of M^{2}EC deployed and, proportionally, the contribution of FMEC (red ribbon) decreases.

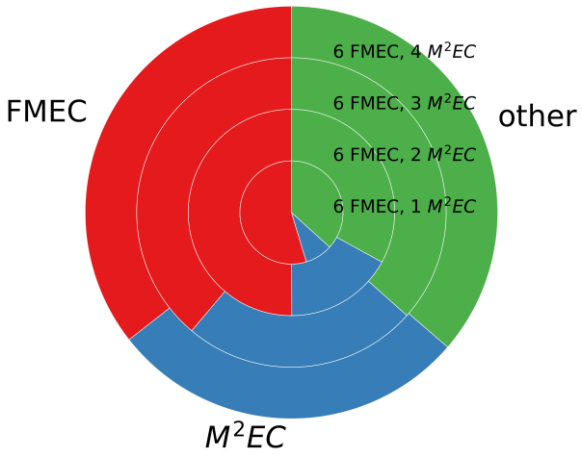


Fig. 17. Average number of requests satisfied with Hybid, FMEC and M^{2}EC architectures.

5. Discussion and Conclusions

The investigation of the synergies between MCS and MEC is still in its early stages, but indeed there are several possible advantages. While MEC provides an infrastructure for the functions of data collection of MCS, MCS itself may provide an extension to the existing MEC infrastructure by means of M2EC. The selection of convenient M2ECs, able to contribute to the overall scalability of the communication infrastructure and even replacing conventional fixed edges, is an open issue. This paper addresses this specific problem and shows that M2EC can replace more than complement FMEC, since they reach users that, in most cases, are later reached by FMEC.
The results we obtained are now boosting further investigations and ongoing research directions. On the one hand, our study is based on a real (although experimental) MCS system and suffers the typical limitations of a datasets that is constrained in time, space, and number of active users. This fact impacts on the results of the clustering and on computation of the centrality scores (that we used as initial mechanism in the selection of M2EC), which can only reflect the behavior of a (relatively) homogeneous community of students in the same town. Therefore, we are now planning to run also simulation experiments with a dataset obtained by combining the real-world mobility traces with a synthetic mobility model so that to complement the results we presented and by removing the above-mentioned limitations. Furthermore, we are evolving our selection strategies to further account the sociality dimension of people. In particular, we are interested in studying evolutionary community detection algorithms able to capture more accurately the clustering evolution of people along the time. In turn, such communities can better reflect the more stable and long-lasting communities to be used for selecting the representative M2ECs.

References

[1] R. K. Ganti, F. Ye, and H. Lei, “Mobile crowdsensing: current state and future challenges”, IEEE Comm. Mag., vol. 49, no. 11, pp. 32-39, 2011

[2] S. Chessa, M. Girolami, L. Foschini, R. Ianniello, A. Corradi, P. Bellavista, “Mobile crowd sensing management with the ParticipAct living lab”, Pervasive and Mobile Computing, 38(2017):200-214.

[3] F. Anjomshoa, B. Kantarci, "Sober-MCS: Sociability-oriented and battery efficient recruitment for mobile CROWD-sensing", Sensors 18(5), 2018. Article number 1593.

[4] M. Tomasoni, A. Capponi, C. Fiandrino, D. Kliazovich, F. Granelli, P. Bouvry, "Why energy matters? Profiling energy consumption of mobile crowdsensing data collection frameworks", Pervasive and Mobile Computing 51, pp. 193-208, 2018

[5] S. Chessa, A. Corradi, L. Foschini, and M. Girolami, "Empowering mobile crowdsensing through social and ad hoc networking", IEEE Communications Magazine, vol 54, no. 7, pp. 108-114, 2016.

[6] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang "A survey on mobile edge networks: convergence of computing, caching and communications", IEEE Access, vol. 5, pp. 6757-6779, 2017.

[7] Mohan, P., Padmanabhan, V.N., Ramjee, R., "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones", 6th ACM Conference on Embedded Networked Sensor Systems, SenSys 2008, Raleigh (NC) United States, Pages 323-336.

[8] Chen, L.-J., Ho, Y.-H., Lee, H.-C., Wu, H.-C., Liu, H.-M., Hsieh, H.-H., Huang, Y.-T., Lung, S.-C.C., "An Open Framework for Participatory PM2.5 Monitoring in Smart Cities", IEEE Access, Volume 5, 6 July 2017, Article number 7970115, Pages 14441-14454

[9] K. M. S. Huq et al., "Enhanced C-RAN Using D2D Network", IEEE Commun. Mag., vol. 55, no. 3, Mar. 2017, pp. 100–107.

[10] P. Bellavista, S. Chessa, L. Foschini, L. Gioia, and M. Girolami, "Human-enabled edge computing : exploiting the crowd as a dynamic extension of mobile edge computing", IEEE Comm. Mag., vol. 56, no.1, pp. 149–155, 2018.

[11] D. Belli, S. Chessa, A. Corradi, G. Di Paolo, L. Foschini and M. Girolami, "Selection of Mobile Edges for a Hybrid CrowdSensing Architecture," *2019 IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, 2019, pp. 1-6.

[12] G. Cardone, A. Cirri, A. Corradi, and L. Foschini, "The ParticipAct Mobile CrowdSensing Living Lab: The Testbed for Smart Cities", *IEEE Communications Magazine*, vol. 52, n. 10, pp. 78-85, 2014.

[13] J. Cao, A. Castiglione, G. Motta, F. Pop, Y. Yang and W. Zhou, "Human-Driven Edge Computing and Communication: Part 1," in *IEEE Communications Magazine*, vol. 55, no. 11, pp. 70-71, Nov. 2017.

[14] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues", *Journal of Network and Computer Applications*, vol. 98, pp. 27 – 42, 2017.

[15] F. Khan, A. Ur Rehman, J. Zheng, M.A. Jan, M. Alam, "Mobile crowdsensing: A survey on privacy-preservation, task management, assignment models, and incentives mechanisms", Future Generation Computer Systems, Volume 100, 2019, Pages 456-472.

[16] T. Taleb, k. Samdanis. B. Mada, H. Flinck, S. Dutta, D Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration", IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1657-1681, 2017.

[17] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G", ETSI White Paper, vol. 11, no. 11, pp. 1-16, 2015.

[18] A. Ahmed and E. Ahmed, "A survey on mobile edge computing", Proc. 10th International Conference on Intelligent Systems and Control (ISCO), pp. 1-8, 2016.

[19] K. Farkas, G. Feher, A. Benczur, and C. Sidlo, "Crowdsending-based public transport information service in smart cities", IEEE Communications Magazine, vol. 53, no. 8, pp. 158–165, Aug 2015.

[20] M. Moreno, A. Skarmeta, and A. Jara, "How to intelligently make sense of real data of smart cities", in International Conference on Recent Advances in Internet of Things (RIoT), pp. 1–6, Apr 2015.

[21] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm", ACM Comput. Surv., vol. 48, no. 1, pp. 1–31, Aug. 2015.

[22] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, "Mobile-edge computing introductory technical white paper", *White Paper, Mobile-Edge Computing (MEC) Industry Initiative*, pp. 1089-7801, 2014.

[23] L. Summers and S. D. Johnson, "Does the configuration of the street network influence where outdoor serious violence takes place? Using space syntax to test crime pattern theory", *Journal of Quantitative Criminology*, vol. 33, no. 2, pp. 397–420, Jun 2017.

[24] S. Rosen, S.-J. Lee, J. Lee, P. Congdon, Z. M. Mao, and K. Burden,"MCNet: Crowdsourcing wireless performance measurements throughthe eyes of mobile devices", *IEEE Communications Magazine*, vol. 52, no. 10, pp. 86–91, 2014.

[25] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications", *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, pp. 1–55, Sep 2014.

[26] H. C. Hsieh, J. L. Chen, A. Benslimane, "5G virtualized multi-access edge computing platform for IoT applications," *Journal of Network and Computer Applications*, vol. 115, pp. 94-102, 2018.

[27] S. Barbarossa, S. Sardellitti, P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks", *IEEE Signal Processing Magazine*, pp. 45-55, 2014.

[28] M. Girolami, P. Barsocchi, S. Chessa and F. Furfari, "A social-based service discovery protocol for mobile Ad Hoc networks", *12th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pp. 103-110, 2013.

[29] Y. Han, Y. Zhu, and J. Yu, "A distributed utility-maximizing algo-rithm for data collection in mobile crowd sensing", in *IEEE Global Communications Conference (GLOBECOM)*, pp. 277–282, Dec 2014.

[30] S. Retal, M. Bagaa, T. Taleb, H. Flinck, "Content delivery network slicing: QoE and cost awareness", *IEEE International Conference on Communications (ICC)*, pp. 1-6, 2017.

[31] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, "Incorporating intelligence in fog computing for big data analysis in smart cities", *IEEE Transactions on Industrial Informatics*, pp. 2140-2150, 2017.

[32] P. Cassarà, F. Potortì, P. Barsocchi and M. Girolami, "Choosing an RSS device-free localization algorithm for Ambient Assisted Living", *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Banff, AB, pp. 1-8, 2015.

[33] N. H. Motlagh, M. Bagaa, T. Taleb, "UAV-based IoT platform: A crowd surveillance use case", *IEEE Communications Magazine*, pp. 128-134, 2017.

[34] E. K. Markakis, I. Politis, A. Lykourgiotis, Y. Rebahi, G. Mastorakis, C. X. Mavromoustakis, E. Pallis, "Efficient next generation emergency communications over multi-access edge computing", *IEEE Communications Magazine*, vol. 55, n. 11, pp. 92-97, 2017.

[35] E. Barka, C. A. Kerrache, N. Lagraa, and A. Lakas, "Behavior-aware UAV-assisted crowd sensing technique for urban vehicularenvironments," in *15th IEEE Annual Consumer CommunicationsNetworking Conference (CCNC)*, Jan 2018, pp. 1–7.

[36] C. Fiandrino, A. B. Pizarro, P. J. Mateo, C. A. Ramiro, N. Ludant, J. Widmer, "openLEON: An end-to-end emulation platform from the edge data center to the mobile user", Computer Communications, vol. 148, pp. 17-26, 2019

[37] Q. H. Nguyen, M. Morold, K. David, F. Dressler, "Car-to-pedestrian communication with MEC-support for adaptive safety of vulnerable road users", Computer Communications, vol. 150, pp. 83-93, 2020.

[38] R. P. Barnwal, N. Ghosh, S. K. Ghosh, S. K. Das, "PS-Sim: A framework for scalable data simulation and incentivization in participatory sensing-based smart city applications", Pervasive and Mobile Computing, vol. 57, pp. 64-77, 2019.

[39] M. Tomasoni, A. Capponi, C. Fiandrino, D. Kliazovich, F. Granelli, P. Bouvry,"Why energy matters? Profiling energy consumption of mobile crowdsensing data collection frameworks, Pervasive and Mobile Computing, Volume 51, 2018, Pages 193-208.

[40] D. Belli, S. Chessa, L. Foschini and M. Girolami, "A Probabilistic Model for the Deployment of Human-enabled Edge Computing in Massive Sensing Scenarios," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2019.2957835.

[41] D. Wu, H. Li, R. Wang, "User characteristic aware participant selection for mobile crowdsensing", *Sensors*, 3959, Nov. 2018.

[42] C. Fiandrino, B. Kantarci, F. Anjomshoa, D. Kliazovic, P. Bouvry, J. Matthews, "Sociability-driven user recruitment in mobile crowdsensing internet of things platforms", *IEEE Global Communications Conference GLOBECOM*, pp. 1-6, 2016.

[43] A. Fazel, B. Kantarci, "SOBER-MCS: Sociability oriented and battery efficient recruitment for mobile crowd-sensing", *Sensors*, 1593, May 2018.

[44] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, "Piggyback crowdsensing (PCS): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proc. of the ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys, 2013.

[45] W. Sherchan, P. P. Jayaraman, S. Krishnaswamy, A. Zaslawsky, S. Loke, A. Sinha, "Using on-the-move mining for mobile crowdsensing", *IEEE 13th International Conference on Mobile Data Management*, pp. 115-124, 2012.

[46] F. Restuccia, S. K. Das & J. Payton, "Incentive mechanisms for participatory sensing: Survey and research challenges", *ACM Transactions on Sensor Networks (TOSN)*, vol. 12, no. 2, 2016.

[47] Y. Chen, H. Chen, S. Yang, X. Gao, F. Wu, "Jump-start crowdsensing: A three layer incentive framework for mobile crowdsensing," *IEEE/ACM 25th International Symposium on Quality of Services*, pp. 1-6, 2017.

[48] J. Lin, M. Li, D. Yang, and G. Xue, "Sybil-proof online incentive mechanisms for crowdsensing", in *IEEE Conference on Computer Communications (INFOCOM)*, Apr 2018.

[49] M. A. Alsheikh, Y. Jiao, D. Niyato, P. Wang, D. Leong, Z. Han, "The accuracy-privacy trade-off of mobile crowdsensing," *IEEE Communications Magazine*, pp. 132-139, 2017.

[50] H. Xiong, D. Zhang, L. Wang, J. P. Gibson, and J. Zhu, "EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, pp. 1–26, Apr. 2015.

[51] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovic, P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions and opportunities," *IEEE Communications Surveys & Tutorials*, 2019.

[52] D. Belli, S. Chessa, L. Foschini, M. Girolami, "A social-based approach to mobile edge computing", *IEEE Symposium on Computers and Communications (ISCC)*, pp. 292-297, 2018.

[53] M. Marjanović, A. Antonić, I. P. Žarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, pp. 10662-10674, 2018.

[54] L. Ma, X. Liu, Q. Pei, Y. Xiang, "Privacy-preserving reputation management for edge computing enhanced mobile crowdsensing," *IEEE Transactions on Services Computing*, 2018.

[55] P. Bellavista, D. Belli, S. Chessa, L. Foschini, "A social-driven edge computing architecture for mobile crowd sensing management", *IEEE Communications Magazine*, vol. 57, pp. 68-73, 2019.

[56] F. Restuccia, S. K. Das & J. Payton, "Incentive mechanisms for participatory sensing: Survey and research challenges", ACM Transactions on Sensor Networks (TOSN), vol. 12, no. 2, 2016.

[57] Y. Chen, H. Chen, S. Yang, X. Gao, F. Wu, "Jump-start crowdsensing: A three layer incentive framework for mobile crowdsensing," IEEE/ACM 25th International Symposium on Quality of Services, pp. 1-6, 2017.

[58] J. Lin, M. Li, D. Yang, and G. Xue, "Sybil-proof online incentive mechanisms for crowdsensing", in IEEE Conference on Computer Communications (INFOCOM), Apr 2018.

[59] Michela Papandrea, Karim Keramat Jahromi, Matteo Zignani, Sabrina Gaito, Silvia Giordano, Gian Paolo Rossi, On the properties of human mobility, Computer Communications, Volume 87, 2016, Pages 19-36.

[60] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," inProc.IEEE Conf. Comput. Commun. (INFOCOM), Hong Kong, Apr. 2015,pp. 2542–2550.

[61] H. Li, T. Li, and Y. Wang, "Dynamic participant recruitment of mobilecrowd sensing for heterogeneous sensing tasks," inProc. IEEE 12thInt. Conf. Mobile Ad Hoc Sensor Syst., Oct. 2015, pp. 136–144.

[62] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruit-ment for mobile crowdsensing over opportunistic networks," inProc.IEEE Conf. Comput. Commun. (INFOCOM), Hong Kong, Apr. 2015,pp. 2254–2262.

[63] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos,"TRAC: Truthful auction for location-aware collaborative sensing inmobile crowdsourcing," inProc. IEEE Int. Conf. Comput. Commun.(INFOCOM), 2014, pp. 1231–1239.

[64] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter:Selecting participants for piggyback crowdsensing under probabilis-tic coverage constraint," inProc. ACM Int. Joint Conf. PervasiveUbiquitous Comput. (UbiComp), 2014, pp. 703–714

[65] L. Pappalardo, F. Simini, G. Barlacchi, R. Pellungrini,"scikit-mobility: a Python library for the analysis, generation and risk assessment of mobility data",arXiv preprint arXiv:1907.07062, 2019.

[66] Luca Pappalardo, Salvatore Rinzivillo, Filippo Simini, Human Mobility Modelling: Exploration and Preferential Return Meet the Gravity Model, Procedia Computer Science, Volume 83, 2016, Pages 934-939.

[67] J. De Benedetto, P. Bellavista, L. Foschini, "Proximity discovery and data dissemination for mobile crowd sensing using LTE direct", Computer Networks, Volume 129, Part 2, 2017, Pages 510-521.