Selection of Mobile Edges for a Hybrid CrowdSensing Architecture

Dimitri Belli
Department of Computer Science
University of Pisa, Italy
dimitri.belli@di.unipi.it,
https://orcid.org/0000-0003-1491-6450

Stefano Chessa
Department of Computer Science
and Istituto di Scienza e Tecnologie
dell'Informazione CNR-ISTI
University of Pisa, Italy
stefano.chessa@di.unipi.it,
https://orcid.org/0000-0002-1248-9478

Antonio Corradi
Dipartimento di Informatica: Scienza e Ingegneria
University of Bologna, Bologna, Italy
antonio.corradi@unibo.it,
https://orcid.org/0000-0002-5107-1023

Giampiero Di Paolo
Department of Computer Science
University of Pisa, Italy
giamdip@gmail.com

Luca Foschini
Dipartimento di Informatica: Scienza e Ingegneria
University of Bologna, Bologna, Italy
luca.foschini@unibo.it,
https://orcid.org/0000-0001-9062-3647

Michele Girolami
Istituto di Scienza e Tecnologie
dell'Informazione CNR-ISTI
National Council of Research, Pisa, Italy
michele.girolami@isti.cnr,it,
https://orcid.org/0000-0002-3683-7158

***Abstract* — Mobile crowdsensing aims at the collection of sensor data on the environment by leveraging personal devices, usually smartphones. Its popularity is due to the ability of reaching capillary even the most remote areas (provided humans live there), with no infrastructure costs. This is possible because it leverages on existing 4G/5G communication infrastructures that are now rapidly evolving towards edge computing models. In this work we address the synergy between mobile crowdsensing and multi-access edge computing by analysing and assessing strategies for the selection of fixed and mobile edges to support the collection of mobile crowdsensing data.**

Keywords — mobile crowdsensing, multi-access edge computing, clustering, sensor data collection

I. Introduction

Mobile Crowdsensing (MCS) [1] technologies are becoming a precious source of information for the optimization and management of smart cities, due to their ability to capillary collect large amount of data of a wide range of types. The key of their success lies in the wide diffusion of personal devices (wearable, smartphones etc…) that people carry with them almost all the time, and that embed a wide range of sensors and powerful data fusion algorithms that enable an even wider range of sensing applications. In recent years, MCS has inspired research towards aspects, like the optimization of the energy consumption of the users' devices [2, 3] and of the task assignment [4] or the massive involvement of MCS volunteers [5].

However, MCS technologies do not exist alone. They are possible because Internet provides ubiquitous connectivity, and powerful servers in the cloud can manage the flow of data from billions of devices worldwide. More recently, this infrastructure is becoming hierarchical, where localized, fixed servers at the edge of the network, called in the following Fixed Multi-access Edge Computing nodes (FMECs) provide a first level of data filtering, aggregation, analysis and storage, to reduce the burden on the network core and on the remote cloud servers. This infrastructure is commonly known as Multi-access Edge Computing (MEC) [6].

A large amount of work in the last few years has focused on the development of synergies to achieve a strict integration between MCS and MEC, for MCS applications devoted to data collection for off-line, big-data analytics. These applications require a huge effort of data collection and transmission to the personal devices, while, at the same time, do not have stringent requirements in terms of latency. A conventional use of MCS and MEC technologies in this case, would require the personal devices a continuous activity of collection and data transmission through broadband wireless links even to reach the fixed MEC. However, considering that the latency requirements are rather relaxed in this case, we consider architectures in which the communication with the FMEC can also happen opportunistically by short range radio interfaces (like Bluetooth or WiFi) when the user becomes in range with the edge itself [7]. In the effort of a further reduction of communication overhead, [8] adopted bloom filters to reduce the number of redundant data transmitted to edge nodes. Furthermore, in order to reduce the deployment and maintenance costs of FMEC [9], the users' devices themselves may be configured by the MCS platform to act temporarily as mobile edges. We define them as Mobile Multi-access Edge Computing nodes (M^2ECs) to stress their ability to opportunistically collect (during their roaming) data from other user devices that come into their short-range radio interfaces [10]. That allows to trade the costs of broadband communications (both in terms of energy and subscription costs) with latency in the communications.

Along this trend of research, this paper focuses on the strategies for the selection of FMEC and M^2EC and on their synergies. We build over our previous work [11], in which we explored the opportunity of selecting M^2EC based on the sociality of users. Differently than that work, however, in the present work we perform a comprehensive evaluation of several alternative strategies both for the selection of FMEC and M^2EC, to identify the best combination. In particular, we consider a selection of FMECs based on the mobility of the MCS users and we assess clustering algorithm for the identification of the best places for the deployment of these edges. Then, we consider a selection of M^2ECs based on the sociality of the MCS users, by leveraging community detection and centrality measures to select the most "social" users to promote as M^2ECs. Finally, we analyze the synergetic behavior of both kind of edges.

Our simulation experiments conducted over a dataset obtained from real data of an MCS system, clearly show an improvement of clustering algorithms as DBSCAN against heuristics in the selection of FMEC, and a better performance of heuristics based on betweenness in the selection of M^{2}EC. The simulations concerning the combined use of FMEC and M^{2}EC show that the latter can replace more than complement FMEC, since in many cases they provide a data collection service to users later receive the same service from the FMEC.

The rest of the paper presents the hybrid architecture combining FMEC and M^{2}EC in Section II, the simulation methodology in Section III, and the results of the simulations with FMEC alone, of M^{2}EC alone and in combination in sections IV, V and VI, respectively. Section VII draws the conclusions.

II. A Hybrid Architecture for Human Edge Computing

This section provides some more background material about our hybrid architecture based on our previous work on Human-Enabled Edge Computing [10]. That model proposes to complement FMECs proxies, i.e., static base-stations which only act as intermediary between devices and the cloud, with M^{2}EC acting as FMEC at predetermined interval of time to post logical bounded regions in which people tend to stay for a while. In particular, as shown in [10] the monitoring of human movements leveraging MCS can ease the identification of strategic hotspots where to install M^{2}EC and then, leveraging local one-hop communications and store-and-forward principle, it is possible to enable the migration of data from FMECs to M^{2}EC and vice versa. In the following we report more details about our platform and the combined use of MCS, FMEC, and M^{2}EC.

An MCS platform implements a broad-range community sensing paradigm that consists of three components: individuals, devices and centralized, cloud-based servers. The individuals, who express their willingness to take part in the MCS platform and to its campaigns, wear mobile devices equipped with sensors, short-range communication interfaces and the MCS mobile application. The MCS mobile application can collect data autonomously through sensors or with the support of the user. Conventionally, data stored within devices' memory is forwarded to a remote server for storage or for further processing in two possible ways: (i) via broadband communication (e.g. 4G LTE or 5G) to directly connect the mobile device to the server on the cloud; (ii) via MEC proxies, herein called FMECs, that may be present in the territory as an additional access layer between the cloud-level and the core network.

It should be observed that, while the broadband communications are long-range and then usually available regardless the position of a mobile device (but they do have a higher cost for the users), the communications with the FMEC may instead rely on short-range communications (for example based on Wi-Fi, with a reduced spatial coverage) with lower battery of mobile user devices and a reduction of the communication burden over the core network. In a Human-Enabled Edge Computing architecture, a further set of mobile MECs, namely M^{2}ECs, can be implemented by users' mobile devices, carrying out the same functions of FMECs. M^{2}ECs are selected from those carried by individuals and may collect opportunistically all the data produced by other devices that come in the range of their short-range radio interfaces. M^{2}ECs are chosen among the users' devices, based on their opportunity to meet other users' devices during their travels. M^{2}EC thus introduce a so-called social coverage [11], which is defined as the set of users' devices met by a M^{2}EC over a period of time.

Finally, for the sake of space limitation in this paper we do not report more information about the inner architecture of our FMEC and M^{2}EC nodes that is currently based on modern virtualization and containerization technologies, for which we refer interested readers to [10].

III. Experimental Methodology

Our experiments rely on the ParticipAct CrowdSensing project [12] carried out by approximately 170 students from the University of Bologna, Italy. Volunteers were equipped with an Android smartphone provisioned with the ParticipAct mobile app able to track their location every 2.5 minutes through the Google location APIs. The location is obtained by the synergistic use of information coming from Wi-Fi Hot Spot coordinates, GPRS and cell phone base stations.

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 on the widest presence of the participants in the territory, which occurred in spring time.

In order to evaluate the performance of the hybrid architecture introduced in Section II, we implemented a Python-based CrowdSensing simulator able to mimic the collection of information from the crowd. The simulator offers the possibility of selecting a number of FMEC and M^{2}EC according to a given strategy. We currently support DBSCAN, grid and random strategies for FMEC selection (see Section IV), and a M^{2}EC selection strategy [11] based on two well-known graph centrality measures, namely betweenness and eigenvector centrality (see Section V). The simulator also generates an arbitrary number of requests. A request represents any kind of data a device generates and that needs to be uploaded to the Cloud. Requests are generated by randomly selecting devices not previously elected as FMEC or M^{2}EC. Each request is assigned to a device with a given timestamp and a TTL. The timestamp denotes the time at which the device produces new data, while the TTL is the maximal amount of time before the request is uploaded by using a direct, broadband communication link to the cloud. Requests are generated only during daily hours, from 9.00 AM to 8 PM and during the first 3 weeks of March 2014, so that to enable requests generated during the first 3 weeks of March to run for completion by the end of March.

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 Cloud by means of short-range network interfaces. In particular, we assume devices can interact with interfaces such as Bluetooth or WiFi. To this purpose, we extract from the GPS traces of ParticipAct a more compact trace, referred to as the co-location trace. A pair of users is co-located if their devices can exchange information with a short-range interface. Therefore, the pair must lay on the same place at the same time for a time interval. We assume that users are co-located within a

TABLE I.

Property	Value
Number of participants	170
Observation period	March 1st – 30th 2014
Number of requests	5×10^3
Request time interval	9.00 AM – 8.00 PM
TTL	7 days
Co-location distance	100 *m*
FMEC strategies	DBSCAN, GRID, RANDOM
M^2EC strategies	Betweenness, Eigenvector centrality

distance up to 100 meters, an acceptable distance for wireless outdoor communications.

Finally, we assess the performance of the hybrid architecture proposed by studying how varies *latency* and the *number of satisfied requests* by increasing the number of FMEC, M^2EC and with a combination of them. The latency measures the time interval between the request generation and the time at which the request is assigned to a FMEC or M^2EC (hence the request is sent to the Cloud). The number of requests satisfied assesses the amount of information generated by the crowd that the architecture can deliver to the Cloud. Table 1 summarizes our experimental settings.

IV. Strategies for FMEC Selection

The hybrid architecture we propose relies only on the FMEC (Fixed MEC, see Section II). We identify a number of strategies for electing a node acting as FMEC. Such strategies span from spatial clustering to randomly selection. This sections first describes each of the strategies proposed for FMEC selection and, second it reports the performance we obtained.

We adopt a spatial-based strategy, namely *DBSCAN* (Density-Based Spatial Clustering of Applications with Noise). It is a spatial clustering algorithm. Given a set of points located in the space, the algorithm clusters the closest points according to a distance measure and a distance threshold ε. The DBASCAN strategy analyses the user's locations given by the ParticipAct dataset over a period of 3 weeks, and it returns the existing clusters. Such clusters correspond to the highest populated regions; the clusters are finally ranked according to the number of users inside each cluster so that to select a subset of them. For every cluster, we identify its centroid where we assume to deploy a FMEC. We configure DBSCAN with the haversine distance and a minimum number of samples set to 10 points and ε = 50 *meters*.

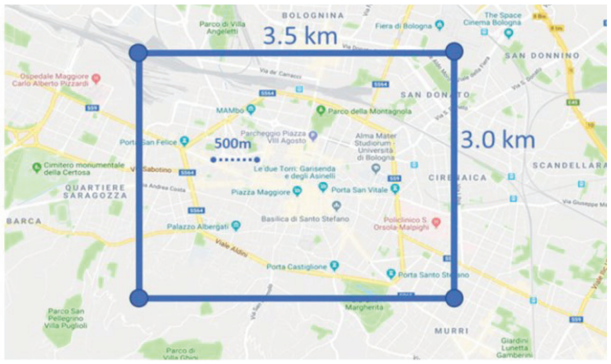

Fig. 1. Geographical region for grid and random strategies.

We also consider two other strategies based on a *grid* and a *random* distribution of FMEC. The grid strategy selects a set of FMEC deployed according to a regular rectangle grid, while the random strategy arbitrary deploys FMEC over the selected region. For both of the strategies, we select a minimum distance between a pair of FMEC set to 500 meters. Moreover, we bound the FMEC selection to the Bologna city centre (see Section III), where the majority of user's locations are recorded. Fig. 1 shows the bounding box of the geographical region we considered for the grid and random strategies.

We now present the results we obtained by studying the latency and satisfied requests by varying the number of FMEC. We vary the number of FMEC in the following range [6, 9, 12, 16]. Fig. 2 and Fig. 3 show the average latency (in hours) and the satisfied requests, respectively. As expected, the higher the number of FMEC the lower the latency. In particular, we observe that DBSCAN outperforms the grid and random strategies. The 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, they are bound between 72 and 75 hours respectively with 6 FMEC and 69 and 65 hours respectively with 16 FMEC. Concerning the random strategy, we show the results after several runs of the random strategy so that to obtain stable values of latency and of the satisfied requests. Results for the satisfied requests are also shown in Fig. 3. The DBSCAN strategy reports higher performance

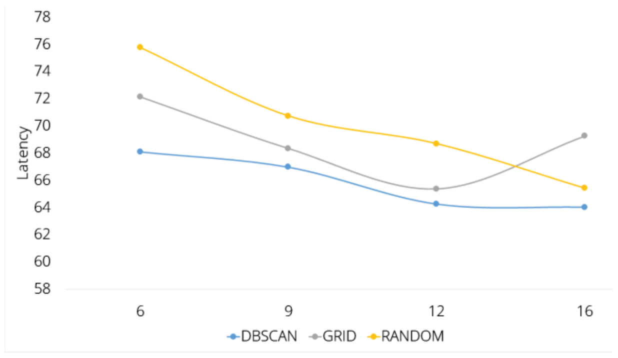

Fig. 2. Performance results for FMEC selection with DBSCAN, grid and random strategies: latency of satisfied requests

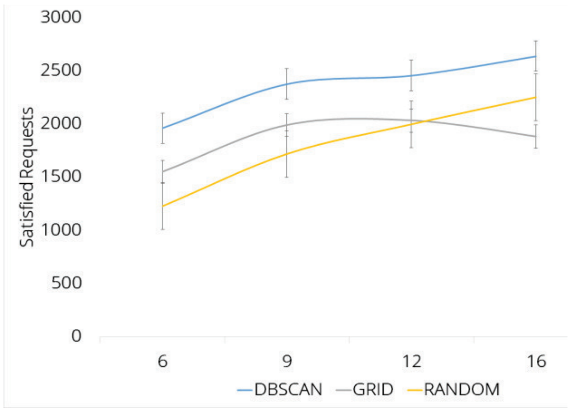

Fig. 3. Performance results for FMEC selection with DBSCAN, grid and random strategies: number of requests satisfied.

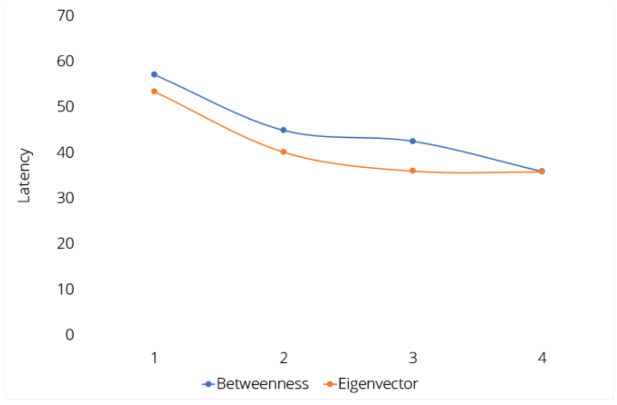

Fig. 4. Performance results for M^{2}EC selection with betweenness and eigenvectory centrality measures: latency of satisfied requests

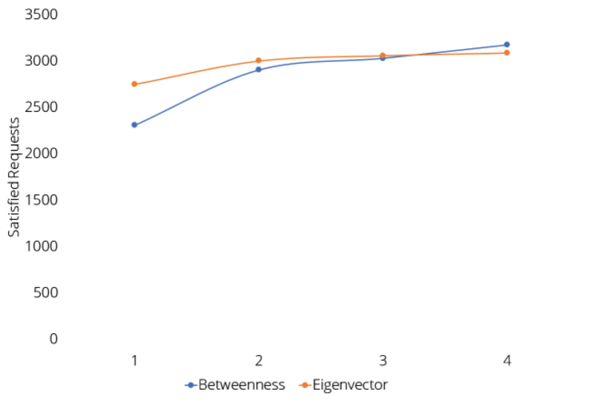

Fig. 5. Performance results for M^{2}EC selection with betweenness and eigenvectory centrality measures: number of satisfied requests

with respect to the grid and random strategies, in particular the number of requests satisfied varies from 2031 (40%) up to 2414 (48%). Differently, grid and random strategies obtain the lowest values with 1551 to 1882 and 1227 to 2252 requests satisfied respectively. It is worth to notice that the grid strategy decreases its performance after a certain number of FMEC selected, from our results we observe that with 12 FMEC the random strategy performs better than that the grid-based deployment.

Finally, we show in Fig. 6 the strategy with the lowest latency and the highest number of satisfied requests, namely DBSCAN, with 16 FMEC. The figure reports for each values of latency *x*, the number of requests *y* satisfied in *x* hours. Such strategy satisfies 2414 of the 5000 requests generated (48%). The majority of the requests are satisfied within 36 hours, in particular 690 within 12 hours, 426 within 24 hours, and 235 within 36 hours. From the results presented in this section, we derive that the MCS architecture based only on FMEC is able to satisfy at maximum 48% of the total number of requests generated with an average latency of 64 hours. Sections V and VI present an enhanced MCS architecture that combines FMEC and M^{2}EC.

V. Strategies for M^{2}EC Selection

The MEC architecture analyzed in Section IV relies on fixed MEC only. We now evaluate an MCS architecture in which MEC are bound to users with their own mobility and sociality, we refer to them as Mobile MEC, M^{2}EC.

We refer to the M^{2}EC selection strategy presented in [11]. Such strategy relies on two steps. Firstly, the algorithm detects the dynamic communities of users. Communities are clusters of users' devices evolving along with the time. Secondly, the algorithm selects a representative device from each community, acting as *bridge* between the community itself and the rest of the population. The algorithm takes as input the list of communities detected and the desired number of M^{2}EC, and as a result it outputs the devices elected as M^{2}EC. It follows that the number of M^{2}EC selected can never exceed the number of communities detected. In order to select a M^{2}EC for every community, the algorithm measures the centrality of the community's members (in our implementation we rely on the betweenness and on the eigenvector centrality measures). More precisely, the algorithm measures the capability of such devices to establish bindings between the community's members and devices not part of such community. Briefly, the betweenness is a centrality measure based on shortest paths, that is, a node is assigned a given score on the basis of the number of shortest paths that cross it. In network theory this value represents the interaction degree that a node has with other nodes of its network. Differently, the eigenvector centrality is a measure of the influence that a node exerts within the network. The score assigned to a node depends essentially on the score assigned to each node of its neighborhood. The more the connections of neighbour nodes the higher the score of the node itself. We refer to [11] for a detailed description of the M^{2}EC selection strategy.

In the following we present joint results obtained through the execution of the algorithm with both betweenness and eigenvector centrality. As for the FMEC (see Section IV), each test with M^{2}EC includes a number of requests to be satisfied (5000 requests). The request expiration time, TTL is set to 7.5 days. The graphs in Fig. 4 and Fig. 5 show the average latency and average number, respectively, of satisfied requests as a function of the number of M^{2}EC selected. We restricted 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 (ParticipAct). 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 pruning small and unstable communities.

The graph in Fig. 4 shows the latency with both strategies. The curves have similar trends, the higher the M^{2}EC, the lower the latency. Eigenvector outperforms betweenness up to 4 M^{2}EC, after which both of the strategies are comparable. With the eigenvector measure the latency is bound between 53 hours with 1 M^{2}EC and 35 hours with 4 M^{2}EC, while the latency for the betweenness measure is bound between 57.16 and 35 hours. For what concerns the average number of satisfied requests, the two strategies slightly differ. In particular, we observe that the betweenness increases the number of requests satisfied, ranging from 2312 (46%) with 1 M^{2}EC up to 3175 (63%) with 4 M^{2}EC, while the eigenvector ranges from 2753 (55%) with 1 M^{2}EC up to 3089 (61%) with 4 M^{2}EC.

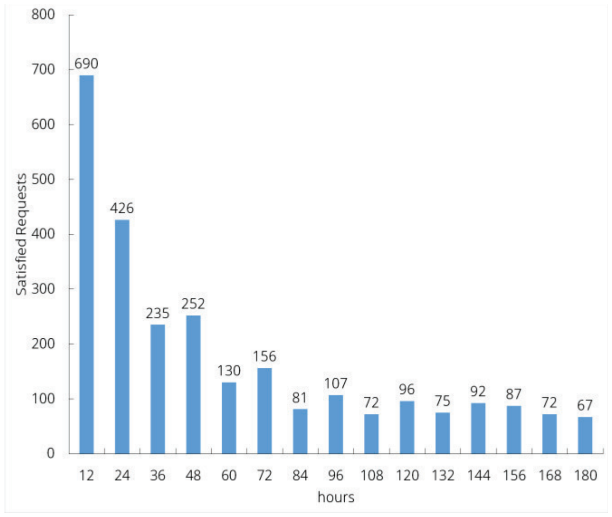

Fig. 6. Latency of the satisfied requests with DBSCAN and 16 FMEC.

When comparing Fig. 4 with Fig. 2 and Fig. 5 with Fig. 3, we observe that the M^{2}EC strategies always outperform the FMEC ones. More precisely, the optimal FMEC strategy (DBSCAN) obtains values of latency higher than that the two M^{2}EC strategies, in particular 64 hours with 16 FMEC (DBSCAN) versus 53 and 57 hours with eigenvector and betweenness respectively, but with only 1 M^{2}EC selected. Similar considerations apply also for the number of satisfied requests. In this case the 16 FMEC selected with DBSCAN are able to satisfy up to 48% of the total requests, while the 4 M^{2}EC selected with eigenvector and betweenness satisfy up to 63% and 61% respectively.

Fig. 7 reports the latency of the satisfied requests with 4 M^{2}EC selected with the betweenness measure. The trend shows that the majority of the requests is satisfied during the first 12 hours, in particular 1618 out of 3175 (50%). From the results presented in Fig. 3 and Fig. 5, we conclude that the selection of members of the communities as M^{2}EC reduces significantly the latency and increases the requests satisfied as well.

VI. Evaluation of the Hybrid Architecture

We now present a combination of the strategies introduced in Section IV and V. In particular, we evaluate an MCS architecture made up of FMEC and M^{2}EC. To this purpose, we select the DBSCAN as the best FMEC selection strategy

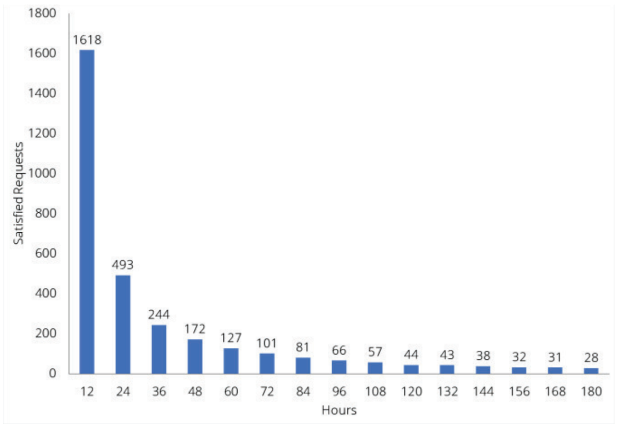

Fig. 7. Latency of the satisfied requests with betweenness and 4 M^{2}EC.

and the betweenness centrality as the best M^{2}EC selection strategy. We present the latency and the average number of satisfied requests by varying the number of M^{2}EC selected in the range 1–4 (as discussed in Section V). Concerning the number of FMEC, we consider 6 FMEC so that to obtain the maximum performance from the fixed MCS architecture. We refer to such setting as the hybrid MCS architecture.

Fig. 8 and Fig. 9 show the comparison among hybrid, FMEC only and M^{2}EC only architectures. From Fig. 8 it is seen that the latency obtained with the hybrid architecture is the lowest among the three solutions. In particular, the hybrid architecture reports latency values ranging from 34 hours with 6 FMEC and 1 M^{2}EC down to 33 hours with 6 FMEC and 4 M^{2}EC. We also observe that the M^{2}EC strategy obtains latency values comparable with respect to the hybrid architecture when deploying 4 M^{2}EC. Similar considerations apply also for the number of satisfied requests (in Fig. 9). In this case the hybrid architecture allows to satisfy approximately the 63% of the requests generated. Lower values are obtained with FMEC architecture only, while similar results to the hybrid architecture are also obtained with the M^{2}EC architecture only, selecting 4 M^{2}EC.

The performance analysis reported in Fig. 8 and Fig. 9 leads us to consider the hybrid architecture as the best choice in scenario in which is required to reduce the number of M^{2}EC. In fact, despite the restricted number of M^{2}EC (1 to 3) the overall performance increases with respect to architectures only made up of FMEC or M^{2}EC only. Moreover, when the

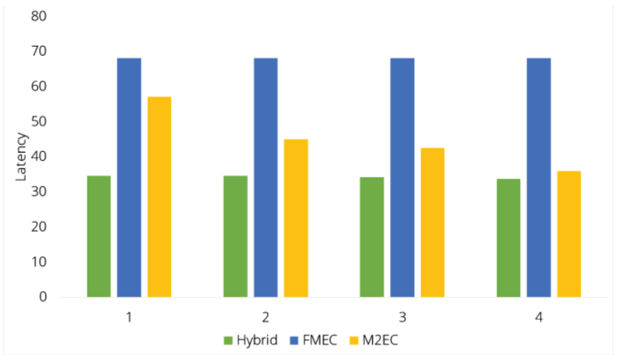

Fig. 8. Performance of hybrid, FMEC and M^{2}EC architectures: latency of satisfied requests

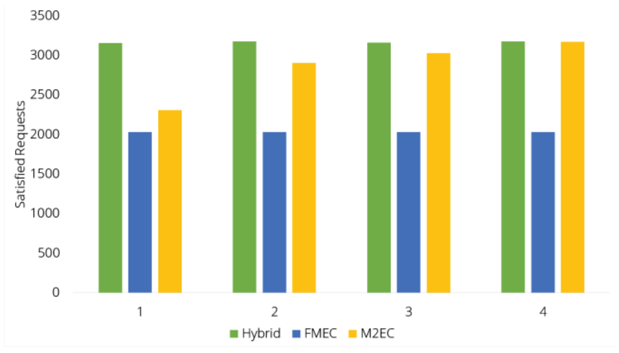

Fig. 9. Performance of hybrid, FMEC and M^{2}EC architectures: number of requests satisfied.

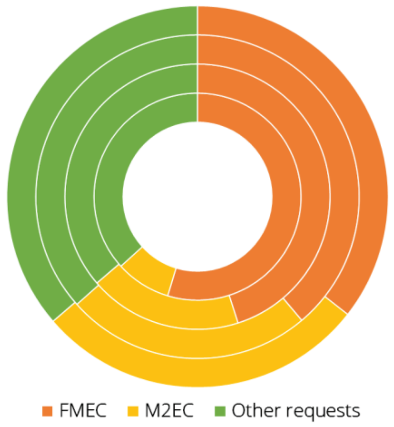

Fig. 10. Details of the contribution of FMEC and M^{2}EC: from 6 FMEC and 1 M^{2}EC in the inner circle to 6 FMEC and 4 M^{2}EC in the outer circle

number of M^{2}EC increases, we observe that the M^{2}EC contribution is higher than that the contribution given by FMEC. Therefore, a pure M^{2}EC architecture might be the best option. We finally show in Fig. 10 the specific contribution of FMEC and M^{2}EC in terms of the proportion of requests satisfied. The inner circle shows the proportion with 6 FMEC and 1 M^{2}EC, the outer circle shows the proportion with 6 FMEC and 4 M^{2}EC. It is worth to notice that the proportion of requests not satisfied by FMEC or M^{2}EC (the green ribbon) remains stable along with the different settings. Differently, the requests satisfied by M^{2}EC (yellow ribbon) increase with the number of M^{2}EC deployed and, proportionally, the contribution of FMEC (orange ribbon) decreases.

VII. Discussion and Conclusions

The investigation of the synergies between MCS and MEC is still in an early stage. On the one hand, MEC provides an infrastructure for the functions of data collection of MCS, while, on the other hand, MCS itself may provide an extension to the existing MEC infrastructure by means of M^{2}EC. This second aspect, however, is still mostly unexplored, and this work moves in this precise direction. Specifically, we address the problem of how M^{2}EC can complement and, to same extent, even replace conventional fixed edges of a MEC infrastructure. The results presented in our study show that M^{2}EC can replace more than complement FMEC, since they reach users that, in most cases, are later reached by FMEC.

However, although being a significant first step, this work still leaves space for several improvements. In fact, being based on a dataset obtained from a real (although experimental) MCS system, our experiments suffer of the typical limitation of such datasets of being constrained in time and space, and in the number and kind of users. This fact impacts the results of the clustering and of the community detection algorithms (that we used as primitive mechanisms in the selection of M^{2}EC), which can only reflect the behaviour of a (relatively) homogeneous community of students in the same town. The consequence is that we cannot conclude with this study that M^{2}EC can really replace FMEC in all cases, because the communities of users and the places where they meet may be too limited and overlapped, thus resulting in an overlapping of the space of activities of M^{2}EC and FMEC. For this reason, we are now planning simulation experiments also with synthetic datasets that may complement these results by removing the above-mentioned limitations.

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

[3] 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

[4] 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.

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

[8] M. Marjanovic, A. Antonic, I. P. Zarko, "Autonomous data acquisition in the hierarchical edge-based MCS ecosystem", 6th IEEE Int. Conf. on Future Internet of Things and Cloud Workshops, W-FiCloud 2018; Barcelona; Spain; 6-8 August, pp. 34-41

[9] 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.

[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, L. Foschini and M. Girolami, “A Social-Based Approach to Mobile Edge Computing”, IEEE International Symposium on Computers and Communications (ISCC), Natal, Brazil, 25-28 June 2018, pp. 292-297

[12] G. Cardone, A. Cirri, A. Corradi, L. Foschini, The participact mobile crowd sensing living lab: the testbed for smart cities, IEEE Commun. Mag. 52 (10) (2014) 78–85