Geo-partitioning of MEC resources

Mathieu Bouet
Thales Communications & Security
4 avenue des Louvresses
Gennevilliers, France 92230
mathieu.bouet@thalesgroup.com

Vania Conan
Thales Communications & Security
4 avenue des Louvresses
Gennevilliers, France 92230
vania.conan@thalesgroup.com

ABSTRACT

Mobile Edge Computing (MEC) is an emerging technology that aims at pushing applications and content close to the users (e.g. at base stations, access points, aggregation networks) to reduce latency, improve quality of experience, and ensure highly efficient network operation and service delivery. It principally relies on virtualization-enabled MEC servers with limited capacity at the edge of the network. One key issue is to dimension such systems in terms of server size, server number and server operation area to meet MEC goals. In this paper, we propose a graph-based algorithm that, taking into account a maximum MEC server capacity, provides a partition of MEC clusters, which consolidates as many communications as possible at the edge. We use a dataset of mobile communications to evaluate it with real world spatio-temporal human dynamics. In addition to quantifying macroscopic MEC benefits, the evaluation shows that our algorithm provides MEC area partitions that largely offload the core, thus pushing the load at the edge (e.g., with 10 small MEC servers around 55% of the traffic stay at the edge), and that are well balanced through time.

CCS CONCEPTS

•**Networks** → *Network design principles;*

KEYWORDS

MEC, server clustering, load-balancing.

1 INTRODUCTION

Mobile Edge Computing (MEC - also know as Multi-access Edge Computing [6], and similar to fog computing [7]) has emerged as a key enabling technology for realizing the IoT and 5G visions. It aims at reducing latency and ensuring efficient network operation and service delivery and pushing content and services close to the users. Numerous MEC applications are already envisioned and investigated, for example: chat/video analytics, video acceleration, augmented reality, location-based services, connected cars, and IoT gateways [8].

In a MEC deployment *MEC servers* are postionned in the infrastructure close to the edge of the network (see Fig. 1): they are small-scale datacenters with low to moderate resources collocated with the base stations, access points and/or placed in the access/aggregation network. They leverage virtualization to support MEC applications run as virtual machines, containers, unikernels etc. The purpose of MEC servers is to host as many applications as possible at the edge to improve latency and alleviate congestion in the core. MEC thus performs application and network offloading from the *core data center* on to the edge.

The central decision in a MEC design is to decide which users, applications and share of traffic should be handled by the MEC servers. To address this key issue we name *MEC clusters* the area, and by extension the base stations and the users in the area, served by a MEC server. Indeed the efficiency of a MEC system heavily depends on such aspects as the distribution of communications and workloads in time and space. Its cost depends on server density, capacity and interconnection. Imbalanced MEC clusters that handle highly different traffic volumes would lead to an inefficient use of resources and to unequal Quality of Experience (QoE). Put in terms of MEC clustering, the key question is thus: how to have an efficient partition of MEC areas? From this, a placement of MEC servers can be derived.

The problem is aggravated by the well documented fact that mobile traffic is very dependent on time and locality. Indeed mobile

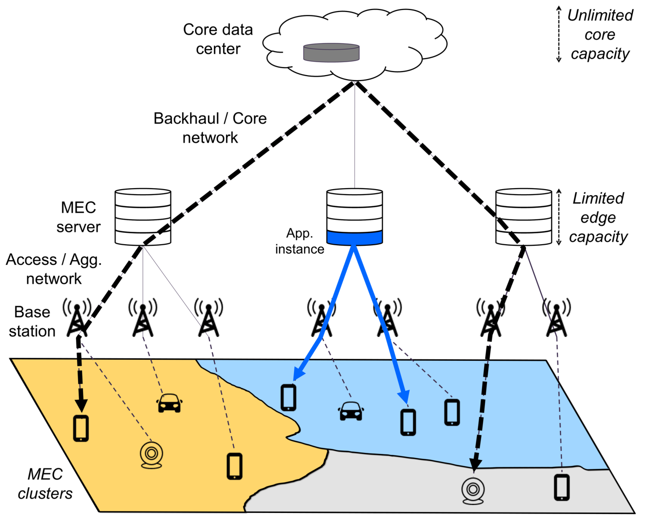


Figure 1: MEC deployment: tasks and applications (e.g., video analytics, video customization) are mainly offloaded onto MEC servers at the edge of the network to reduce latency and offload the core network. Note that it can be an n-level architecture.

MECOMM '17, August 21, 2017, Los Angeles, CA, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN: 978-1-4503-5052-5/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3098208.3098216

communications are spatially distributed according to the population density and activity, which vary in time. For instance, the mobile traffic in the business areas differ from the mobile traffic in the transport, residential and entertainment areas [11, 14, 17]. As it was shown by Qazi et al. [12], in a MEC perspective, such variations have a direct impact on the load of the potential MEC servers. In addition, it was shown by Tastevin et al. [14] that mobile communications in an urban environment have a high spatial locality - they tend to follow a power law, which motivates a local consolidation of applications at MEC servers. Such properties will be amplified with the realization of the IoT and 5G visions [8].

The large-scale dimension of MEC systems and mobile communications makes classic analytical and simulation-based approaches inapplicable. In this paper, we thus investigate a graph-based method. We propose an algorithm that, based on the spatial distribution of the communications, finds a MEC partition that favors application instantiation at the edge instead of at the core. The resulting clusters correspond to MEC areas. Our algorithm takes into account the maximum server capacity, that we express as the maximum number of served communications, but that can be easily expressed in terms of resources (CPU, storage...) or application instances. We evaluate it using a dataset of mobile communications in a city provided by a mobile operator. We first show that the clustering takes into account the spatial distribution of the communications to provide MEC clusters whose loads are well balanced and that keep a large portion of the traffic at the edge, thus offloading the core. Then, we evaluate the MEC partition over a week of communications and show that it largely supports the temporal dynamics. There is almost no server saturation, i.e. traffic offloaded to the core, while the loads remain balanced.

In summary, this paper makes the following contributions:

(1) We design a MEC clustering algorithm (Sec. 2) that consolidates as many communications as possible at the edge.
(2) We use a real-world dataset of spatially and temporally distributed mobile communications (Sec. 3.1).
(3) We evaluate our proposal and show that, despite the spatial-temporal dynamics of the traffic, our algorithm provides well-balanced MEC areas that serve a large part of the communications (Sec. 3.2 and Sec. 3.3).

We discuss related work in Sec. 4 and conclude in Sec. 5.

2 MEC AREA CLUSTERING

In this section, we formulate the MEC area clustering problem that we address and we present our graph-based algorithm.

2.1 Problem formulation

From a network system standpoint we consider a MEC deployment as presented in Fig. 1. All users belong to a *MEC cluster*, a geographic area whose traffic can be handled by a *MEC server*, that is a small-scale datacenter with low to moderate compute and storage resources. All user communications and applications, for instance ephemeral per-communication unikernel-based video analytics applications, are either handled by the local MEC server (e.g. the blue plain line in Fig. 1) or by a highly capacitated *core data center* (e.g. the black dotted line in Fig. 1), which can be farther in terms of latency.

We argue that one of the key design issues in a MEC based system is to efficiently dimension MEC areas (or clusters). Such a MEC geo-partitioning must have the following properties:

(1) MEC servers, as any compute, storage and network node, have a maximum capacity (e.g. in terms of CPU, storage resources or application hosting capabilities).
(2) MEC server loads should be balanced both spatially and temporally to improve user experience.
(3) The traffic between the MEC servers and the core should be minimized, in particular by consolidating applications at the MEC server level, such that the global latency is reduced.

2.2 Graph-based Algorithm

We now introduce our algorithm that, given a maximum MEC server capacity, finds MEC clusters (also referred to as MEC areas) which tend to maximize the traffic handled inside the clusters (i.e. by the MEC servers).

Our algorithm is divided in two phases that are repeated iteratively. Assume that we start with two graphs that have the same set N of nodes (see Fig. 2). These nodes correspond to the discretization of the area where the MEC communications demands are distributed. The first graph $G_a = (N, E_a)$ represents the adjacencies of the nodes on the area. For instance, in a square grid, a node (a grid cell) has up to 8 adjacent nodes (grid cells). The second graph $G = (N, E)$ represents the interactions (i.e. the communications) between the nodes. The weight $w_{i,j} \in \mathbb{R}$ of the edge $e_{i,j} \in E$ represents the amount of interaction (e.g. the number of communications) between node i and node j. Note that a node i can have interaction with itself, leading to a self-loop $e_{i,i}$. So, in this initial partition there are as many MEC clusters as there are nodes.

First, we consider all the edges in E and we select the edge $e_{i,j}$ that has the highest weight such that: i) node i and node j are neighbors in the area (i.e. $\exists\, e^a_{i,j} \in E_a$), ii) the amount of interaction between node i and node j is equal or lower than the maximum cluster capacity C (i.e. $w_{i,i} + w_{i,j} + w_{j,i} + w_{j,j} \leq C$). The selected edge corresponds to the best interaction reduction at this stage.

Secondly, we cluster node i and node j, updating the graphs G_a and G with a new node that represents their clustering. To do so, the neighbors of the new node in G_A and G are the former neighbors of node i and node j. The weights of the links between the new node and its neighbors in G are given by the sum of the weight of the links between the former node i and its neighbors and the former node j and its neighbors. Finally, the weight of the new self-loop corresponds to the sum of the two former self-loops plus the weights between node i and node j. Once this second phase (clustering) is completed, it is then possible to re-apply the first phase (selection) of the algorithm to the resulting graphs and to iterate (see Fig. 2).

By construction, the number of nodes (clusters) decreases at each pass. The passes are iterated until there are no more changes meaning that a local minimum of MEC cluster interaction is attained. The final result of our algorithm corresponds to a partition of the area into MEC clusters/areas whose load (self-loop weights) is inferior but close to the maximum MEC server capacity C. Note

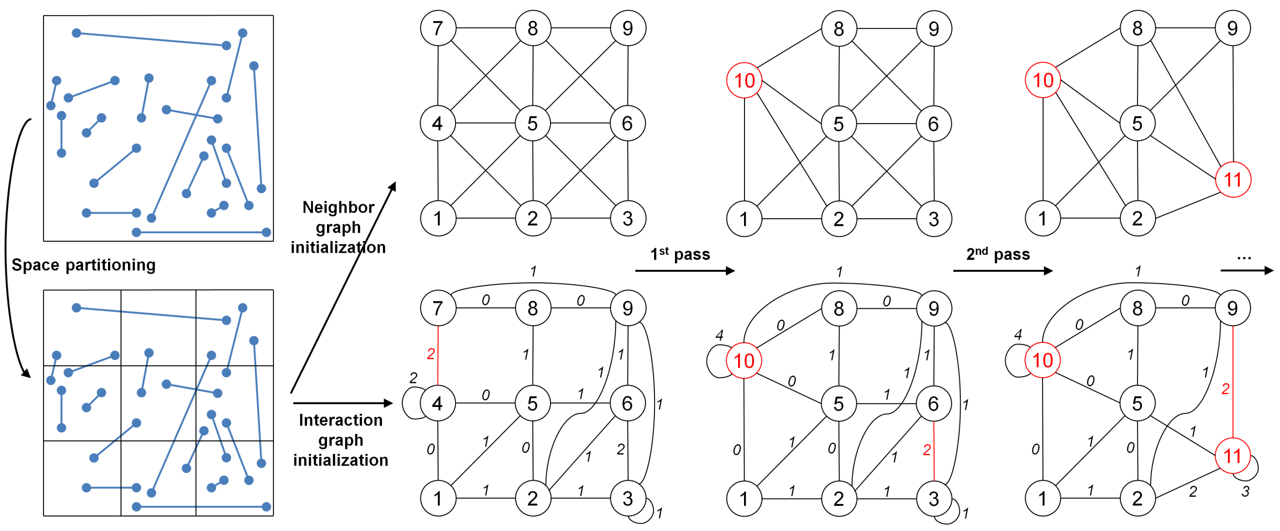

Neighbor graph initialization
Space partitioning
Interaction graph initialization
1st pass
2nd pass
...

Figure 2: Visualization of the steps of our graph-based algorithm. The area where are distributed the MEC communications is discretized into nodes which form MEC clusters. Each pass is made of two phases: one where the pair of neighbor nodes that interact the most, while respecting the maximum cluster capacity, are selected; one where the two selected nodes are clustered to build a new/updated graphs. The passes are repeated iteratively until no pair of neighbor nodes can be merged. The result corresponds to a spatial partition of MEC clusters.

that it can be used in an n-level MEC architecture by applying it at each level.

The algorithm is reminiscent of the community detection algorithms in complex networks (e.g. Louvain algorithm [3]) in the way that it iteratively clusters nodes to increase a modularity (in our case function of the weight of the self-loops). However, it differs on several important points. First, it takes into account spatial properties between the nodes via the graph G_a, which constrains the clustering. Second, only two nodes are clustered at each iteration. Third, it considers a maximum clique/cluster capacity, that corresponds to the weight of the self-loops. Note that this threshold we introduce is adequate to our partitioning problem. However, it could be removed from our algorithm, making it a similar yet different hierarchical clustering algorithm than community detection algorithms. Finally, we purposely present a simple description of the algorithm, but heuristics may be introduced to improve its performance or introduce variants (e.g. order the edges in the first phase, consider more complex interactions such as communication groups, perform a local search on the final result, consider different maximum cluster capacities etc.).

The time complexity of the algorithm described as above is $O(E.N)$, where E is the number of edges and N the number of vertices. Indeed, the first phase basically consists in going through all the E edges of G and finding the non self-loop edge with the maximum weight. The second phase consists in adding and removing (i.e. clustering) nodes in graphs (G_a and G). With adjacency lists, you simply need to iterate over the edge list of the nodes to be clustered and update all those nodes. The algorithm stops when no pair of nodes can be merged anymore, which means maximum $N - 1$ passes are done. The number of edges E depends on the nature of the graph. For example, it averages $k.(N - 1)/2$ in an Erdös-Rényi graph [5], where $k > 1$ is the mean vertex degree. The complexity of our algorithm would thus be $O(N^2)$. Note that it is a pessimistic upper bound since at each pass both the number of edges and vertices, and thus the number of operations, decrease.

3 EVALUATION AND ANALYSIS

In this section we evaluate our MEC clustering algorithm with a real dataset of mobile communications. We first analyze the results considering different day types and different periods of the day. Then, we evaluate it through time.

3.1 Dataset

To evaluate our algorithm and show the benefits of the MEC approach compared to a classic centralized architecture, we use the dataset published as Open Data by Telecom Italia in 2014 [1].

This dataset contains geo-referenced Call Detail Records (CDRs) over the city of Milan from November 1st, 2013 to January 1st, 2014 [2]. During this period, every time a mobile user engaged a telecommunication interaction with another mobile user in the region, a CDR was created containing the date time of the call and the geographical locations of the mobile users (derived from the location of the base stations they used). The dataset was created combining all this anonymous information, with a temporal aggregation of time slots of ten minutes and a spatial aggregation of square grid calls of 235x235 meters (a grid of 100x100 cells to cover the city of Milan). The number of records in the dataset $S_i'(t)$ follows the rule: $S_i'(t) = k.S_i(t)$ where k is a constant defined by Telecom Italia. It aims at hiding the true number of calls. Since Telecom Italia only possesses the data of its own customers, the computed interactions are only between them. This means that (at

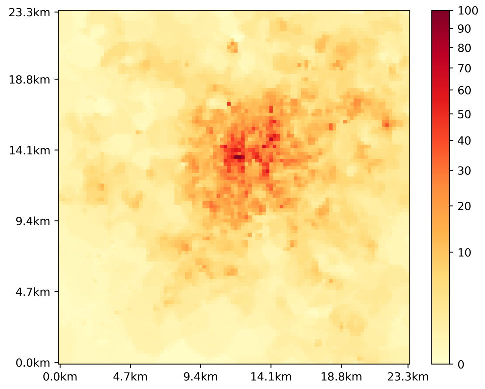


Figure 3: Normalized mobile communication intensity in the city of Milan (5pm-6pm, 11/04/2013). The communications are concentrated in the city center.

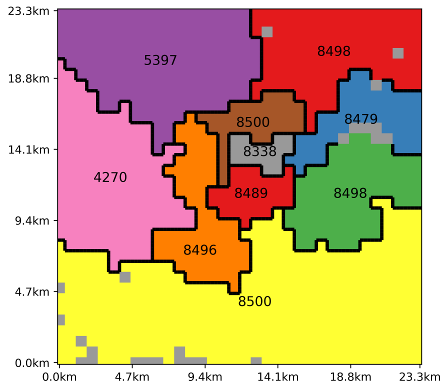


Figure 4: Clustering result for a maximum cluster capacity of 5% of the total communications, i.e. 8,500 communications (5pm-6pm, 11/04/2013). The numbers in the clusters correspond to their load.

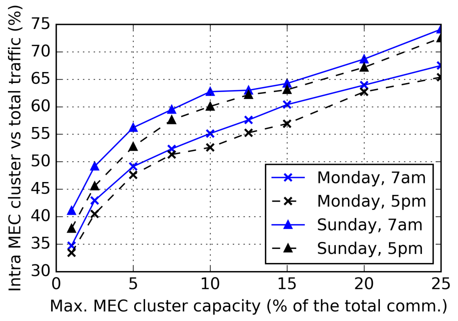


Figure 5: Proportion of intra MEC cluster traffic (Monday 11/04/2013 and Sunday 11/10/2013).

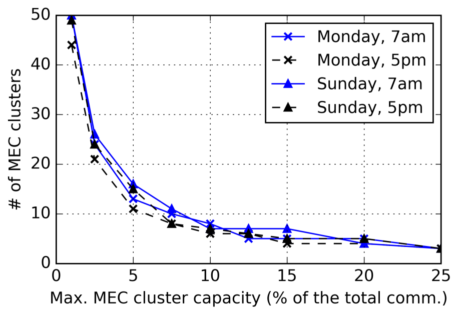


Figure 6: Number of MEC clusters (Monday 11/04/2013 and Sunday 11/10/2013).

most) 34% of population's data is collected, due to Telecom Italia's market share. Around 1,3 million people live in the city of Milan.

To evaluate our algorithm, we generate from this anonymized dataset batches of communications aggregated by hour and spatially distributed over the city of Milan. The volume of generated communications every hour is proportional to what was measured by Telecom Italia (see Sec. 3.3).

Fig. 3 shows the normalized mobile communication intensity distributed in the city of Milan between 5pm and 6pm during a working day (Monday). We can clearly distinguish the city center, which gathers most of the mobile calls.

3.2 MEC resources partitioning

Geo-clustering. We first illustrate the result of the geo-partitioning algorithm. Fig. 4 presents the results of the geo-clustering algorithm on Monday 11/04/2013 between 5pm and 6pm with a maximum cluster capacity of 5% of the total communications, that represents maximum 8,500 communications. The algorithm started with 1,089 clusters (33 x 33) and took, without any code optimization, 18.73 seconds to provide this result. The grey atomic squares correspond to grid cells with no or very low traffic. However, they could not be ultimately merged to a neighbor cluster because it would have increased their load above the maximum threshold. We thus consider that if their intra-cluster traffic is lower than 0.01% of the maximum capacity, they do not form a cluster and their communications are directly served in the core. We can see that the area of the clusters that cover the city center is smaller than the ones that serve low-density regions, the density of communications being higher there (see Fig. 3). Moreover, as seen in Fig. 7, most of the loads are close to the maximum server capacity. The MEC areas are thus well balanced.

Core offloading. We then turn to the benefits of the MEC approach with respect to core offloading, that is we look at which portion of the communications is directly served at the edge. As the communication demands are spatially distributed in time and space according to human activity (residential, business, entertainment, transport etc.), we consider a working day (Monday 11/04/2013) and a weekend day (Sunday 11/10/2103) at two periods: beginning of the activities (7am to 8am) and communication peak (5pm to 6pm). Fig. 5 shows the the amount of communications directly handled at the MEC servers in function of the maximum MEC server capacity (expressed in percentage of the total communications to serve).

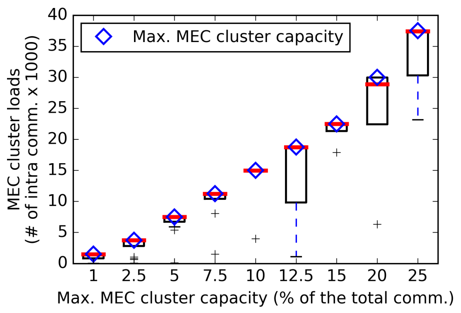


Figure 7: MEC cluster loads (5pm-6pm, 11/04/2013).

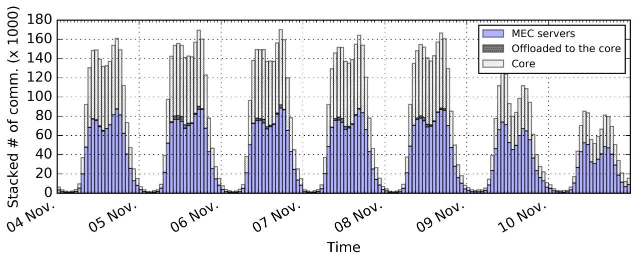
170,000 communications, derived from the dataset, were considered at each point. We can observe that an important part of the communications does not have to go up to the core. For example, with a maximum cluster capacity of 7.5%, which represents 12,750 communications, between 50% and 60% of the traffic is directly absorbed by the MEC servers. The core offloading remains important even for very small cluster capacities. We can also observe that the gain varies according to the day and the time of the day. The lower gains are at the peak hour of the working day, while the upper gains are at the beginning of the weekend day. These observations can be explained by i) the spatial locality of the mobile communications and ii) the difference of human activities (mainly business and transportation on Monday at 5pm and residential on Sunday at 7am). Fig. 6 presents the corresponding number of MEC servers. There is no major difference. Naturally, as the maximum cluster capacity diminishes the number of clusters increases to serve traffic at the edge. Note that with a spatial uniform distribution of the communications, we would have for instance 20 servers, instead of 11-15, for a maximum capacity of 5%.

Server load balancing. Fig. 7 shows the loads of the clusters at the peak hour on Monday. We can observe that the partition, and hence the load, is well balanced. Indeed, most of the clusters have a load close to the maximum cluster capacity. Moreover, in all cases, the median values almost match the maximum values. We had the same observations for the other periods we evaluated.

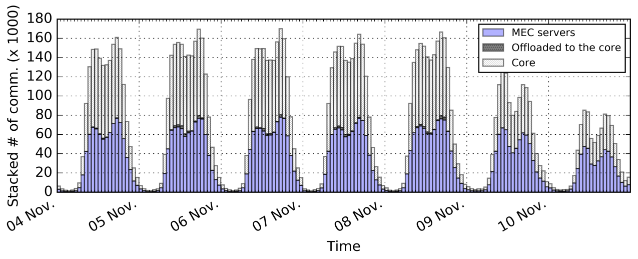
3.3 Through time

Core offloading. We finally evaluate the performance of our algorithm through time. To this aim, we first use the mobile data on a full week. We consider that at the peak hour of this period (Thursday, 5pm, 11/08/2013), there are 170,000 communications per hour. It represents more or less the volume of communications in the city of Milan for the market share of Telecom Italia in 2013. We retrieved from the dataset the proportion of communications hour by hour and their spatial distribution. We then considered three partitions obtained on Monday 11/04/2013 at different hours (7am-8am and 5pm-6pm) of the day and with different maximum cluster capacity (5% and 10% of the total communications at this period of the day).

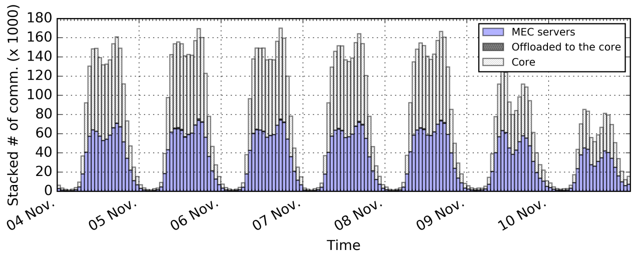
In Fig. 8a, we can observe that around 53% of the communications are directly handled by the MEC servers during the working days. This share increases up to 61% during the week-end. Obviously, if



(a) Partition done at 5pm-6pm on Monday 11/04/2013 with a maximum cluster capacity of 10% of the total communications.



(b) Partition done at 5pm-6pm on Monday 11/04/2013 with a maximum cluster capacity of 5% of the total communications.



(c) Partition done at 7am-8am on Monday 11/04/2013 with a maximum cluster capacity of 5% of the total communications.

Figure 8: MEC servers and core traffic distributions over a week for different partitions.

we consider a maximum cluster capacity of 5% (Fig. 8b), the global load distribution through time remains the same, but the traffic share of the MEC servers drops to approximately 45%. In both cases, we can notice that the traffic offloaded to the core, that corresponds to the local traffic that could not be handled by MEC servers because they were saturated, is very small (maximum 3.1% on 11/05/2013 at 12am). Finally, we can notice that, if we consider a partition done in the morning (Fig. 8c) instead of the peak hour (Fig. 8b), the share of the MEC servers slightly decreases, while they remain almost unsaturated.

Server load balancing. At last, Fig. 9 presents the distribution of the MEC server loads. The partition corresponds to the one shown on Fig. 4. It was done with the communications that occurred between 5pm and 6pm. We can distinguish human activities, that is low activity until 8am and after 9pm and two peaks around 10 am and 5pm. At each hour, the load is homogeneously distributed on the servers.

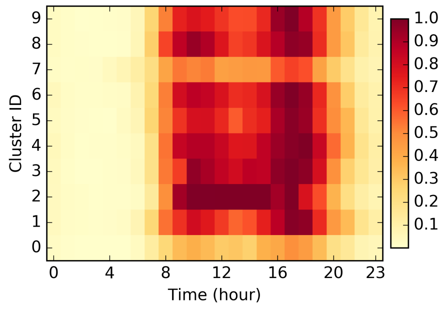


Figure 9: Normalized MEC cluster loads over a day (11/04/2013) with a partition done at 5pm and a maximum cluster capacity of 5% of the total communications.

4 RELATED WORK

In the past few years, in parallel notably to the ETSI initiative [6] and to the OpenFog Consortium [7], MEC has emerged as a new research area. We present in this section related work.

Partitioning and MEC server placement: The MEC server placement problem was illustrated by Qazi et al. [12] who showed that the number and the locations of MEC servers have a direct impact on the QoE (imbalance loads and high latencies) and on the operational cost. However, they did not address the problem. They proposed an NFV-based orchestration for MEC. Note that the server placement problem is significantly different from the conventional base station site selection problem since, although both problems are constrained by the deployment budget, placing edge sites is coupled with the computational resource provisioning. Ceselli et al. [4] have proposed an integer linear programming formulation of the joint problem of base stations allocation to MEC servers and routing to reduce infrastructure cost. Our proposal mainly differs on three important aspects. First, they assume the locations of the LTE 4G base stations are known. Their analytical formulation does not scale properly. Most of all, the clusters they obtained are not geo-consistent, meaning that the base stations associated to a MEC server can be completely scattered in space.

System approaches: While NFV has gained momentum, recent proposals have focused on shortening network functions instantiation and reducing their system footprint with approaches based on unikernels [16]. In particular, it has been shown that an inexpensive commodity server is able to concurrently run up to 10,000 specialized virtual machines, instantiate a VM in as little as 10 milliseconds, and migrate it in under 100 milliseconds [9]. This technology is very promising in an MEC context where an application could be instantiated on the fly at MEC servers for a user or a group of users and shutdown once the communication is ended. Progress in this direction complements our deployment work as it would make it easier to instantiate locally applications at MEC servers.

Application/task offloading: Application offloading, both from the device to the edge and from the core to the edge, has been extensively studied [10]. It notably includes task decomposition, assignment, and migration, server scheduling, and selection, content caching and pre-fetching. Some of the proposals are similar to those addressed in Mobile Cloud Computing (MCC), which addresses distributed clouds [13, 15].

5 CONCLUSION AND PERSPECTIVES

MEC is a key technology to support the ever-increasing growth of communication capability demands and realize the IoT and 5G visions. As operators are transforming their network architectures and are looking for deploying computation resources close to the user to improve QoE, it is necessary to adequately dimension MEC systems. In this paper, we presented a graph-based algorithm that enables finding a partition of MEC areas that consolidates traffic at the edge, in MEC servers. We evaluated it using a real world dataset from a mobile operator. The evaluation results, beyond quantifying the benefits of the MEC approach, show that the core can be largely offloaded. They also show that the algorithm provides MEC areas that are well balanced in terms of load. Finally, we ran simulations over one week of communications and observed that there is almost no saturation of the MEC servers, while the traffic on the core is largely reduced. In future work, we expect to explore several aspects such as group communications, energy saving, and latency. We also aim at combining our approach with online application offloading and migration.

ACKNOWLEDGMENT - This work was partially supported by the H2020 ANASTACIA project (No. 731558).

REFERENCES

[1] 2014. Open Big Data. https://dandelion.eu/datamine/open-big-data/. (2014). Accessed: 2017-03-23.
[2] G. Barlacchi, M De Nadai, R Larcher, and et al. 2015. A multi-source dataset of urban life in the city of Milan and the Province of Trentino. *Scientific Data* 2, 150055 (2015).
[3] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E.L.J.S. Mech. 2008. Fast unfolding of communities in large networks. *J. Stat. Mech* (2008).
[4] A. Ceselli, M. Premoli, and S. Secci. 2017. Mobile Edge Cloud Network Design Optimization. *IEEE/ACM Transactions on Networking* 99 (2017), 1–14.
[5] Paul Erdos and Alfréd Rényi. 1961. On the evolution of random graphs. *Bull. Inst. Internat. Statist* 38, 4 (1961), 343–347.
[6] European Telecommunications Standards Institute (ETSI). 2015. *Mobile-Edge Computing (MEC); Service Scenarios (GS MEC-IEG 004).*
[7] OpenFog Consortium Architecture Working Group. 2016. *OpenFog Architecture Overview (OPFWP001.0216).*
[8] Y. C. Hu, M. Patel, D. Sabella, and et al. 2015. *Mobile Edge Computing A key technology towards 5G (ETSI White Paper No. 11).* European Telecommunications Standards Institute (ETSI).
[9] F. Manco, J. Martins, K. Yasukata, and et al. 2015. The Case for the Superfluid Cloud. In *Proceedings of USENIX HotCloud Workshop.*
[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. Ben Letaief. 2017. Mobile Edge Computing: Survey and Research Outlook. *CoRR* abs/1701.01090 (2017). http://arxiv.org/abs/1701.01090
[11] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica. 2015. Large-scale Mobile Traffic Analysis: a Survey. *IEEE Communications Surveys and Tutorials* (2015).
[12] Z. Ayyub Qazi, P. Krishna, V. Sekar, and et al. 2016. KLEIN: A Minimally Disruptive Design for an Elastic Cellular Core. In *Proceedings of ACM Symposium on SDN Research (SOSR).*
[13] H. Tan, Z. Han, X.Y. Li, and F.C.M. Lau. 2017. Online Job Dispatching and Scheduling in Edge-Clouds. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM).* 1–9.
[14] N. Tastevin and M. Bouet. 2016. Characterizing and modeling the distance of mobile calls: A metropolitan case study. In *Proceedings of IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC).*
[15] L. Tong, Y. Li, and W. Gao. 2016. A hierarchical edge cloud architecture for mobile computing. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM).* 1–9.
[16] P.L. Ventre, C. Pisa, S. Salsano, and et al. 2016. Performance Evaluation and Tuning of Virtual Infrastructure Managers for (Micro) Virtual Network Functions. In *Proceedings of IEEE NFV-SDN Conference.*
[17] H. Wang, F. Xu, Y. Li, P. Zhang, and D. Jin. 2015. Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment. In *Proceedings of ACM Internet Measurement Conference (IMC).*