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
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 $(i, j) \in E$ 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. $\exists e_{i,j} \in E_a$), ii) the amount of interaction between node $i$ and node $j$ is equal or lower than the maximum cluster capacity $C$ (i.e. $w_{i,j} + w_{j,i} + w_{i,j} \leq C$). 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
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 Erdős-Rényi graph [3], 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
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).
170,000 communications, derived from the dataset, were consi-
idered 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 ab-
sorbed 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 communi-
cations 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 al-
gorithm 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 
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.
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