
Article:
Bozorgchenani, A. orcid.org/0000-0003-1360-6952, Tarchi, D. orcid.org/0000-0001-7338-1957 and Corazza, G.E. (2019) Centralized and Distributed Architectures for Energy and Delay Efficient Fog Network-Based Edge Computing Services. IEEE Transactions on Green Communications and Networking, 3 (1). pp. 250-263. ISSN 2473-2400

https://doi.org/10.1109/tgcn.2018.2885443

Centralized and Distributed Architectures for Energy and Delay Efficient Fog Network based Edge Computing Services

Arash Bozorgchenani, *Student Member, IEEE,* Daniele Tarchi, *Senior Member, IEEE,* and Giovanni Emanuele Corazza, *Senior Member, IEEE*

***Abstract*—Edge computing techniques allow to exploit the devices at the network borders for computing efforts in order to reduce the centralized cloud requests. A fog network is a feasible solution for implementing edge computing services. Within this scenario, the deployed Fog Nodes (FNs) are able to offload different portions of a single task to the available nodes to be processed at the network edge. However, to partially offload, FNs consume an extra amount of energy for transmission and reception of the tasks while saving energy by not processing the whole task on their own. Moreover, offloading requires an extra transmission and reception time to the task processing time. In this work, the focus is on a partial offloading approach where the trade off between FN energy consumption and task processing delay is considered when estimating the portion to be offloaded to the available devices at the edge of the network by comparing a centralized and a distributed architecture. Simulation results demonstrate the effectiveness of the proposed estimation solutions in terms of FN energy consumption, average task delay and network lifetime.**

***Index Terms*—Edge Computing, Fog Networking, Partial Offloading, Energy Consumption, Clustering**

I. Introduction

EDGE computing is a recently introduced framework aiming at bringing the computing capability of the cloud at the edge of the network with the goal of minimizing the time required for responding to a task request and reducing the traffic at the fronthaul [1]. Among different architectural proposals Mobile Edge Computing (MEC) [2] seems the most promising, having on one hand the advantages of edge computing, while considering the presence of mobile nodes as an additional flexibility, by extending the Mobile Cloud Computing (MCC) approach [3], [4]. The increasing interest in MEC is also evident by looking at the standardization effort in ETSI, where some possible use cases have been identified: active device location tracking, augmented reality content delivery, video analytic, radio access network aware content optimization, distributed content and DNS caching and application-aware performance optimization [5]. Recently, Mobile Edge Computing has been also renamed as Multiaccess Edge Computing, where the acronym still remains MEC, while giving much more emphasis on the communication heterogeneity that is at the basis of the architectural proposal [6].

While MEC seems to place much more focus on the necessity of having edge computing capability limited to process the traffic requests at the edge, a similar approach named Fog Computing was recently introduced. Fog Computing seems to be an extension of MEC into two different directions: on one side by trying to exploit as many devices as possible in a unified approach by resorting to the Internet of Things (IoT) principles [7], and on the other side by trying to go beyond the Edge vs Centralized approaches in cloud computing by introducing a much more fluid view to the architectural definition, where different devices can be seen as a continuum from the edge to the centralized cloud of several hierarchical layers with different capabilities [8].

Within the Fog computing scenario, and the related Fog networking scenario, that aims at considering the communication and networking challenges introduced by the Fog architecture, the aim of this paper is that of considering different edge computing approaches.

Within this scenario, we envisage to consider a two-layer architecture composed of Fog Nodes (FNs), battery operated nodes with lower computing capabilities, and Fog Access Points (F-APs), fixed nodes usually connected to the electrical power network with higher computational capabilities. These two types of nodes are logically organized into two interconnecting layers; to this aim two communication paradigms are usually considered: Device to Device (D2D) communication among FNs, and infrastructure communications between FNs and F-APs [9].

Among different applications that can be envisaged in an edge computing infrastructure, we focus here on computation offloading, characterized by the possibility of offloading the tasks computation to the nearby devices [10], [11]. Due to the limited FNs capabilities, the edge computing requests cannot be completely fulfilled with a requested target (e.g., delay or energy consumption). To overcome this problem, a joint exploitation of both FNs and F-APs is here considered. In D2D communications, FNs are able to share their resources with the neighboring FNs, while in infrastructure communications, requests are sent from FNs to F-APs for being computed. This allows to exploit the advantages of both when offloading a task computation. Moreover, due to the presence of multiple

The authors are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy, email: {arash.bozorgchenani2, daniele.tarchi, giovanni.corazza}@unibo.it.

This work has been supported by the project "GAUChO - A Green Adaptive Fog Computing and Networking Architecture" funded by the MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2015 - grant 2015YPXH4W_004.

Manuscript received June 01, 2018; revised October 05, 2018; accepted November 29, 2018.

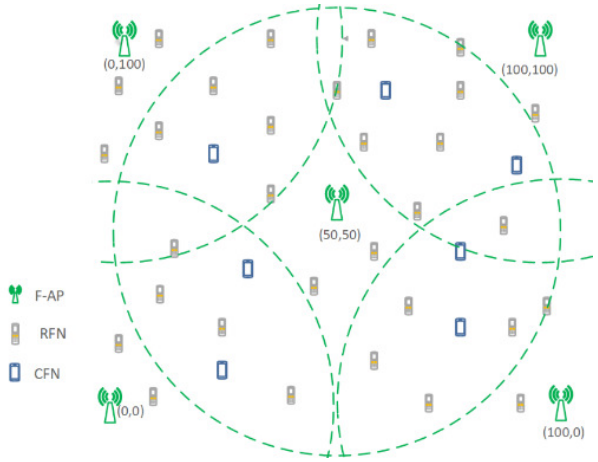


Fig. 1. The considered two-layer Fog Network architecture

nodes nearby, a task can be further divided into some portions, each one allocated to a different node. This technique is called partial offloading [12].

The considered Fog architectural model is shown in Fig. 1, where it is possible to see the FNs and the F-APs that interact in forming the Fog computing infrastructure. The FNs in this figure are divided into two types: the Requesting FNs (RFNs), the devices offloading the computational tasks, and the Computing FNs (CFNs), the devices accepting tasks to be computed from the RFNs.

The goal of this paper is the optimal distribution of the computational effort among the nodes by jointly minimizing the overall task processing delay and the energy consumption while maximizing the network lifetime [13]. To this aim, the optimization has been carried out by resorting to two different approaches, a distributed and a centralized. While in the distributed each RFN is selecting autonomously the nodes to be used for computing, in the centralized approach we foresee to optimize the system by centrally driving the offloading requests of the RFNs.

Differently from [14], where energy consumption and task processing delay have been considered for the optimization of computation offloading, a new optimization approach is here proposed. This work mainly features the following characteristics with respect to [14]:

(a) Architecture: The F-APs in [14] are considered as an alternative option in case of no availability of FNs, while in this work a more realistic architecture is considered, in which both F-APs and FNs are available for computation offloading. More importantly a centralized architecture is additionally proposed in this work. To this aim, one of the major goals of this work is that of comparing the centralized and the distributed architectures.

(b) Partial offloading estimation: By the availability of all types of nodes for computation offloading in the proposed architectures, a new offloading estimation is proposed. Differently from [14], which only considers the data rate for the estimation of the portion to be offloaded to the available nodes, here, we have considered also the computational power of the available nodes.

TABLE I
ACRONYM LIST

Acronym	Term
MEC	Mobile Edge Computing
MCC	Mobile Cloud Computing
IoT	Internet of Things
FN	Fog Node
F-AP	Fog-Access Point
D2D	Device to Device
RFN	Requesting Fog Nodes
CFN	Computing Fog Node
FLOPS	Floating-Point Operation Per Second
LPFN	Low Power Fog Node
HPFN	High Power Fog Node
FCH	Fog Cluster Head
FCM	Fog Cluster Member

The paper is organized in the following way. In Section II, a literature review of the most influential papers in the area is given. Then, in Section III, the system model and the problem formulation are introduced. In Section IV, the proposed centralized and distributed solutions are described, while in Section V the numerical results are given. Finally, in Section VI, the conclusions are drawn. A list of acronyms used in the following is also shown in Tab. I.

II. Related Works

Although fog computing and networking has been recently introduced, the research community is very active in this field. The task offloading problem has been formulated in [15] as a joint radio and computational resources problem. From the architectural point of view, the network in [16] is broken down into several layers in a way that some cloudlets for mobile cloud computing are considered. To enhance network capacity and offloading probability, in this work a D2D-based heterogeneous MCC network is proposed. Clustering in edge networking has also been proposed in some works. In [17] clustering was performed among the access points considering channel and caching status. A clustering algorithm was also proposed in [18] for the radio access points dealing with joint computation and communication resource allocation inside the cluster.

Some works have considered the delay as the optimization goal in edge computing. To target a fog server from the user point of view, [19] considers both communication and computing delays. However, an assumption made in [19] was that all the users can access all the fog servers. In [20], the authors have studied the multi-user computation partitioning problem between mobile devices and cloud. They have proposed an offline heuristic and considered a large-scale mobile cloud application with the aim of minimizing the average completion time.

On the other hand, some papers have considered the energy consumption as the goal of their optimization. The authors in [21] studied the impact of offloading on reducing the energy consumption by focusing on an intensive communication scenario. An Energy-Efficient Computation Offloading (EECO) algorithm is proposed in [11] based on three main phases:

classifying the nodes considering their energy and cost feature, prioritizing them by giving a higher offloading priority to the nodes which cannot meet the processing latency constraint, and the radio resource allocation of the nodes considering the priority. The proposed EECO algorithm allows to decrease the energy consumption by up to 15% in comparison with computation without offloading. The authors in [22] have proposed a matching game approach to solve the problem of resource allocation of the cached content in fog computing environments. The aim of the work is minimizing the energy consumption for the content access for which they have considered the energy consumption of the requesting node, embedded computing on the node, baseband processing at the edge, radio interface, transmission through core network and internet, and processing in the cloud. The authors in [23] have introduced an energy efficient fog computing framework in homogeneous fog networks and proposed a low complexity maximal energy efficient task scheduling algorithm to derive the optimal scheduling decision for a task node. In [24] a node discovery approach in IoT-fog environment was proposed. The authors mainly worked on the impact of the dynamicity of the advertiser nodes on device discovery success, which was proved to be 100%, and also sustainability of the battery-powered IoT nodes. In [25] the challenges of energy efficiently multimedia sensing as a service at the cloud edges IoT was investigated. The authors proposed a resource allocation approach to achieve optimal multimedia transmission quality in addition to guaranteeing wireless communication energy efficiency.

However, computation offloading affects both FN energy consumption and task delay. Few works have considered both metrics. The authors in [26] proposed energy-efficient offloading policies for transcoding tasks in a mobile cloud system. With the objective of minimizing the energy consumption while meeting the latency constraint, they introduced an online offloading algorithm which decides whether the task should be offloaded to the cloud or executed locally. Task processing in [27] was based on a decision of either local processing or total offloading. The authors aimed at minimizing the local execution energy consumption for applications with strict deadline constraints. Authors in [28] studied the problem of network energy minimization while satisfying applications' delay requirement in cloud radio access networks. A joint optimization of beamforming design and power allocation with a decision making strategy is considered. A heuristic offloading decision algorithm was proposed in [29] with the aim of maximizing system utility which considers task completion time and the FN energy consumption. However, in [29] only a single MEC server was considered. Energy consumption and latency have also been targeted in [30] for an offloading approach. In this work, the authors targeted energy consumption and response time for the offloading scenario to the centralized cloud.

In our work, we have considered both centralized and distributed architectures for the partial offloading problem. Moreover, we have introduced two approaches working on both FN and F-AP layers considering the FN energy consumption and the task processing delay for a sub-optimal solution to the partial offloading problem. The proposed approaches estimate the amount to be offloaded to the available nodes in both layers such that average task delay, FN energy consumption and network lifetime are optimized.

III. System Model

In this work a two-layer architecture for fog computing is considered. On one hand $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_N\}$ represents the set of FNs in the first layer. The FNs, characterized by limited computational capabilities, are battery powered and are the sources of the computational requests in the system. FNs are able to communicate among themselves for enabling the direct offloading through direct link technologies (e.g., D2D or WiFi-Direct). On the other hand, the second layer is composed by F-APs, whose set is indicated as $\mathcal{A} = \{a_1, \ldots, a_m, \ldots, a_M\}$, characterized by a higher computation capability. The F-APs are plugged to the electrical network, resulting in a virtual unlimited energy, and could also be used for the connection of the FNs with the centralized cloud. F-APs can be exploited by the FNs for computation offloading. In our system we focus on both layers by limiting the offloading requests to these layers and avoiding any offloading requests to outer clouds. The F-APs act as mobile edge computing resources, providing the ability of running multiple computations at the same time [11]. In the following, the FNs are considered to be steady and able to offload their tasks to the neighboring FNs or to the F-APs.

The goal of this paper is to optimally estimate the task portion to be offloaded by each FN having some tasks to be computed in order to jointly minimize the energy consumption and the task processing delay. For pursuing such optimization we resort to a partial offloading technique that allows to select the amount of data to be offloaded to each of the possible candidates among the available FNs and F-APs, while the remaining can be processed locally. For the sake of readability, in Tab. II the parameters definitions to be used in the following equations are listed.

Each FN can be considered in one of four possible states during its life: transmitting, receiving, computing or idle, i.e., $\mathcal{S} = \{tx, rx, com, id\}$. The transmitting and receiving states refer to the interaction with other FNs or F-APs and the computing state refers to the computation performed in the FN itself (either for a local task or for an offloaded task); the idle state is considered the remaining time. In the rest of the paper, nodes refer to both FNs or the F-APs, unless otherwise stated. The overall energy consumed by the generic ith FN can be defined as:

$$E^i_{FN} = E^i_{tx} + E^i_{rx} + E^i_{com} + E^i_{id} \tag{1}$$

where E^i_{tx}, E^i_{rx} and E^i_{com} are, respectively, the energy consumed during transmission, reception and computation states and E^i_{id} is the energy the ith FN spends during its idle state. The energy spent by the ith FN in a certain state s can be defined as:

$$E^i_s = P^i_s T^i_s, \qquad s \in \mathcal{S} \tag{2}$$

where P^i_s represents the power and T^i_s the time spent by the ith FN in the state s.

TABLE II
SYSTEM MODEL PARAMETERS DEFINITION

Parameter	Definition
E_s^i	The energy consumption of the ith FN in state s
$E_r^i(t)$	The remaining energy of the ith FN at time instant t
P_s^i	The consumed power of the ith FN in state s
B_{ij}	The bandwidth of the link between two nodes i and j
h_{ij}	The channel coefficient between two nodes i and j
P_{N_j}	The noise power at receiver side
T_s^l	The time spent by the ith FN in state s
$T_{w_j}^l$	The waiting time of the lth task in the queue of the jth node
r_{ij}	Data rate of the link between the ith and jth node
L_{s_l}	Size of the lth task
L_{r_l}	Size of the lth task result
O_l	Number of operations to process the lth task
η_{comp_j}	The computational power of the jth node
$\alpha_{loc,i}^l$	Local portion of the lth task of the ith node
$\alpha_{off,i}^l$	Offloading portion of the lth task of the ith node
$\alpha_{off,ij}^l$	Offloading portion of the lth task of the ith node to the jth node
E_{CFN}^j	Energy consumption of the jth CFN
E_{RFN}^i	Energy consumption of the ith RFN
$T_{loc,i}^l$	Local computation time for the lth task of the ith node
$T_{off,ij}^l$	Offloading time of the lth task of the ith node to the jth node
D_i^l	Total delay of the lth task of the ith node

We suppose that the initial energy of the ith node is $E_r^i(0)$. All the FNs consume a certain amount of energy when they transmit, receive or compute tasks or even when they are idle. Therefore, by a certain time t, the ith FN has consumed $E_c^i(t)$ Joule of energy. Thus, the remaining energy of the ith FN at certain time instant t can be calculated as:

$$E_r^i(t) = E_r^i(0) - E_c^i(t) \tag{3}$$

where,

$$E_c^i(t) = \int_0^t E_{FN}^i(\tau)d\tau \tag{4}$$

In general, the time spent by the jth node, whether it is an FN or an F-AP, for processing the lth task can be defined as:

$$T_{comp_j}^l = \frac{O_l}{\eta_{comp_j}} \tag{5}$$

where O_l represents the number of processing operations related to the lth task and η_{comp_j} is the Floating-point Operation Per Second (FLOPS) depending on the CPU of the jth processing node, which can be an FN or an F-AP.

In case of offloading, the lth task should be transmitted; hence, the transmission time from the ith FN to the jth node for the lth task can be written as:

$$T_{tx,ij}^l = \frac{L_{s_l}}{r_{ij}} \tag{6}$$

where L_{s_l} is the size of the lth task offloaded by the ith FN and r_{ij} is the data rate of the link between the ith FN and the jth node. Following this, the result of the processed task should be sent back from the jth node to the ith FN, leading to a reception time defined as:

$$T_{rx,ij}^l = \frac{L_{r_l}}{r_{ij}} \tag{7}$$

where L_{r_l} is the size of the result of the requested task sent back to the requesting FN, by supposing a symmetric channel in terms of data rate between the ith FN and the jth node. By considering the Shannon capacity formula, the data rate between the ith FN and the jth node can be written as:

$$r_{ij} = B_{ij}\log_2\left(1 + \frac{|h_{ij}|^2 P_{tx}^i}{P_{N_j}}\right) \tag{8}$$

where B_{ij} is the bandwidth of the link, P_{tx}^i represents the transmission power of the ith FN, h_{ij} is the channel coefficient between the ith FN and the jth node and P_{N_j} is the noise power at the receiver side, defined as $P_{N_j} = N_T B_{ij}$.

In our system we are supposing that the F-APs are able to process the tasks received from other FNs, while the FNs can process both tasks generated by themselves or received from other nearby FNs. To this aim, in reference to the role they are having, in the following we will refer to the RFNs as those FNs asking other nodes to process a task, and CFNs as those FNs computing a task on behalf of other FNs.

Each CFN and F-AP in our work is supposed to have a queue holding the tasks of the RFNs to be processed. The waiting time of the lth task at the jth node can be defined as:

$$T_{w_j}^l(p) = \sum_{\pi=1}^{p-1} T_{comp_j}^{\pi} \tag{9}$$

where p is the number of tasks already in the queue of the jth node at a given time instant.

The concept behind partial offloading is to delegate only a portion of the computation load to another node. This allows to have a higher flexibility and optimize the energy consumption and the time spent for processing the tasks. We define $\alpha_{loc,i}^l$ as the portion of the lth task that can be processed locally by the ith RFN generating that task, and $\alpha_{off,i}^l$ as the amount that can be offloaded by the ith RFN, where $\alpha_{off,i}^l = 1 - \alpha_{loc,i}^l$. We are considering that the offloaded portion can be further split among the available nodes. In this case, the offloaded portion can be written as:

$$\alpha_{off,i}^l = \sum_{j \in \mathcal{N}(i)} \alpha_{off,ij}^l \tag{10}$$

where $\mathcal{N}(i)$ is the set of the neighbor nodes available for accepting the offloaded portion, and $\alpha_{off,ij}^l$ is the portion offloaded by the ith RFN to the jth node.

The time required for offloading the portion of a task from the ith RFN to the jth node can be written as the sum of the time for offloading the portion of the task, the time the task should wait in the jth node queue, the time for computing that task and the time needed for having the result back:

$$T_{off,ij}^l(\alpha_{off,ij}^l) = \alpha_{off,ij}^l T_{tx,ij}^l + T_{w_j}^l + \alpha_{off,ij}^l T_{comp_j}^l + \alpha_{off,ij}^l T_{rx,ij}^l \tag{11}$$

while the time for local computation, can be defined as the time needed for computing the remaining portion of the task:

$$T_{loc,i}^l(\alpha_{off,i}^l) = \alpha_{loc,i}^l T_{comp_i}^l = (1 - \alpha_{off,i}^l) T_{comp_i}^l \tag{12}$$

By assuming that the local and the offloaded portions can be performed in parallel, the total delay for processing a task can be rewritten as the maximum of all the offloading times and the local time, i.e.,

$$D_i^l(\alpha_{off,i}^l) = \max_{\forall j \in \mathcal{N}(i)} \left\{ T_{off,ij}^l(\alpha_{off,ij}^l), T_{loc,i}^l(\alpha_{off,i}^l) \right\} \quad (13)$$

On the other hand, the energy consumption of the jth CFN, in case of partial offloading, could be rewritten as:

$$E_{CFN}^j = \alpha_{off,ij}^l (E_{rx}^{ji} + E_{com}^j + E_{tx}^{ji}) + E_{id}^j \quad (14)$$

where E_{rx}^{ji} and E_{tx}^{ji} are the energy amounts spent by the jth node for receiving from and transmitting to the ith node, respectively; it corresponds to the sum of the energy of reception, computation and transmission of the portion that is offloaded plus the idle energy of the jth CFN. On the RFN side the energy consumption can be rewritten as:

$$E_{RFN}^i = \sum_{j \in \mathcal{N}(i)} \left(\alpha_{off,ij}^l \left(E_{tx}^{ij} + E_{rx}^{ij} \right) \right) + \alpha_{loc,i}^l E_{com}^i + E_{id}^i \quad (15)$$

corresponding to sum of the energy required for offloading the portion to the nearby nodes, performing the computation of the rest of the task locally, and the energy of idle state. We introduce now the following Boolean variable:

$$U_{FN}^i = \begin{cases} 1 & \text{if the } i\text{th FN is a CFN} \\ 0 & \text{if the } i\text{th FN is an RFN} \end{cases} \quad (16)$$

representing the two possible conditions in which an FN can be.

In this work, the goal is to minimize the average FN energy consumption in the network and the overall delay. This leads to a formulation of the partial offloading problem as:

$$\min_{\boldsymbol{\alpha_{off}}} \left\{ \sum_{i=1}^{N} \left(U_{FN}^i \sum_{j \in \mathcal{N}(i)} \left(\alpha_{off,ij}^l \left(E_{rx}^{ji} + E_{com}^j + E_{tx}^{ji} \right) + E_{id}^j \right) + \left(1 - U_{FN}^i \right) \cdot \left(\sum_{j \in \mathcal{N}(i)} \left(\alpha_{off,ij}^l \left(E_{tx}^{ij} + E_{rx}^{ij} \right) \right) + \alpha_{loc}^l E_{com}^i + E_{id}^i \right) \right) \right\} \quad (17)$$

$$\min_{\boldsymbol{\alpha_{off}}} \left\{ \frac{\sum_{i=1}^{N} \sum_l D_i^l(\alpha_{off,i}^l)}{\sum_{i=1}^{N} \sum_l \Lambda_l^i} \right\}$$

subject to:

$$\begin{cases} \eta_{comp_m} > \eta_{comp_i} \quad \forall m, i & (18a) \\ L_{s_l} > L_{r_l} & (18b) \\ P_{com}^i \geq \{P_{tx}^i; P_{rx}^i\} & (18c) \\ d_{i,\iota} \leq R \quad u_\iota \in \mathcal{U} & (18d) \\ d_{i,m} \leq F \quad a_m \in \mathcal{A} & (18e) \\ \alpha_{loc,i}^l + \sum_{j \in \mathcal{N}(i)} \alpha_{off,ij}^l = 1 & (18f) \end{cases}$$

where $\boldsymbol{\alpha_{off}}$ is the set of the offloaded portions of all tasks at a given time instant, and

$$\Lambda_l^i = \begin{cases} 1 & \text{if the } i\text{th FN generates a task to be processed} \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

allows to consider the total number of tasks generated by the FNs. Hence, the minimization of the FN energy consumption corresponds to finding the optimal partial offloading parameter $\boldsymbol{\alpha_{off}}$ allowing to minimize the energy consumed by all the FNs, i.e., including both RFN and CFN, or task delay, corresponding to finding the optimal partial offloading parameter $\boldsymbol{\alpha_{off}}$ allowing to minimize the task processing delay, defined as the ratio between the time needed for processing all the tasks generated at a certain time instant and the overall number of tasks generated within the same time interval; they are shown in (17).

Constraint (18a) introduces the hypothesis that the processing speed of F-APs is higher than FNs', that is at the basis of every Fog Network deployment. Constraint (18b) shows that the length of the requested packet is higher than the packet result. It is shown in Constraint (18c) that computing power for an FN is higher than transmission and reception power, leading the offloading as a feasible solution. Constraint (18d) ensures that the distance between two FNs should not exceed threshold R, which is the FN coverage area. Likewise, the distance between an F-AP and an FN should be smaller than threshold F as shown in Constraint (18e). The constraint that the local computation plus the offloading of the ith FN should be equal to one is shown in (18f). In the following section we propose a low complexity solution, by decomposing the problem in three steps, and exploiting two architectural hypotheses.

IV. Centralized and Distributed Partial Offloading Approaches

In order to solve the problem we resort to a decomposition in three steps. At first, we will classify the nodes based on their energy status. This selection allows to divide the nodes into groups where the higher energy nodes are able to process the tasks for other nodes, while the lower energy nodes benefit from offloading. In the second step, each FN belonging to the lower energy group selects the potential nodes for offloading the processing task; this step is performed in two possible ways by resorting to a centralized and a distributed approach. Finally, in the third step, each RFN is optimally selecting the portion to be offloaded to each of the selected nodes.

On one hand we are dealing with a centralized approach where the offloading decision is taken from a central entity supposed to be able to know the status of each node, while in the distributed approach each FN is selfishly deciding its offloading policy by optimizing the offloaded portion among the nearby nodes.

We are considering in this paper that the tasks can be computed by both F-APs and FNs, and we have also categorized the FNs in CFNs, those that can perform the computation, and RFNs, those that are offloading a task; the basic idea of the centralized architecture is that the RFNs and the related offloaded portion are centrally selected, while in the distributed architecture the RFNs select the CFNs and the F-APs for offloading.

A. FN Classification

Since one of the two objectives we are pursuing is related to the minimization of the energy, the first step aims at classifying

Fig. 2. The 3-quantile function of the remaining energy level distribution

the FNs based on their energy level. We suppose that the FNs having a higher remaining energy are better candidates for performing the computation of the incoming tasks. On the other hand, FNs with a lower remaining energy are preferred to offload the computation to others to save energy. FN classification is updated every time a new task is generated by FNs, so that the classification is based on the most updated remaining energy level. To this aim, we exploit a 3-quantile function to classify the FNs considering their remaining energy level. All FNs are classified into two lists, High Power FNs (HPFNs) and Low Power FNs (LPFNs), as shown in Fig. 2. The FNs whose remaining energy is higher than the upper quantile index, $E_{r,Q2}$, of the energy level distribution of all FNs are considered as HPFNs, and the rest are the LPFNs. We define the LPFN and HPFN lists, $\Im_{LPFN}$ and $\Im_{LPFN}$, at time instant t respectively as:

$$\Im_{LPFN}(t) = \left\{u_i | E_r^i(t) \leq E_{r,Q2}(t)\right\} \quad (20)$$

$$\Im_{HPFN}(t) = \left\{u_i | E_r^i(t) > E_{r,Q2}(t)\right\} \quad (21)$$

The upper quantile index is:

$$E_{r,Q2}(t) = \inf\left\{E_r^i(t), i = 1, \ldots, N | p \leq F_{E_r}(i)\right\} \quad (22)$$

where p is equal to 2/3, in case of upper 3-quantile, and $F_{E_r}(i)$ represents the distribution of the remaining energy of all FNs. The pseudo-code of the FN classification is shown in Algorithm 1, where, for each FN (line 3), the remaining energy at time instant t is compared with the upper quantile index (line 4) in order to classify the FNs into one of the two lists, i.e. $\Im_{LPFN}$ and $\Im_{LPFN}$ (lines 5-7).

Algorithm 1 3-Quantile Function

1: **Input:** E_r^i $i = 1, \ldots, N$
2: **Output:** $\Im_{LPFN}$ and $\Im_{HPFN}$
3: **for** each $u_i \in \mathcal{U}$ **do**
4: **if** $E_r^i(t) \geq E_{r,Q2}$ **then**
5: $\Im_{HPFN} \leftarrow u_i$
6: **else**
7: $\Im_{LPFN} \leftarrow u_i$
8: **end if**
9: **end for**

B. Architectural Approaches

The second step deals with the offloading decision. To this aim, two approaches have been considered.

1) Centralized Offloading Approach: In the centralized approach, the idea is that of primarily selecting the nodes able to process the tasks, i.e, the CFN; such nodes will select the nearby RFNs. To this aim we resort to a cluster architecture where the FNs can be classified in two types: Fog Cluster Heads (FCHs) and Fog Cluster Members (FCMs). Each cluster can be composed of one FCH and several FCMs. The FCHs are selected in a way that they are able of performing the computations of the tasks requested by the FCMs within their cluster [31]. Hence, the FCHs result to be the CFN while the FCMs are the RFNs.

The cluster formation is started by the FCHs (or CFNs), which are taken from $\Im_{HPFN}$, due to their higher energy amount. Each FCH considers potential FCM candidates, taken from $\Im_{LPFN}$, for the cluster formation as long as they are within its coverage area. FCMs are better candidates for becoming RFN due to their lower energy amount and, hence, asking for offloading to the CFN represented by the FCH. We have considered two policies based on the two layers for the computation offloading:

(a) FN layer
(b) FN and F-AP layers

In the first policy, we are only considering the presence of the FNs in the network. Hence, FCMs partially offload their tasks to the associated FCHs. In this case, the set of clusters is $C^{\mathcal{FN}} = \{c_1^{FN}, \ldots, c_g^{FN}, \ldots, c_G^{FN}\}$ and $|C^{\mathcal{FN}}| \leq |\Im_{HPFN}|$, where the gth cluster is defined as:

$$c_g^{FN} = \{u_i | u_i \in \Im_{LPFN}, |c_g^{FN}| < c_{cap}^{FN}, d_{g,i} \leq R\} \quad (23)$$

where c_{cap}^{FN} is the capacity of a cluster corresponding to the maximum number of cluster members. Similar to the FNs classification step, this procedure is updated every time a new task is generated, since the FNs' energy level change in a different way depending on the role they have. Indeed, through the clusters updating, the FNs having consumed more energy, i.e., FCHs, might change their role to FCMs, and the reverse. This approach results also in increasing the overall life time of the network by indirectly equalizing the FNs energy level by allowing a higher consumption for those nodes having a higher amount of energy and a lower consumption for the FNs having a lower amount of energy.

However, the FCMs not being associated to any cluster, are inserted into the set $\mathcal{L}_1$, which is the set of nodes performing the computation locally including all the FCHs generating their own tasks; it is defined as:

$$\mathcal{L}_1 = \left\{u_i | \{u_i \in \Im_{LPFN}, u_i \notin c_g^{FN}\}; \{u_i \in \Im_{HPFN}\}\right\} \forall g \quad (24)$$

On the other hand, in the second policy, both FNs and F-APs layers are available. Hence, FCMs can partially offload to the associated FCH and F-APs. In this case, the FCMs which are not in any clusters can still exploit the F-APs if available. Likewise, the FCHs can partially offload to the F-APs within their coverage area. As shown in Fig. 3, the FCMs can be connected to the FCHs and the nearby F-APs. Likewise, the FCHs (or the CFNs) are also able to offload to nearby F-APs their own tasks. The set of F-APs cluster having FNs in their coverage area is $C^{\mathcal{F-AP}} = \{c_1^{F-AP}, \ldots, c_m^{F-AP}, \ldots, c_M^{F-AP}\}$ and $|C^{\mathcal{F-AP}}| \leq M$, where the set of FNs connected to the mth F-AP is defined as:

$$c_m^{F-AP} = \left\{u_i | u_i \in \{\Im_{LPFN}; \Im_{HPFN}\}, |c_m^{F-AP}| < c_{cap}^{F-AP}, d_{m,i} \leq F\right\} \quad (25)$$

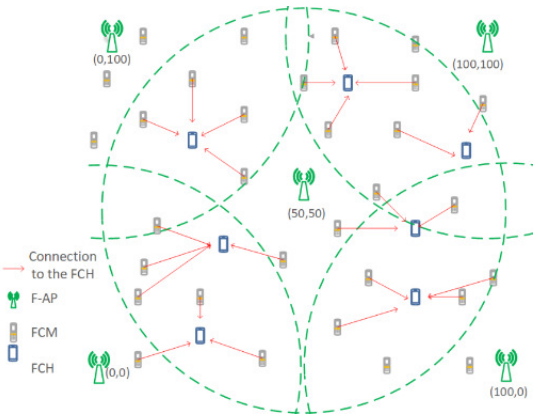


Fig. 3. Centralized Architecture

where c_{cap}^{F-AP} is the capacity of an F-AP corresponding to the maximum number of nodes that an F-AP can manage. However, the FNs not able to offload to any neighboring nodes belong to the local list, $\mathcal{L}_2$, defined as:

$$\mathcal{L}_2 = \{u_i | \{u_i \in \Im_{LPFN}, u_i \notin c_g^{FN}, u_i \notin c_m^{F-AP}\}; \{u_i \in \Im_{HPFN}, \notin c_m^{F-AP}\}\}, \ \forall g, m \quad (26)$$

The pseudocodes of the centralized architecture when using the policy limited to the first layer (a) or both layers (b) are shown in Algorithm 2 and 3, respectively. Algorithm 2 has as input the two sets of nodes $\Im_{HPFN}$ and $\Im_{LPFN}$ (line 1), having the HPFNs and LPFNs previously categorized, while the output (line 2) is represented by the set of clusters $C^{\mathcal{FN}}$, including one FCH and at least one FCM each, and $\mathcal{L}_1$, the set of nodes performing the local computation. The algorithm starts by considering each HPFN as an FCH candidate (lines 3-4), and, for each of them, the FCMs, selected among the LPFN, having a distance with respect to the selected FCH lower than the coverage range, are put into its cluster, up to the cluster maximum capacity (lines 5-9). In the end, the remaining LPFNs and all the HPFNs are put into the set $\mathcal{L}_1$, the list of the nodes performing the local computation (lines 11-16). Similarly, Algorithm 3 has as input the two sets of nodes $\Im_{HPFN}$ and $\Im_{LPFN}$ (line 1), while the output is represented by the set of clusters $C^{\mathcal{FN}}$, the set of clusters $C^{\mathcal{F}-\mathcal{AP}}$, including one F-AP and at least one FN each, and $\mathcal{L}_2$, the set of nodes performing the local computation (line 2). The algorithm starts by first populating the FN clusters, each one composed by one FCH, selected among the HPFNs, and at least one FCM, selected among the LPFNs. The selected FCMs should have a distance with respect to the FCH lower than the coverage range, and each cluster can be composed by a maximum number of FNs (lines 3-10). Moreover, due to the presence of the F-APs, the FNs are put into the F-AP clusters, if respecting the same constraints, i.e, the distance with respect to the F-AP and the F-AP cluster capacity (lines 11-23); this is performed for both LPFNs (lines 12-17) and HPFNs (lines 18-22). In the end, if there are FNs not belonging to any cluster, they are put into the set $\mathcal{L}_2$, the list of nodes performing a local computation (lines 24-29).

Algorithm 2 Centralized architecture (a)

1: **Input:** $\Im_{HPFN}$, $\Im_{LPFN}$
2: **Output:** $C^{\mathcal{FN}}$ and $\mathcal{L}_1$
3: **for** each $u_i \in \Im_{HPFN}$ **do**
4: $c_g^{FN} \leftarrow u_i$
5: **for** each $u_\iota \in \Im_{LPFN}$ **do**
6: **if** $d_{i,\iota} \leq R$ **and** $|c_g^{FN}| < c_{cap}^{FN}$ **then**
7: $c_g^{FN} \leftarrow u_\iota$; $|c_g^{FN}| = |c_g^{FN}| + 1$; remove u_ι from $\Im_{LPFN}$
8: **end if**
9: **end for**
10: **end for**
11: **for** each $u_i \in \Im_{HPFN}$ **do**
12: $\mathcal{L}_1 \leftarrow u_i$
13: **end for**
14: **for** each $u_\iota \in \Im_{LPFN}$ **do**
15: $\mathcal{L}_1 \leftarrow u_\iota$
16: **end for**

Algorithm 3 Centralized architecture (b)

1: **Input:** $\Im_{HPFN}$, $\Im_{LPFN}$
2: **Output:** $C^{\mathcal{F}-\mathcal{AP}}$, $C^{\mathcal{FN}}$ and $\mathcal{L}_2$
3: **for** each $u_i \in \Im_{HPFN}$ **do**
4: $c_g^{FN} \leftarrow u_i$
5: **for** each $u_\iota \in \Im_{LPFN}$ **do**
6: **if** $d_{i,\iota} \leq R$ **and** $|c_g^{FN}| < c_{cap}^{FN}$ **then**
7: $c_g^{FN} \leftarrow u_\iota$;$|c_g^{FN}| = |c_g^{FN}| + 1$
8: **end if**
9: **end for**
10: **end for**
11: **for** each $a_m \in \mathcal{A}$ **do**
12: $c_m^{F-AP} \leftarrow a_m$
13: **for** each $u_\iota \in \Im_{LPFN}$ **do**
14: **if** $d_{\iota,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
15: $c_m^{F-AP} \leftarrow u_\iota$; $|c_m^{F-AP}| = |c_m^{F-AP}| + 1$;remove u_ι from $\Im_{LPFN}$
16: **end if**
17: **end for**
18: **for** each $u_i \in \Im_{HPFN}$ **do**
19: **if** $d_{i,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
20: $c_m^{F-AP} \leftarrow u_i$; $|c_m^{F-AP}| = |c_m^{F-AP}| + 1$;remove u_i from $\Im_{HPFN}$
21: **end if**
22: **end for**
23: **end for**
24: **for** each $u_i \in \Im_{HPFN}$ **do**
25: $\mathcal{L}_2 \leftarrow u_i$
26: **end for**
27: **for** each $u_\iota \in \Im_{LPFN}$ **do**
28: $\mathcal{L}_2 \leftarrow u_\iota$
29: **end for**

2) Distributed Offloading Approach: Unlike centralized approach, in the distributed approach the idea is that the RFNs, belonging to the set $\Im_{LPFN}$, select the available nodes for task computation. In this architecture, the RFNs can offload to multiple available CFNs within their coverage area [14]. Differently from the centralized approach, where the RFNs were selected by the CFNs, in the distributed approach the RFNs autonomously select the CFNs among the $\Im_{HPFN}$. Likewise, the RFNs can select several F-APs in the second layer as long as they are within their coverage area. The scenario is represented in Fig. 4.

By taking into account the presence of FNs and F-APs, two policies have been considered for the distributed approach by exploiting the two layers for the computation offloading:

(a) FN layer
(b) FN and F-AP layers

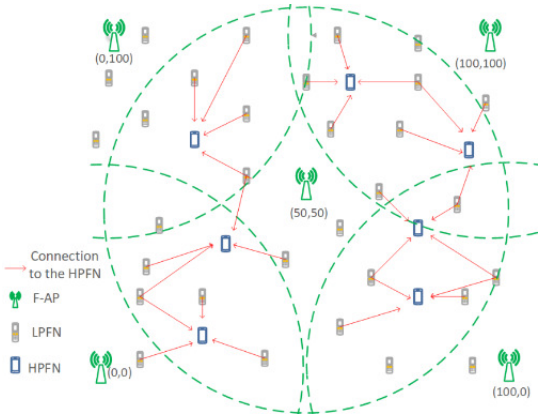


Fig. 4. Distributed Architecture

In the first layer, RFNs partially offload to the nearby CFNs while performing the remaining computation locally. Unlike the centralized method, where the RFNs were able to offload only to one associated CFN, in the distributed architecture they can offload to multiple CFNs at the same time. The CFNs available for one RFN are arranged into clusters. The set of clusters centered on RFNs is shown as $\mathcal{B}^{RFN,1} = \{b_1^{RFN,1}, \ldots, b_x^{RFN,1}, \ldots, b_X^{RFN,1}\}$, where $|\mathcal{B}^{RFN,1}| \leq |\mathfrak{I}_{LPFN}|$. Moreover, the RFNs that can be connected to the zth CFN are put in the group b_z^{CFN}; the set of groups compose $\mathcal{B}^{CFN} = \{b_1^{CFN}, \ldots, b_z^{CFN}, \ldots, b_Z^{CFN}\}$. The xth RFN when the F-APs are not available is defined as:

$$b_x^{RFN,1} = \left\{u_i | u_i \in \mathfrak{I}_{HPFN}, |b_z^{CFN}| < b_{cap}^{CFN}, d_{x,i} \leq R\right\} \quad (27)$$

where b_{cap}^{CFN} is the computing capacity of a CFN. On the other hand, in the second policy, RFNs can partially offload to both CFNs in the first layer and F-APs in the second layer. Moreover, the CFNs perform a local computation for their own tasks in the first policy while they can partially offload to the F-APs in the second policy. In case of working on both layers, the set of RFNs cluster in which both CFNs and F-APs are available is shown as $\mathcal{B}^{RFN,2} = \{b_1^{RFN,2}, \ldots, b_x^{RFN,2}, \ldots, b_X^{RFN,2}\}$. The xth RFN when both layers are available is defined as:

$$b_x^{RFN,2} = \left\{u_i | \{u_i \in \mathfrak{I}_{HPFN}, d_{x,i} \leq R, |b_z^{CFN}| < b_{cap}^{CFN}\}; \{a_m \in \mathcal{A}, d_{x,m} \leq F\}, |c_m^{F-AP}| < c_{cap}^{F-AP}\right\} \quad (28)$$

The pseudocodes of the distributed architecture for layer one (policy (a)) and both layers (policy (b)) are shown in Algorithms 4 and 5, respectively. In both cases the inputs are represented by the sets $\mathfrak{I}_{HPFN}$ and $\mathfrak{I}_{LPFN}$, including all the HPFNs and LPFNs previously categorized (line 1). Algorithms 4 has, as outputs, the set of RFNs centered clusters $\mathcal{B}^{RFN,1}$ and the list of nodes performing the local computation, $\mathcal{L}_1$ (line 2). In this algorithm, the RFNs, belonging to the LPFN set, select as many CFNs as possible as long as the distance and capacity constraints are respected (lines 3-10). In the end, the remaining LPFNs, not able to select any CFNs among the HPFNs list, and all the CFNs, are put into the set $\mathcal{L}_1$, the list of the nodes performing local computation (lines 12-17). In Algorithm 5, the outputs are instead the set of RFNs centered clusters $\mathcal{B}^{RFN,2}$, composed of both HPFNs and F-APs, the F-AP based clusters C^{F-AP} including the FNs within their coverage range, and the list of nodes performing the local computation, $\mathcal{L}_2$ (line 2). In this algorithm, the RFNs select as many CFNs (lines 4-9) and F-APs (lines 10-14) as possible respecting the distance and capacity constraints. Moreover, due to the presence of the F-APs, the HPFNs are put into the F-AP clusters, if respecting the same constraints, i.e, the distance with respect to the F-AP and the F-AP cluster capacity (lines 18-23). Finally, the remaining LPFNs and HPFNs are put into the list of the nodes performing local computation, $\mathcal{L}_2$ (lines 25-30).

Algorithm 4 Distributed architecture (a)

1: **Input:** $\mathfrak{I}_{HPFN}$, $\mathfrak{I}_{LPFN}$
2: **Output:** $\mathcal{B}^{RFN,1}$ and $\mathcal{L}_1$
3: **for** each $u_\iota \in \mathfrak{I}_{LPFN}$ **do**
4: **for** each $u_i \in \mathfrak{I}_{HPFN}$ **do**
5: **if** $d_{\iota,i} \leq R$ **and** $|b_z^{CFN}| < b_{cap}^{CFN}$ **then**
6: $b_x^{RFN,1} \leftarrow u_\iota$
7: $b_x^{RFN,1} \leftarrow u_i; |b_z^{CFN}| = |b_z^{CFN}| + 1$
8: **end if**
9: **end for**
10: **end for**
11: remove u_ιs which are in $\mathcal{B}^{RFN,1}$ from $\mathfrak{I}_{LPFN}$
12: **for** each $u_\iota \in \mathfrak{I}_{LPFN}$ **do**
13: $\mathcal{L}_1 \leftarrow u_\iota$
14: **end for**
15: **for** each $u_i \in \mathfrak{I}_{HPFN}$ **do**
16: $\mathcal{L}_1 \leftarrow u_i$
17: **end for**

Algorithm 5 Distributed architecture (b)

1: **Input:** $\mathfrak{I}_{HPFN}$, $\mathfrak{I}_{LPFN}$
2: **Output:** $C^{\mathcal{F}-\mathcal{AP}}$, $\mathcal{B}^{RFN,2}$ and $\mathcal{L}_2$
3: **for** each $u_\iota \in \mathfrak{I}_{LPFN}$ **do**
4: **for** each $u_i \in \mathfrak{I}_{HPFN}$ **do**
5: **if** $d_{\iota,i} \leq R$ **and** $|b_z^{CFN}| < b_{cap}^{CFN}$ **then**
6: $b_x^{RFN,2} \leftarrow u_\iota$
7: $b_x^{RFN,2} \leftarrow u_i$; $|b_z^{CFN}| = |b_z^{CFN}| + 1$
8: **end if**
9: **end for**
10: **for** each $a_m \in \mathcal{A}$ **do**
11: **if** $d_{\iota,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
12: $b_x^{RFN,2} \leftarrow a_m$; $|c_m^{F-AP}| = |c_m^{F-AP}| + 1$
13: **end if**
14: **end for**
15: **end for**
16: remove u_ιs which are in $\mathcal{B}^{RFN,2}$ from $\mathfrak{I}_{LPFN}$
17: **for** each $a_m \in \mathcal{A}$ **do**
18: **for** each $u_i \in \mathfrak{I}_{HPFN}$ **do**
19: **if** $d_{\iota,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
20: $c_m^{F-AP} \leftarrow u_i$; $|c_m^{F-AP}| = |c_m^{F-AP}| + 1$
21: **end if**
22: **end for**
23: **end for**
24: remove u_is which are in $C^{\mathcal{F}-\mathcal{AP}}$, from $\mathfrak{I}_{HPFN}$
25: **for** each $u_\iota \in \mathfrak{I}_{LPFN}$ **do**
26: $\mathcal{L}_2 \leftarrow u_\iota$
27: **end for**
28: **for** each $u_i \in \mathfrak{I}_{HPFN}$ **do**
29: $\mathcal{L}_2 \leftarrow u_i$
30: **end for**

A comparison between the centralized and distributed architectures is briefly shown in Tab. III.

TABLE III
COMPARISON OF THE CENTRALIZED AND DISTRIBUTED APPROACHES

	Centralized	Distributed
Selection Policy	RFNs are selected by CFNs and F-APs	RFNs select the CFNs and F-APs
RFNs	FCMs taken from $\Im_{LPFN}$	LPFNs taken from $\Im_{LPFN}$
CFNs	FCHs taken from $\Im_{HPFN}$	HPFNs taken from $\Im_{HPFN}$
FN Layer	connection with one CFN	connection with multiple CFNs
F-AP Layer	FCMs and FCHs	LPFNs and HPFNs

TABLE IV
PARAMETERS DEFINITION

Parameter	Definition
β_{ij}^{l}	Offloaded portion of the lth task from the ith node to the jth node
$\dot{\beta}_{ij}^{l}$	Offloaded portion of the lth task from the ith node to the jth node considering goal (a)
$\ddot{\beta}_{ij}^{l}$	Offloaded portion of the lth task from the ith node to the jth node considering goal (b)
γ	Estimation weight coefficient
$\tilde{T}_{off,i}^{l}$	Offloading time for the lth task of the ith node considering each of the estimation goals
$\tilde{\alpha}_{loc,i}^{l}$	Estimated local portion of the lth task of the ith node considering each of the estimation goals
$\tilde{E}_{CFN}^{j}$	The estimated energy consumption of the jth CFN
$\tilde{E}_{RFN}^{i}$	The estimated energy consumption of the ith RFN

C. Partial Offloading Estimation

Even if the problem in (17) cannot be solved in a closed way, in the previous sections we introduced two steps that allow to relax the problem. The problem relaxation allows to simplify the problem and scale it down by optimizing separately the amount of data to be offloaded. In this section, by exploiting this relaxation we will calculate in a closed form the optimal amount of data to be offloaded based on the constraints.

In order to evaluate the amount of data to be offloaded we proceed in a two step method. First of all, we estimate the portion to be offloaded to each of the available nodes; in order to do this we will consider that each RFN is aware of the processing power of the nearby computing nodes and data rate of each link. Then, considering the offloaded portion and the characteristics of the neighboring nodes, we estimate the portion that should be performed locally. With the proposed approach the estimation of the local and offloaded portion is performed in the same way in both centralized and distributed architectures.

1) Partial Offloading Portion estimation: To estimate the amount that should be offloaded to each of the available nodes that are going to perform the computation of a task portion, we have considered both data rate and computational power of the available neighboring nodes.

When the ith FN decides to offload a task, there are $j \in \mathcal{N}(i)$ available nodes, where $\mathcal{N}(i)$ is a set of neighboring nodes of the ith FN available for computing. As a result, the task can be divided into several portions to be offloaded to each of those available nodes. We define β_{ij}^{l} as the portion of lth task to be offloaded from the ith node to the jth node. Due to the impact of the offloaded portion on task processing delay and energy consumption of all FNs, we have considered two goals for the estimation of partial offloading portion, β_{ij}^{l}:

(a) Task processing delay
(b) Node energy consumption and task processing delay

If the task processing delay is considered, the amount of the lth task to be offloaded to the jth node can be defined as:

$$\dot{\beta}_{ij}^{l} = \gamma \cdot \frac{r_{ij}}{\sum_{j\in\mathcal{N}(i)} r_{ij}} + (1-\gamma) \cdot \frac{\eta_{comp_j}}{\sum_{j\in\mathcal{N}(i)} \eta_{comp_j}} \tag{29}$$

where γ is a coefficient giving a weight to the importance of the data rate and computational power in the estimation. This estimation considers the data rate of the link and the computational power of the node, for offloading a higher portion to the nodes with better characteristic. If both task processing delay and energy consumption are considered the amount of the lth task to be offloaded to the jth node can be defined as:

$$\ddot{\beta}_{ij}^{l} = \gamma \cdot \frac{\frac{r_{ij}}{E_{tx}^{ij}+E_{rx}^{ij}}}{\sum_{j\in\mathcal{N}(i)} r_{ij}} + (1-\gamma) \cdot \frac{\frac{\eta_{comp_j}}{E_{id}^{i}}}{\sum_{j\in\mathcal{N}(i)} \eta_{comp_j}} \tag{30}$$

In this formula the ith RFN's energy is affecting the delay and computational power metrics to estimate the portion to be offloaded to each available node; in particular, the energy spent in transmission and reception is affecting the data rate metric while the energy spent in idle affect the computational power metric. Hence, a higher portion will be offloaded to those nodes allowing to consume less energy. As a result not only delay is minimized but the energy consumption is also the target of the minimization. For the sake of readability, Tab. IV has been provided presenting the definitions of the parameters used in the equations of Section IV-C.

2) Local Computation Portion estimation: After having estimated the portions to be offloaded to the available neighboring nodes for computation, we have to estimate the amount that should be performed locally considering the characteristics of the available neighboring nodes.

Since one of the objectives is minimizing the delay, the idea is that of imposing that the amount of time spent for the local computation is equal to the amount of time spent for the offloading phase. This corresponds to minimize the idle time for any FN; hence imposing:

$$T_{loc,i}^{l} = T_{off,i}^{l} \tag{31}$$

where:

$$T_{loc,i}^{l} = \alpha_{loc,i}^{l} \frac{O_l}{\eta_{comp_i}} \tag{32}$$

and

$$T_{off,i}^{l} = \max_{j\in\mathcal{N}(i)} \left\{ \alpha_{off,ij}^{l} \frac{L_{s_l}}{r_{ij}} + T_{w_j}^{l} + \alpha_{off,ij}^{l} \frac{O_l}{\eta_{comp_j}} + \alpha_{off,ij}^{l} \frac{L_{r_l}}{r_{ij}} \right\} \tag{33}$$

corresponding to set the local computation time to the maximum among all the times spent for offloading to the neighbor

nodes of the ith RFN. The amount of data offloaded to the jth node from the ith node is represented by $\alpha^l_{off,ij}$. By resorting to the estimation of the partial offloaded amount obtained in (29) and (30), it is possible to write that:

$$\alpha^l_{off,ij} = \beta^l_{ij}(1-\alpha^l_{loc,i}) \tag{34}$$

where $\beta^l_{ij} = \dot{\beta}^l_{ij}$ or $\beta^l_{ij} = \ddot{\beta}^l_{ij}$ depending on the selected goal. By neglecting the queue waiting time, T^l_w, that is assumed to be unknown by FNs, we can rewrite (33) as:

$$\tilde{T}^l_{off,i} = \max_{j\in\mathcal{N}(i)} \left\{ \beta^l_{ij}(1-\alpha^l_{loc,i})\frac{L_{s_l}}{r_{ij}} + \beta^l_{ij}(1-\alpha^l_{loc,i})\frac{O_l}{\eta_{comp_j}} + \beta^l_{ij}(1-\alpha^l_{loc,i})\frac{L_{r_l}}{r_{ij}} \right\} \tag{35}$$

Hence, by exploiting (31), (32) and (35), it is possible to write:

$$\alpha^l_{loc,i}\frac{O_l}{\eta_{comp_i}} = \max_{j\in\mathcal{N}(i)} \left\{ \beta^l_{ij}(1-\alpha^l_{loc,i})\frac{L_{s_l}}{r_{ij}} + \beta^l_{ij}(1-\alpha^l_{loc,i})\frac{O_l}{\eta_{comp_j}} + \beta^l_{ij}(1-\alpha^l_{loc,i})\frac{L_{r_l}}{r_{ij}} \right\} \tag{36}$$

Through simple algebraic operations, it is possible to estimate the amount of local computation for the ith node as:

$$\tilde{\alpha}^l_{loc,i} = \frac{\max\limits_{j\in\mathcal{N}(i)} \left\{ \beta^l_{ij}\left(\frac{L_{r_l}}{r_{ij}} + \frac{L_{s_l}}{r_{ij}} + \frac{O_l}{\eta_{comp_j}} \right) \right\}}{\frac{O_l}{\eta_{comp_i}} + \max\limits_{j\in\mathcal{N}(i)} \left\{ \beta^l_{ij}\left(\frac{L_{r_l}}{r_{ij}} + \frac{L_{s_l}}{r_{ij}} + \frac{O_l}{\eta_{comp_j}} \right) \right\}} \tag{37}$$

In the end, any jth node is requested to process an amount equal to $(1-\alpha^l_{loc,i})\cdot\beta^l_{ij}\cdot O_l$ related to the lth task of the ith node.

Having estimated the local and offloading portion, considering $\tilde{\alpha}^l_{off,i} = 1-\tilde{\alpha}^l_{loc,i}$ and exploiting (13) we can calculate the total delay, when the ith FN is offloading to the available neighboring nodes, as:

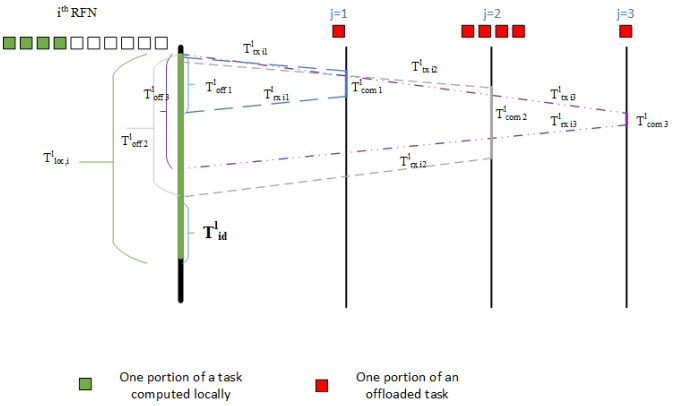
$$D^l_i(\tilde{\alpha}^l_{off,i}) = \max_{j\in\mathcal{N}(i)} \left\{ \beta^l_{ij}\tilde{\alpha}^l_{off,i}\frac{L_{s_l}}{r_{ij}} + T^l_{w_j} + \beta^l_{ij}\tilde{\alpha}^l_{off}\frac{O_l}{\eta_{comp_j}} + \beta^l_{ij}\tilde{\alpha}^l_{off,i}\frac{L_{r_l}}{r_{ij}}, \left(1-\tilde{\alpha}^l_{off,i}\right)\frac{O_l}{\eta_{comp_i}} \right\} \tag{38}$$

On the other hand, the energy consumed by the jth CFN, by exploiting (14), can be written as:

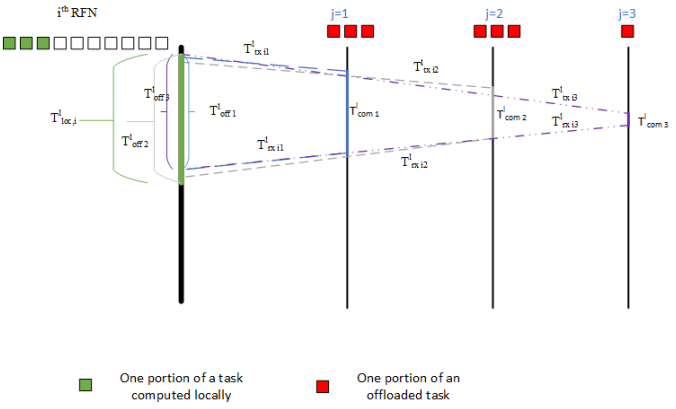
$$\tilde{E}^j_{CFN} = \tilde{\alpha}^l_{off,i}\beta^l_{ij}\left(E^j_{rx} + E^j_{com} + E^j_{tx}\right) + E^j_{id} \tag{39}$$

which is the energy consumption for receiving, computing and transmitting the offloaded part from the ith node to the jth node plus the idle energy of the jth node. Likewise, the energy consumption for the ith RFN, by exploiting (15), can be written as:

$$\tilde{E}^i_{RFN} = \sum_{j\in\mathcal{N}(i)} \left(\beta^l_{ij}\tilde{\alpha}^l_{off,i}\left(E^{ij}_{tx} + E^{ij}_{rx}\right) \right) + \left(1-\tilde{\alpha}^l_{off,i}\right)\cdot E^i_{com} + E^i_{id} \tag{40}$$



(a) Not Optimized



(b) Optimized

Fig. 5. Task delay for offloaded and local portions.

It is worth to be noticed that in (38), (39) and (40), β^l_{ij} could be equal to either (29) or (30) depending on the minimization goal.

In Fig. 5a the total delay for the lth task of the ith RFN is shown, when the $\alpha^l_{loc,i}$ and β^l_{ij} are not optimized. As seen, in this example 4 portions of the lth task are performed locally and the rest are offloaded to the 3 available nodes, which could be CFNs or F-APs. If the offloading portion is not optimized it might lead to having the local delay longer than the offloading delay, or the reverse. However, as shown in Fig. 5b, if $\alpha^l_{loc,i}$ and β^l_{ij} are optimized both local and offloading delay are equal by minimizing the idle time leading to a shorter delay.

V. Numerical Results

In this section, the numerical results obtained through computer simulations are presented. In the following we are comparing the performance for the single layer scenario with the performance of the two-layer scenario. Moreover, the comparison is performed between the delay minimization policy and the joint energy and delay minimization policy. These options have been considered in both centralized and distributed architectures.

The computer simulations are performed in Matlab where the considered parameters are listed in Tab. V. The simulation is performed for comparing the performance in terms

TABLE V
SIMULATION PARAMETERS

Parameter	Value
Dimension	100m x 100m
Communication Protocol	IEEE 802.11
Task size (L_{s_l})	[1-5] MB
Task result size (L_{r_l})	[0.2-1] MB
h_{ij}	Outdoor RRH/Hotzone, Model 1: Pico to UE [32]
Bandwidth (B)	10 MHz
Noise Density (N_T)	-174 dBm/Hz
FN to FN coverage range (R)	25 m
F-AP coverage range (F)	50 m
Maximum Initial energy ($E_r^I(0)$)	5000 J
Task Operation (O_l)	50G
FN Flops	15G FLOPS
F-AP Flops	150G FLOPS
Computation power (P_{com})	0.9 W
Idle power	0.01 W
FN Transmission power (P_{tx}^{FN})	1.3 W
F-AP Transmission power (P_{tx}^{F-AP})	1.5 W
FN reception power (P_{rx}^{FN})	1.1 W
F-AP reception power (P_{rx}^{F-AP})	1.3 W

of average task delay, average FN energy consumption and network lifetime as:

- Average Task Delay: The average time spent by a task for transmitting, waiting, computing and receiving back the result;
- Average Node Energy Consumption: The average energy all FNs have consumed per second;
- Network Lifetime: The time instant beyond which 20% of the FNs deplete their battery [13].

In the following we briefly describe the simulation environment implemented in Matlab. We hypothesize an area of 100×100 meters, with a variable number of FNs randomly positioned in the area, while there are 5 F-APs placed in the locations shown in Fig. 1, so that when working on two layers every FN can be always connected to at least one F-AP. Once the FNs are placed in the area, each of them randomly generates tasks with a Poisson distribution having average one task every 50 s; this value have been selected after a careful optimization. The size of tasks generated by each FN has a uniform distribution between 1 MB to 5 MB; the selection of this interval is driven by the application scenario that considers the case of nodes offloading heavy computing tasks to the nearby nodes for saving energy and reduce the overall delay. Although all of the FNs are identical in terms of computational power, we have also considered F-APs with higher computational capabilities resulting in a heterogeneous network. We have considered a battery capacity for all the FNs equal to 5000 Jouls; however, each FN has an initial random energy level between 70% and 100% of the battery capacity. When the tasks are generated by each FN, based on the architecture, either centralized or distributed, the available nodes for the task offloading are identified. In the centralized architecture the FCHs, belonging to the set $\Im_{HPFN}$, are selected and their Euclidean distance with all the other LPFNs, belonging to the set $\Im_{LPFN}$, is evaluated. The LPFNs enter the cluster as long as they meet the distance and cluster capacity requirement. In each run of the simulation the clusters are updated and, because the energy level of the FNs changes, the clusters and the connections change as well. In the distributed architecture, instead, the LPFNs identify the HPFNs within their coverage and select them by taking also into account the cluster capacity. The rest of algorithm for both centralized and distributed architecture works based on the policies defined in IV-B. In the end, the RFNs offload a task portion to each of the selected nodes based on the estimation they have made considering either, delay or joint delay and energy consumption.

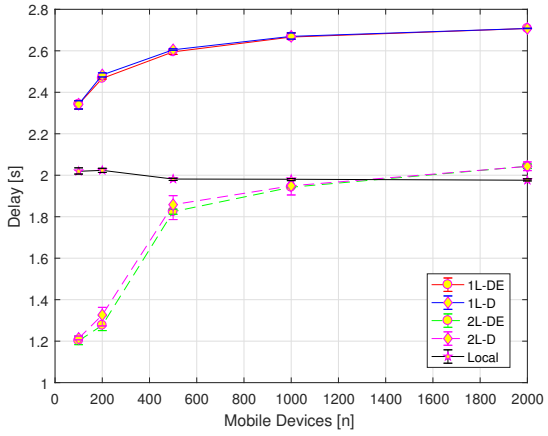
The simulation scenarios are defined based on the connections they have with the layers and their estimation goals; each simulation runs 1000 seconds. Moreover, in order to obtain steady results each simulation run has been carried out 10 times; to this aim, in the following figures, each point on the curves represent the average over the 10 runs, while the error bar represent the variance of the 10 runs. A fifth scenario labeled as *Local*, in which all FNs perform a local computation, has been also considered as a benchmark.

In the figures legend we are considering that the numbers corresponds to the number of layers involved in the computation (i.e., 1 and 2), while the letters D and DE show, respectively, delay minimization (29) and the joint delay and energy minimization (30) policies for the $\alpha_{loc,i}^l$ estimation.

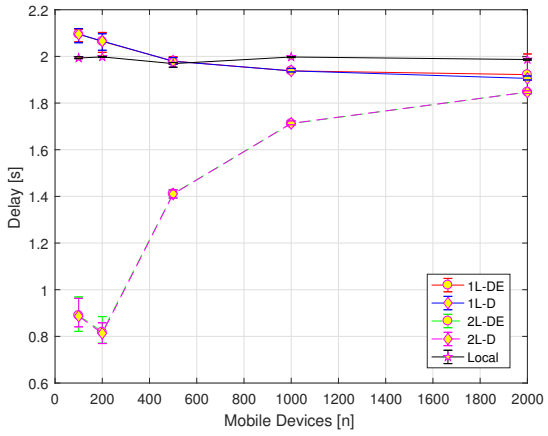
First of all, we evaluate the performance of the centralized and distributed architectures in terms of task delay, by comparing the scenario with only one layer and the scenario with both layers.

(a) FN Layer: As seen in both Fig. 6a and Fig. 6b, scenarios working only on the first layer seem to make smaller changes when the number of FNs is increasing comparing with the scenarios performing on both layers. In centralized scenario, when the number of FNs increases, the delay increases; this is because when there are few FNs, only few of them can be assigned to a CFN (FCH), and as a result more FNs perform a local computation which leads to a lower delay. On the other hand, increasing the number of FNs, the possibility of having an FCH nearby is higher and the delay for offloading to an FCH is higher than performing a local computation. However, in the distributed scenario that is the reverse. When the number of FNs is not high, there are few CFNs (HPFNs) available but when the number of FNs increases more CFNs (HPFNs) would be available for performing the computation in parallel which leads to a lower delay. The centralized scenario has a higher delay comparing with distributed scenarios due to the fact that when the centralized scenario is limited to the first layer, there is only one FCH in each cluster performing the computation for the RFNs (FCMs). However, in the distributed scenario each RFN (LPFNs) can offload to several CFNs (HPFNs) leading to parallel computation which results in a lower delay.

(b) FN and F-AP layers: As depicted in both Fig. 6a and Fig. 6b, the delay is increasing sharply when the number of FNs increases. When the number of FNs is reduced in both centralized and distributed approaches the FNs have

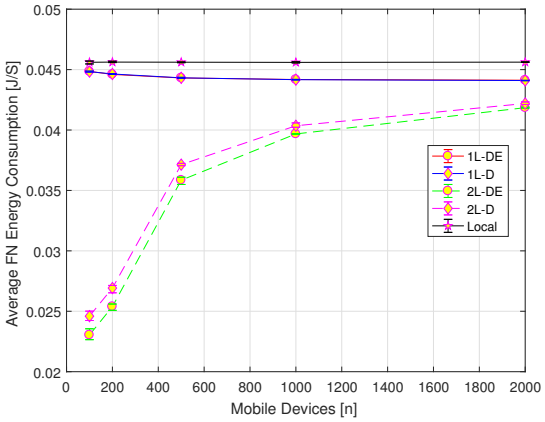


(a) Centralized Architecture

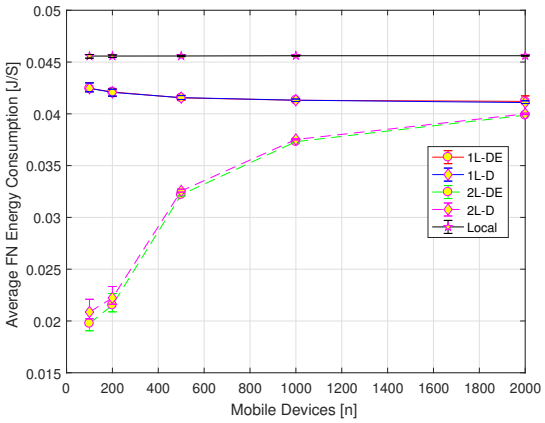


(b) Distributed Architecture

Fig. 6. Average Task Delay



(a) Centralized Architecture



(b) Distributed Architecture

Fig. 7. FN Energy Consumption

lower possibility of having CFNs nearby to offload; as a result higher portions are offloaded to the F-APs, leading to a lower delay. However, when increasing the number of FNs, more CFNs (FCH for centralized and HPFN for distributed) are available and a lower portion is offloaded to the F-APs in the second layer which leads to a higher delay.

In the end, as the number of FNs performing the computation in first layer increases, delay decreases, due to a parallel computation as depicted in Fig. 6b. Furthermore, when computational power is higher (i.e. offloading to F-APs), delay is also lower, as depicted in the scenarios working on both layers in both Figs. 6a and 6b. When all the FNs perform local computation the delay would be the same for both centralized and distributed scenarios as shown in scenario labeled *Local*.

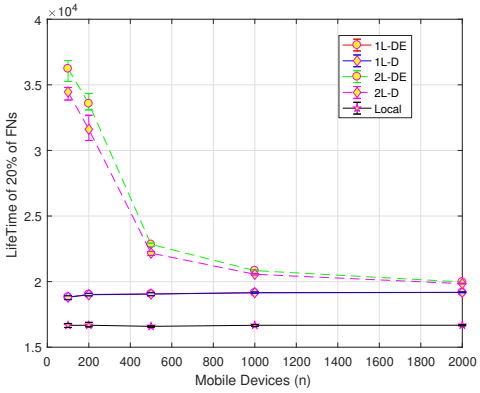
We evaluate then the performance of the centralized and distributed architectures in terms of average FN energy consumption by comparing the system performance in only one layer or both layers.

(a) FN Layer: According to both Fig. 7a and Fig. 7b, the scenarios limited to the first layer consume more energy in comparison with scenarios working on both layers by showing that offloading only to the nearby FNs results in higher energy consumption. Moreover, there is no significant difference if considering only the delay or both delay and energy optimization because there is no difference in offloading different task portions to different nodes and, in the end, the FNs are consuming energy for performing the computation regardless of the portion that was offloaded.

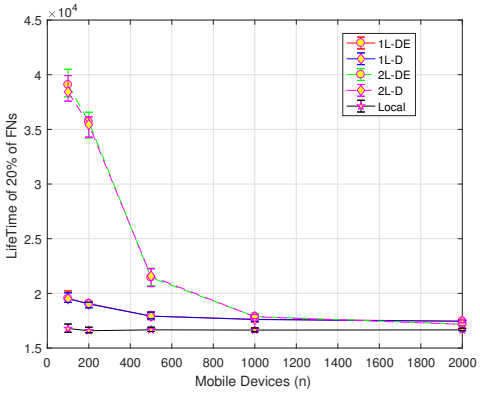
(b) FN and F-AP layers: When working on two layers, both centralized and distributed architectures result in a lower energy consumption, as seen in both Fig. 7a and Fig. 7b. By exploiting the availability of F-APs some portions are offloaded to the F-APs leading to a lower energy consumption. Moreover, considering both the delay and energy, a higher portion is offloaded to F-APs, resulting in a slight improvement.

The performance of the centralized and distributed architectures in terms of Network Lifetime is finally evaluated and the results are here compared in case of working only on the first layer or on both layers.

(a) FN Layer: As seen in both Fig. 8a and Fig. 8b, in the scenarios working on first layer, independently from the parameter considered for the estimation of $\alpha^l_{loc,i}$, the nodes go off at the same time and earlier than the scenarios working on two layers in both centralized and distributed architecture. Moreover, it could be seen that number of FNs do not have an impact on the network lifetime; this is due to the fact that all the generated tasks are processed by the FNs regardless of the portions they



(a) Centralized Architecture



(b) Distributed Architecture

Fig. 8. Network Lifetime (20%)

were offloaded.

(b) FN and F-AP layers: It can be seen in both Fig. 8a and Fig. 8b that when number of FNs is reduced in both centralized and distributed scenarios there are fewer options in the first layer and as a result more portions are offloaded to the second layer which results in a higher energy saving and longer network lifetime. However, when the FNs are more, there are also more options available in the first layer for offloading, resulting in higher energy consumption and shorter lifetime. Furthermore, in the centralized scenario in which there is maximum one available FCH for the FCMs, lifetime is slightly higher and this is due to the fact that in the distributed architecture there are more CFNs (HPFNs) involved in the first layer for computation, resulting in consuming more energy comparing with the centralized architecture. Furthermore, considering both delay and energy for the estimation of offloaded portion results in a longer lifetime in the centralized architecture.

VI. Conclusions

In this work, partial offloading in edge computing has been studied. Two architectures solutions, i.e., centralized and distributed, have been considered for the partial offloading scenario. We have proposed a heuristic solution based on relaxing some of the hypotheses of the partial offloading optimization problem, for minimizing task processing delay and FN energy consumption. Considering these two parameters we have estimated the portion to be offloaded to each of the available nodes at the network edge in order to meet the objectives. Simulation results demonstrate the impact of different parameters, i.e., delay and joint energy and delay, different layers and different architectures on the performance of the network in terms of FN energy consumption, task processing delay and network lifetime. It is possible to notice that the distributed architecture appears to be more appropriate for partial offloading scenarios when delay has higher priority, due to the fact that it can exploit parallel computation by a larger number of FNs. On the other hand, the centralized architecture appears to be more suitable when priority is given to FNs energy consumption and network lifetime, due to the fact that F-APs are more involved in the computation with respect to the distributed architecture. It is also interesting to note that, even if the presence of F-APs is always an advantage, when the FN density is increasing the performance obtained by using only the FN layer is similar to the scenario with both layers when priority is given to FNs energy consumption and network lifetime. This is an indication in favor of structureless wireless networks.

References

[1] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, Nov. 2018.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

[3] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, First Quarter 2014.

[4] A. u. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, First Quarter 2014.

[5] Y. Chao Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing - a key technology towards 5G," ETSI, White Paper 11, Sep. 2015.

[6] F. Giust, X. Costa-Perez, and A. Reznik, "Multi-access edge computing: An overview of ETSI MEC ISG," *IEEE 5G Tech Focus*, vol. 1, no. 4, Dec. 2017. [Online]. Available: https://5g.ieee.org/tech-focus/december-2017/multi-access-edge-computing-overview-of-etsi

[7] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[8] *OpenFog Reference Architecture for Fog Computing*, OpenFog Consortium Architecture Working Group White Paper, Feb. 2017.

[9] X. Chen, J. Zhang, and S. Misra, "Socially-aware cooperative D2D and D4D communications toward fog networking," in *Fog for 5G and IoT*, M. Chiang, B. Balasubramanian, and F. Bonomi, Eds. Hoboken, NJ, USA: Wiley Telecom, 2017.

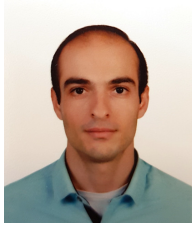
[10] D. Mazza, D. Tarchi, and G. E. Corazza, "A unified urban mobile cloud computing offloading mechanism for smart cities," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 30–37, Mar. 2017.

[11] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.

[12] D. Mazza, D. Tarchi, and G. E. Corazza, "A partial offloading technique for wireless mobile cloud computing in smart cities," in *2014 European Conference on Networks and Communications (EuCNC)*, Bologna, Italy, Jun. 2014.

[13] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A survey of network lifetime maximization techniques in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 828–854, Second Quarter 2017.

[14] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "An energy and delay-efficient partial offloading technique for fog computing architectures," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Singapore, Dec. 2017.
[15] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.
[16] M. Jo, T. Maksymyuk, B. Strykhalyuk, and C.-H. Cho, "Device to device based heterogeneous radio access network architecture for mobile cloud computing," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 50–58, Jun. 2015.
[17] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud RAN," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6118–6131, Sep. 2016.
[18] J. Oueis, E. Calvanese Strinati, and S. Barbarossa, "Distributed mobile cloud computing: A multi-user clustering solution," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016.
[19] K. Intharawijitr, K. Lida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5G cellular networks," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, Sydney, Australia, Mar. 2016.
[20] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2014.
[21] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[22] B. Assila, A. Kobbane, A. Walid, and M. E. Koutbi, "Achieving low-energy consumption in fog computing environment: A matching game approach," in *2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON)*, May 2018, pp. 213–218.
[23] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "Maximal energy efficient task scheduling for homogeneous fog networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 274–279.
[24] R. Venanzi, B. Kantarci, L. Foschini, and P. Bellavista, "MQTT-driven node discovery for integrated IoT-fog settings revisited: The impact of advertiser dynamicity," in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Mar. 2018, pp. 31–39.
[25] W. Wang, Q. Wang, and K. Sohraby, "Multimedia sensing as a service (MSaaS): Exploring resource saving potentials of at cloud-edge IoT and fogs," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 487–495, Apr. 2017.
[26] W. Zhang, Y. Wen, and H. H. Chen, "Toward transcoding as a service: energy-efficient offloading policy for green mobile cloud," *IEEE Netw.*, vol. 28, no. 6, pp. 67–73, Nov 2014.
[27] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[28] J. Cheng, Y. Shi, B. Bai, and W. Chen, "Computation offloading in cloud-RAN based mobile cloud computing system," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016.
[29] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.
[30] Y. D. Lin, E. T. H. Chu, Y. C. Lai, and T. J. Huang, "Time and energy aware computation offloading in handheld devices to coprocessors and clouds," *IEEE Syst. J.*, vol. 9, no. 2, pp. 393–405, Jun. 2015.
[31] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "An energy-aware offloading clustering approach (EAOCA) in fog computing," in *2017 International Symposium on Wireless Communication Systems (ISWCS)*, Bologna, Italy, Aug. 2017, pp. 390–395.
[32] *Further advancements for E-UTRA physical layer aspects*, 3GPP TR 36.814, Rev. 9.0.0, Mar. 2010.



Arash Bozorgchenani was born in Rasht, Iran on 19th December 1989. He has received a B.Sc. degree in Information Technology in 2013, and a M.Sc degree in Information Technology with a major in Computer Networks in 2016 in Iran. Since 2016, He has been a Ph.D. student in Electronics, Telecommunications and Information Technology at University of Bologna, Italy. He is University of Bologna's full scholarship recipient. He has also been involved in the national research project Gaucho (PRIN 2015) in Italy. His research interest is in the area of wireless communication and resource allocation in wireless networks. He is working more specifically on fog/edge computing toward IoT and 5G applications. He is an IEEE student member since 2017.

Daniele Tarchi [S'99, M'05, SM'12] was born in Florence, Italy, on 4th October 1975. He received his M.Sc. degree in telecommunications engineering and Ph.D. degree in informatics and telecommunications engineering from the University of Florence, Italy, in 2000 and 2004, respectively. From 2004 to 2010 he was a Research Associate with University of Florence, Italy. He is currently an Assistant professor at the University of Bologna, Italy.

His research interests are mainly on resource allocation techniques, heterogeneous networks, Cognitive Radios and Networks, and Wireless Multimedia Networks, applied to both terrestrial and satellite wireless communications. Recently he has mainly focused on Smart City scenarios, multimedia systems, Fog Networks, and Smart Grid. He has been involved in several national and international research projects, and collaborates with several European research institutes; he is now the local research responsible for the project Gaucho (PRIN 2015). He has published more than 100 papers, among which more than 30 on international journals.

Dr. Daniele Tarchi is an IEEE Senior Member since 2012. He is Editorial Board member for IEEE Wireless Communications Letters, IEEE Transactions on Vehicular Technology and IET Communications. He has been symposium co-chair for IEEE WCNC 2011, IEEE Globecom 2014 and IEEE Globecom 2018, and workshop co-chair at IEEE ICC 2015.

Giovanni Emanuele Corazza is a Full Professor at Alma Mater Studiorum University of Bologna, President of the CINECA consortium for supercomputing, founder of the Marconi Institute for Creativity, President of the Scientific Committee of the Fondazione Guglielmo Marconi, Member of the Marconi Society Board of Directors, and Member of the Partnership Board of the 5G Infrastructure Association. He was Member of the Board of Directors of the University of Bologna in the years 2012-2018, Head of the Department of Electronics, Computer Science and Systems (DEIS) in the years 2009-2012, Chairman of the School for Telecommunications in the years 2000-2003, Chairman of the Advanced Satellite Mobile Systems Task Force (ASMS TF), Founder and Chairman of the Integral Satcom Initiative (ISI), a European Technology Platform devoted to Satellite Communications, Member of the Board of the 5G Infrastructure Association and Vice-Chairman of the NetWorld2020 European Technology Platform in the years 2013-2016. In the years 1997-2012, he served as Editor for Communication Theory and Spread Spectrum for the IEEE Transactions on Communications. He is author of more than 300 papers, and received the Marconi International Fellowship Young Scientist Award in 1995, the IEEE 2009 Satellite Communications Distinguished Service Award, the 2013 Newcom# Best Paper Award, the 2002 IEEE VTS Best System Paper Award, the Best Paper Award at IEEE ISSSTA'98, at IEEE ICT2001, and at ISWCS 2005. His research interests are in creativity and innovation, 5G systems, navigation and positioning.