ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

An Energy and Delay-Efficient Partial Offloading Technique for Fog Computing architectures

This is the final peer-reviewed author’s accepted manuscript (postprint) of the following publication:

Published Version:
Arash, B., Daniele, T., Giovanni Emanuele Corazza, (2017). An Energy and Delay-Efficient Partial Offloading Technique for Fog Computing architectures. IEEE [10.1109/GLOCOM.2017.8254703].

Availability:
This version is available at: https://hdl.handle.net/11585/609302 since: 2020-12-15

Published:
DOI: http://doi.org/10.1109/GLOCOM.2017.8254703

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

12 March 2025

This is the final peer-reviewed accepted manuscript of:

A. Bozorgchenani, D. Tarchi and G. E. Corazza, "An Energy and Delay-Efficient Partial Offloading Technique for Fog Computing Architectures," *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Singapore, 2017, pp. 1-6.

The final published version is available online at DOI: https://doi.org/10.1109/GLOCOM.2017.8254703

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

An Energy and Delay-Efficient Partial Offloading Technique for Fog Computing architectures

Arash Bozorgchenani, Daniele Tarchi, Giovanni Emanuele Corazza
Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy
Email: {arash.bozorgchenani2,daniele.tarchi,giovanni.corazza}@unibo.it

***Abstract*—Fog computing is a fascinating paradigm which has drawn attention recently by bringing the cloud capabilities closer to the users. A fog computing infrastructure can be seen as composed by two layers: one including Fog Nodes (FNs) and another the Fog Access Points (F-APs). While FNs are usually battery operated, the F-APs are instead connected to the electrical networks having unlimited energy. Moreover, F-APs facilitate the computation of tasks due to their higher storage and computational capabilities compared to the FNs. Considering FN energy consumption and task processing delay, we propose a suboptimal partial offloading technique aiming at exploiting jointly both FNs and F-APs. The simulation results demonstrate how partial offloading has a profound impact on the network lifetime and reduces energy consumption and task processing delay by comparing the single and two layer architectures.**

I. Introduction

Due to the limitation in mobile resources and their low storage capacity, a new paradigm called Cloud Radio Access Network (C-RAN) has been introduced enabling mobile subscribers to outsource their tasks to a more powerful center whose capacity is considered high [1]. This technology is a combination of wireless and information technology industries which has brought the cloud computing into the Macro Base Stations (MBSs) [2]. However, C-RAN still needs a high-speed fronthaul for enabling the huge exchange of traffic between user equipments and MBSs. To overcome the disadvantages in C-RAN, Fog Radio Access Networks (F-RANs) has been proposed as an alternative to bring the computing capability of the cloud to the network edge aiming at minimizing the task processing delay and lowering the traffic at the fronthaul [3]. The advantages of fog computing have been also highlighted by ETSI [4], proposing several possible applications: active device location tracking, augmented reality content delivery, video analytics, radio access network aware content optimization, distributed content and DNS caching and application-aware performance optimization.

Two different types of nodes can be defined in a fog architecture [5]: Fog Nodes (FNs), battery operated nodes, and Fog Access Points (F-APs), usually connected to electrical network acting as interconnection to the backhaul and having higher computational capabilities. These two types of nodes are logically organized into two interconnecting layers; to this aim two communications paradigms are usually considered: Device to Device (D2D) communication between FNs, and infrastructured communications between FNs and F-APs. A third logical connection between FN and the centralized cloud exists as a backup when the fog infrastructure is not able to manage autonomously the user requests. In the following we consider not having this connection that should be indeed as less used as possible.

This work has been partially 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.

Among different applications that can be envisaged in a fog computing infrastructure, we focus here on computation offloading characterized by the possibility of offloading some tasks to be computed by the nearby devices. However, due to the limited FNs capabilities, sometimes they could not suffice in an efficient implementation of the fog infrastructure. To this aim a joint exploitation of both FN and F-AP is here considered. In particular we will focus here on a partial offloading approach enabling FNs to distribute high computational tasks among several FNs or F-APs [3]; by optimizing the partial offloading sharing among the devices, we aim at minimizing the FN energy consumption and the task processing delay while increasing the network lifetime [6].

Even if introduced only recently, the research community is very active on fog computing and networking issue. The architecture in [7] is broken down into several layers in a way that some cloudlets for mobile cloud computing are considered. It has been proved in [8] that offloading might not be always the best solution for reducing the energy consumption when intensive communication is required in the offloading process. The task offloading problem has been formulated in [9] as a joint radio and computational resources optimization. Energy consumption and latency have also been targeted in [10] for an offloading approach.

Differently from the previous works, we have considered FN energy consumption for selecting the FNs able to perform the computation. Moreover, we introduce a new paradigm working on both FN and F-AP layers considering the node energy consumption and the task processing delay for a sub-optimal solution to the partial offloading problem.

II. 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. All the FNs have computational and storage capabilities which should be exploited in a proper way; FNs can communicate among themselves within a specific range depending on the deployed wireless technology. On the other hand, in the second layer, there are some F-APs, whose set is shown as $\mathcal{C} = \{c_1, \dots, c_m, \dots, c_M\}$, with higher computational and storage capabilities able to communicate with the FNs. The F-APs have a wider range of communication comparing with the FNs and are able to aggregate the FNs' traffic requests. F-APs act as Mobile Edge Computing server which provides the ability of running multiple computation tasks simultaneously [5].

Herein FNs are considered to be fixed devices with the possibility of offloading their tasks to the neighboring FNs or to the upper layer F-APs for computation. The focus in our scenario is on increasing the interaction among nodes at the edge of the network, with the objective of minimizing the task processing delay and the FNs energy consumption. To this aim each FN having a task to be computed can have different choices: do a local computation, offload to a neighbor FN or offload to an F-AP in proximity; the goal of the proposed partial offloading technique is to select the amount of data to be offloaded to each of the possible candidates in order to minimize the FN energy consumption and the task processing delay. Each FN can be in one of four possible states $\mathcal{S} = \{tx, rx, com, id\}$: transmitting, receiving, computing or idle. While the first two states are referred to the interaction with other FNs or F-APs, the computing state refers to the computation performed in the FN itself (either for a local task or for an offloaded task), while the idle state refers to the idling occurring otherwise. To this aim, the overall energy consumed by the 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 is 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. On the other hand, each task can be computed locally or offloaded to a nearby FN or F-AP. Hence, the computational time for the lth task is defined as:

$$T^l_{com} = O_l / Flop \tag{3}$$

where O_l represents the number of operations required for computing the lth task and $Flop$ is the Floating-point Operation Per Second (FLOPS) which depends on the CPU of the processing device.

In case of offloading, each task should be transmitted, hence the transmission time for the lth task should be considered; it can be written as:

$$T^l_{tx,i} = L_{s_l} / r_i \tag{4}$$

where L_{s_l} is the size of the lth task requested from an FN and r_i is the data rate of the link between the ith FN and the receiving device. Later the result of the processed task should be sent back to the ith FN, leading to a reception time defined as:

$$T^l_{rx,i} = L_{r_l} / r_i \tag{5}$$

where L_{r_l} is the size of the result of the requested task sent back from the processing node to the source FN, and we suppose a symmetric channel in terms of data rate. Since r_i represents the data rate of the link, by considering the Shannon capacity formula, it can be defined as:

$$r_i = B \log_2 \left(1 + \frac{P^i_{tx}}{L(d) P_N}\right) \tag{6}$$

where B represents the bandwidth of the link, P^i_{tx} is the transmission power of the ith FN, $L(d)$ is the path loss at a distance d and P_N is the noise power. Noise power can be defined as $P_N = N_T B$, where N_T is the thermal noise.

All the devices are supposed to have an initial energy equal to E_0. All FNs consume a certain amount of energy when they transmit, receive or compute tasks or when they are idle. Therefore, by a certain time t, each FN has consumed $E^i_c(t)$ Joule of energy. Thus, the remained energy of the FNs at certain time instant t can be calculated as:

$$E^i_r(t) = E_0 - E^i_c(t) \tag{7}$$

where,

$$E^i_c(t) = \sum_{\tau=0}^{t} E^i_{FN}(\tau) \tag{8}$$

On the other hand, the delay for computing the lth task can be defined as:

$$D^l = T^l_{tx,i} + T^l_w + T^l_{com} + T^l_{rx,i} \tag{9}$$

where T^l_w is the waiting time for the lth task to be computed. The delay is the sum of the time required for sending a task, waiting for the task to be computed, computing the task and having the result back from the processing FN or an F-AP. Each FN is supposed to have a buffer, holding the tasks to be processed. The waiting time for the lth task can be defined as:

$$T^l_w = \sum_{\lambda=1}^{l-1} T^\lambda_{com} \tag{10}$$

which is the sum of the computation time for all tasks not yet been processed in the buffer of the ith FN represented by λ.

In partial offloading, only a portion of the computation load is delegated to another node, enabling an optimization of the saved energy and time [3]. Let us define α^l_{loc} as the amount of the lth task that should be performed locally and α^l_{off} as the amount that should be offloaded. This means that it is possible to write the time needed for offloading a task as:

$$T^l_{off} = \alpha^l_{off} T^l_{tx} + T^l_w + \alpha^l_{off} T^l_{com} + \alpha^l_{off} T^l_{rx}, \tag{11}$$

while the time for local computation, can be defined as:

$$T^l_{loc} = \alpha^l_{loc} T^l_{com}. \tag{12}$$

Hence the total delay for processing a task, in case of partial offloading should be rewritten as:

$$D^l = \min\{T^l_{off}, T^l_{loc}\}. \tag{13}$$

Similarly the energy consumed by each FN in case of partial offloading should be rescaled by considering that the time variables are multiplied by α^l_{off}.

The goal is to minimize the overall energy consumption for all the FNs in the network and to minimize the delay needed for processing all the tasks depending on our target. This leads to a formulation of the partial offloading problem as an Integer Linear Programing (ILP):

$$\begin{aligned} &\text{Minimize } \sum_i E^i_{FN} \\ &\text{Minimize } \sum_l D^l \end{aligned} \tag{14}$$

subject to

$$T^{F-AP}_{com} > T^l_{com} > T^l_{tx,FN} > T^l_{rx,FN} > 0 \tag{15}$$
$$P^{F-AP}_{com} \geq P^i_{com} \geq P^i_{tx} \tag{16}$$
$$T^l_w \geq 0 \tag{17}$$
$$E_0 \geq E^i_c(t) \geq 0 \tag{18}$$
$$d(u_i, u_k) \leq R \tag{19}$$
$$d(c_m, u_k) \leq F \tag{20}$$
$$\eta(u_i) \leq \overline{\eta_{u_i}} \tag{21}$$
$$\alpha^l_{loc} + \alpha^l_{off} = 1 \tag{22}$$
$$T^l_{off} + T^l_{loc} \leq Q \tag{23}$$

There are two objectives in the formulation, i.e., minimizing the FN energy consumption or task delay, respectively shown in (14). Constraint (15) introduces the hypothesis that the computing time of F-APs is higher than FNs', which itself is higher than transmission and receiving time of FNs. Likewise, power consumption follows the same order and it is shown in constraint (16). This ensures that in offloading there is a tradeoff between saving energy by transmitting instead of computing locally, while needing more time for transmitting and receiving instead of just computing. Constraint (17) shows that the waiting time for a task could be zero or more, depending on the buffer status $\mathcal{B}_{FN_j}$. The initial energy of all FNs is more than the amount they consume for transmitting or computing a task and this requirement is shown in Constraint (18). Constraint (19) 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 (20). Moreover, each FN is supposed to have a maximum traffic capacity $\overline{\eta_{u_i}}$ that limits the amount of instantaneous traffic $\eta(u_i)$ with the nearby nodes, as defined in (21). The constraint that the local computation plus the offloading should be equal to one is shown in (22). All the tasks have a specific time for having the result back, as a result the time spent for offloading a part of the task and performing the rest should not exceed the maximum acceptable delay for a task which is shown in Constraint (23).

Hence, partial offloading can be written as an ILP which has been proved to be NP-hard. Thus, in the following section we propose a suboptimal solution.

III. An Energy and Delay-Efficient Partial Offloading Approach

Our proposal is to divide the problem into four main phases taking into account the steps that should be followed in order to ensure optimal node selection and partial offloading decision for minimizing the FN energy consumption and the task processing delay. In this section, we propose a novel energy and delay efficient approach considering both FN and F-AP layers. To clarify the proposed solution better, we have considered four main steps that should be considered:

A. FN classification
B. LPFN selection
C. Local computation parameter
D. Partial offloading parameter

A. FN classification

Since the main aim is to decrease the overall energy consumption, the first step taken into account is devoted to classifying the nodes based on their remained energy. The idea is that if a node has a higher energy left it can be also used by other nodes for offloading their computation, while a low power node would like to offload its tasks. At first, all FNs are classified into two groups, High Power FNs (HPFN) and Low Power FNs (LPFN), using a quantile function that considers the distribution of the energy of all the FNs in the network.

In the proposed idea, the FNs whose energy is in the highest quantile of the FN energy level distribution of all the FNs, are seen as better candidates to be selected as HPFN, due to their capability to perform the computation of the incoming tasks. The rest of the FNs are considered as LPFNs which offload their tasks to other devices selected based on the policy defined in the following in the second step.

It is worth to be noticed that the FNs classification is performed at run time each time a new task should be executed; this ensures that the HPFN are always those FNs having the highest amount of energy.

B. LPFN selection

The second step is instead focused on the selection to be performed by the LPFNs of the most appropriate devices for offloading their tasks; this operation is performed only by the LPFNs since they are those requiring an external device for offloading the task. To this aim two policies are considered:

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

In the first policy, LPFNs tasks can be offloaded only to HPFNs or, if not possible, LPFNs perform the task locally. In the second policy, instead, the LPFNs not able to offload to any nearby HPFNs, are connected to the F-APs in their coverage area. The selection of the devices is performed based on the coverage areas.

C. Local computation parameter

The third step is instead devoted to selecting the amount of processing to be performed locally and to be offloaded. Since, in partial offloading, the amount of energy and time is affected by the percentage of computation and communication, α^l_{loc}, can be calculated by resorting to two different goal-s:

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

If node energy consumption is considered, the goal is to estimate $\alpha^l_{loc,i}$ in order to minimize the amount of energy spent by the ith FN. As a result, it is possible to define the percentage of local processing of the lth task as:

$$\alpha^l_{loc,i} = \frac{\sum_{k\in\mathcal{N}(i)} E^k_{off}}{\sum_{k\in\mathcal{N}(i)} E^k_{off} + E^k_{loc}} \tag{24}$$

where $\mathcal{N}(i)$ is the set of the neighbor devices for the ith LPFN, while:

$$E^i_{off} = \alpha^l_{off,i}E^i_{tx} + \alpha^l_{off,i}E^i_{rx} + \alpha^l_{off,i}E^i_{id} \tag{25}$$

is the energy spent for offloading the whole task, and

$$E^i_{loc} = E^i_{com} \tag{26}$$

is the energy spent for computing locally the whole task, with $\alpha^l_{loc,i} = 1 - \alpha^l_{off,i}$. The ratio defining $\alpha^l_{loc,i}$ allows to tradeoff between the energy spent for transmitting, receiving and remaining idle and the energy spent for performing locally the computation.

On the other hand, if the task processing delay is considered for defining $\alpha^l_{loc,i}$, the best strategy could be that of offloading an amount of task such that the time needed for local computing is the same for offloading. This corresponds to minimize the idle time and allows that the time needed for offloading corresponds to the time needed for the local computation, i.e.:

$$T^l_{loc} = T^l_{off} \tag{27}$$

where:

$$T^l_{loc} = \alpha^l_{loc,i}\frac{O_l}{Flop_i} \tag{28}$$

and

$$\begin{aligned} T^l_{off} &= \alpha^l_{off,i}T^l_{tx} + T^l_w + \alpha^l_{off,i}T^l_{com} + \alpha^l_{off,i}T^l_{rx} \\ &= \left(1-\alpha^l_{loc,i}\right)\frac{L_{s_l}}{r_i} + T^l_w + \left(1-\alpha^l_{loc,i}\right)\frac{O_l}{Flop} + \left(1-\alpha^l_{loc,i}\right)\frac{L_{r_l}}{r_i} \\ &= \max_{k\in\mathcal{N}(i)}\left\{(1-\alpha^l_{loc,i})\frac{L_{s_l}}{r_k}\right\} + T^l_w \\ &+ \max_{k\in\mathcal{N}(i)}\left\{(1-\alpha^l_{loc,i})\frac{O_l}{Flop_k}\right\} + \max_{k\in\mathcal{N}(i)}\left\{(1-\alpha^l_{loc,i})\frac{L_{r_l}}{r_k}\right\} \end{aligned} \tag{29}$$

where the maximum value for transmission, computation and reception among the nearby devices is considered, by hypothesizing that the computation is performed simultaneously by the selected devices. Since the transmission time to each HPFNs or F-AP is unknown at this point (we have still to decide how much each device should receive), we hypothesize to offload to each device an amount of task proportional to the related data rate, corresponding to ${}^{r_k}/_{\sum_{k\in\mathcal{N}(i)} r_k}$. Having this in mind, we can rewrite T^l_{off} as:

$$\begin{aligned} T^l_{off} &= (1-\alpha^l_{loc,i})\frac{L_{r_l}}{\sum_{k\in\mathcal{N}(i)} r_k} + T^l_w + (1-\alpha^l_{loc,i})\frac{L_{s_l}}{\sum_{k\in\mathcal{N}(i)} r_k} \\ &+ (1-\alpha^l_{loc,i})\max_{k\in\mathcal{N}(i)}\left\{\frac{O_l}{Flop_k}\frac{r_k}{\sum_{k\in\mathcal{N}(i)} r_k}\right\}. \end{aligned} \tag{30}$$

Since we assume that the requesting node is not aware of processing queues of the selected devices for offloading, we assume to ignore T^l_w and, by exploiting (28) and (30), it is possible to write that:

$$\alpha^l_{loc,i} = \frac{\frac{L_{r_l}}{\sum\limits_{k\in\mathcal{N}(i)} r_k} + \frac{L_{s_l}}{\sum\limits_{k\in\mathcal{N}(i)} r_k} + \max\limits_{k\in\mathcal{N}(i)}\left\{\frac{O_l}{Flop_k}\frac{r_k}{\sum\limits_{k\in\mathcal{N}(i)} r_k}\right\}}{\frac{L_{r_l}}{\sum\limits_{k\in\mathcal{N}(i)} r_k} + \frac{L_{s_l}}{\sum\limits_{k\in\mathcal{N}(i)} r_k} + \max\limits_{k\in\mathcal{N}(i)}\left\{\frac{O_l}{Flop_k}\frac{r_k}{\sum\limits_{k\in\mathcal{N}(i)} r_k}\right\} + \frac{O_l}{Flop_i}} \tag{31}$$

D. Partial offloading parameter

Finally, since the percentage of data that should be offloaded to all the selected HPFNs or F-APs corresponds to:

$$\alpha^l_{off} = 1 - \alpha^l_{loc} \tag{32}$$

it is possible to define the amount of data to be offloaded to each of the HPFNs or F-APs, by considering the data rate of each link as:

$$\beta_k = \frac{r_k}{\sum_{k\in\mathcal{N}(i)} r_k} \tag{33}$$

so that, at the end, the kth device is requested to process an amount of the task equal to $\alpha^l_{off} \cdot \beta^l_k \cdot O_l$.

The proposed approach with different policies is shown in Algorithm 1 and 2; these algorithms are executed every time a task has to be processed in order to ensure to always minimize the energy or the delay.

As shown in Algorithm 1, firstly the energy level of all FNs is compared with the quantile function index I and classified as LPFN or HPFNs. Then, according to the number of layers involved in the policy and the parameter considered for finding α^l_{loc}, and considering the capacity and coverage area of the destination device, LPFNs are assigned to them.

IV. Numerical Results

In this section, the numerical results obtained through computer simulations are presented. We consider to have 4 policies: on one side the possibility to use only the nearby FNs or both FNs and F-AP, and on the other side to use the FN energy algorithm or the task processing delay algorithm.

The simulation is performed in Matlab; the information regarding the parameters, set according to the 3GPP recommendations [11], are shown in Tab. I. The remained energy of FNs is considered to have a random value lower than the initial energy at the beginning of simulation in order to have a

Algorithm 1 LPFN assignment

```
Input: $\mathcal{U}$
Output: HPFN and F-AP list
Quantile ($\mathcal{U}$) which gives $I$
for each $u_i \in \mathcal{U}$ do
    if $u_i \geq I$ then
        $HPFN \leftarrow u_i$
    else
        $LPFN \leftarrow u_i$
    end if
end for
if LPFN selection=a then
    for each LPFN$\in \mathcal{U}$ do
        for each $HPFN$ do
            if d($HPFN_j$,$LPFN_k$) $\leq R$ then
                $HPFN_j list \leftarrow LPFN_k$
            end if
        end for
        $LocalCom \leftarrow$ rest of the $LPFNs$
    end for
else if LPFN selection=b then
    for each LPFN$\in \mathcal{U}$ do
        for each $HPFN_j$ do
            if $d(HPFN_j, LPFN_k) \leq R$ then
                $HPFN_j list \leftarrow LPFN_k$
            end if
        end for
        for each $F - AP_M$ do
            if $d(F - AP_M, LPFN_k) \leq F$ then
                $F - AP_M list \leftarrow LPFN_k$
            end if
        end for
    end for
end if
```

Algorithm 2 α^l_{loc} and β_i

```
Input: HPFN and F-AP list
Output: $\alpha^l_{loc}$, $\beta_k$
for each LPFN$\in \mathcal{U}$ do
    if Local computation Parameter=Energy then
        $\alpha^l_{loc}$ calculation using (24)
    else
        $\alpha^l_{loc}$ calculation using (31)
    end if
    $\beta_k$ calculation using (33)
end for
```

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Dimension	200m x 200m
Communication Protocol	IEEE 802.11
Task size (L_s)	5 MB
Task result size (L_r)	1 MB
Path loss ($L(d)$)	140.7+36.7*log10(d) dB [11]
Bandwidth (B)	10 MHz
Thermal noise (N_T)	-174 dBm/Hz [11]
FN to FN coverage range (R)	25 m
F-AP coverage range (F)	100 m
Initial energy (E_0)	5000 J
Task Operation (O)	50G
FN Flops	15G FLOPS
F-AP Flops	150G FLOPS
Computation power (P_{com})	0.9 W
Idle power	0.3 W
FN Transmission power (P_{tx})	0.032 W
F-AP Transmission power (P_{tx})	0.250 W
FN reception power (P_{rx})	0.9 W
Maximum Number of FNs	2000

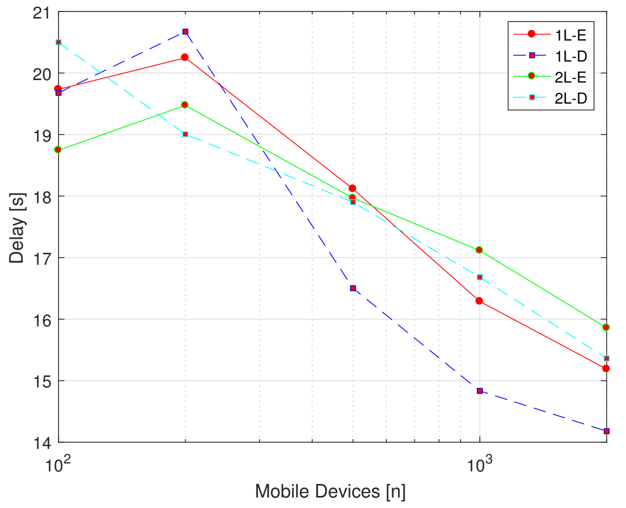

Fig. 1. Average Task Delay

realistic scenario. The simulation is carried out once for 12500 seconds in terms of average task delay, average node energy consumption and network lifetime, defined as:

- Average Task Delay: The average time spent for a task for transmitting, waiting, computing and receiving back the result (See (9)).
- Average Node Energy Consumption: The average energy that all FNs have consumed (See (8)).
- Network Lifetime 2 (NL2): The time instant beyond which 20 percent of the FNs deplete their battery, as defined in [6].

We hypothesize an area of 200x200 meters, with a variable number of FNs, while the F-AP are 5 and placed so that every FN can be always connected to at least one F-AP; the task generation rate is randomly performed with an average of 0.1 tasks per second for each FN. For each line in the figures the number corresponds to the number of layers involved in the computation while the letters E and D show respectively the energy and delay algorithms considered for α_{loc} evaluation.

In Fig. 1 the performance in terms of average task delay is depicted. It is possible to notice the impact of the layers in different FN densities. For a reduced number of nodes the presence of the F-AP helps since the FN density is low and hence some of the nodes could not have any neighbor FNs; hence the presence of F-AP helps. On the other side when the number of FNs becomes large it is possible to see that the lowest delay at the edge is correctly exploited by resorting in a lower delay. Finally, it is possible to note that the task processing delay algorithm performs better than the FN energy consumption algorithm.

The performance in terms of average FN energy consumption is depicted in Fig. 2. Conversely we can see here that the energy minimization algorithms perform better, as expected, while the effect of the layers is less evident than for the delay.

Interestingly in Figure 3 it can be seen that lifetime of the network, corresponding to the time when 20% of the FNs have

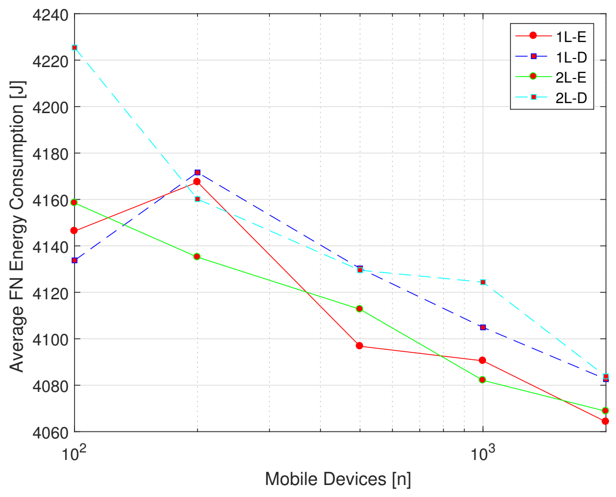

Fig. 2. Average FN Energy Consumption

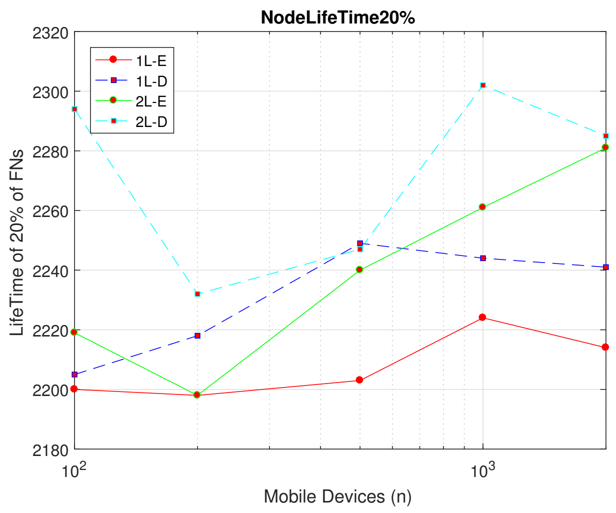

Fig. 3. Network Lifetime 2 (NL2)

finished their energy, is affected by the presence of the layers. This is expected since the presence of additional devices, the F-AP, having unlimited energy have undoubtedly a positive effect of prolonging the life of the nodes.

Finally, in order to have a good understanding of the rule of the task generation rate in the performance in terms of delay, we have presented Tab. II for 1000 FNs, while changing the task generation rate from 0.01 tasks per second, to 0.2 tasks per second. By comparing the results between 1 and 2 layers with the same algorithm, it is possible to see that a low generation rate using 1 layer, allows to gain in terms of delay, while in case of higher generation rate it is better to have 2 layers. This is as expected since the F-AP are used in helping the FNs in higher demand situations like those happening in case of higher generation rates.

The simulation results underscore that partial offloading to the F-APs can greatly enhance network lifetime and reduce average task delay. Moreover, considering the FN energy consumption or delay for deciding how much should be offloaded, average task delay and Average FN energy consumption can be sharply reduced.

TABLE II
DELAY PERFORMANCE IN DIFFERENT TRAFFIC GENERATION RATE

Policy	0.01(task/s)	0.02(task/s)	0.05(task/s)	0.1(task/s)	0.2(task/s)
1L-E	139.27 s	73.12 s	30.74 s	17.33 s	10.86 s
1L-D	124.61 s	64.40 s	27.95 s	16.20 s	10.47 s
2L-E	144.14 s	73.38 s	30.32 s	15.97 s	9.49 s
2L-D	135.22 s	67.81 s	28.61 s	15.66 s	9.54 s

V. CONCLUSIONS

In this paper, a partial offloading approach for fog computing is introduced. We have defined various policies considering energy consumption and task processing delay for deciding the amount of tasks to be offloaded. By considering also the F-AP for offloading it is possible to increase the network lifetime, especially in high demand scenarios that consume much more the network resources.The two algorithms can be selected depending on the scenario request that may go to an energy or a delay reduction.

REFERENCES

[1] T. Nishio, R. Shinkuma, and T. Takahashi, “Service-oriented heterogeneous *Proceedings of the first international workshop on Mobile cloud computing & networking*, Bangalore, India, Jul. 2013, pp. 19–26.

[2] M. Peng, Y. Li, Z. Zhao, and C. Wang, “System architecture and key technologies for 5G heterogeneous cloud radio access networks,” *IEEE Netw.*, vol. 29, no. 2, pp. 6–14, Mar. 2015.

[3] 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.

[4] 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. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[5] 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.

[6] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. Hanzo, “A survey of network lifetime maximization techniques,” *IEEE Commun. Surveys Tuts.*, 2017.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] *Further advancements for E-UTRA physical layer aspects*, 3GPP TR 36.814, Rev. 9.0.0, Mar. 2010.