Smart Probabilistic Fingerprinting for Indoor Localization over Fog Computing Platforms

Andrea Sciarrone†, Claudio Fiandrino⋆, Igor Bisio†, Fabio Lavagetto†, Dzmitry Kliazovich⋆, Pascal Bouvry⋆

† Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN), University of Genoa, Italy

⋆ Computer Science and Communications Research Unit, University of Luxembourg, Luxembourg

E-mails: †{firstname.lastname}@unige.it, ⋆{firstname.lastname}@uni.lu

***Abstract*—Indoor navigation and localization are becoming fundamental services nowadays. WiFi-based solutions such as FingerPrinting (FP) are the most widely adopted techniques for positioning and provide better results if compared to other approaches. They require to compare WiFi Received Signal Strength (RSS) with an pre-computed radio map called fingerprint. The recently proposed *Smart* Probabilistic FingerPrinting (P-FP) algorithm reduces the computational complexity of the traditional FP approach without any accuracy detriment. On the other hand, fog computing has emerged as a new promising paradigm in the recent years, which extends traditional mobile cloud computing capabilities towards the edge of the network and enables location-aware services. In this paper we propose to offload *Smart* P-FP computation over a fog platform exploiting a novel distributed algorithm. Performance evaluation validates the effectiveness of the proposed approach with the analysis of: *i)* the amount of power saved and *ii)* the efficiency of candidate selection process for offloading. Having 2 or more devices in the vicinity contributing to the computation makes offloading beneficial from a power standpoint. The offloading effectiveness increases with the number of devices willing to contribute and the amount of data to be transferred. Power savings can be as high as 80% if compared with local computation.**

I. Introduction

In modern societies, people perform the majority of their activities, including business, entertainment and socialization, indoors [1]. As a result, the demand for indoor navigation and positioning services is continuously increasing. Several technologies provide positioning and navigation services to the users. Global Positioning System (GPS), 3G/4G, WiFi, and Bluetooth signals can be employed for either indoor and outdoor navigation. As GPS signals are usually unavailable inside buildings, WiFi positioning has become a common solution for indoor navigation and positioning due to the brought advantages [2]. It operates relying on the common widely deployed WiFi infrastructure available almost everywhere. Furthermore, WiFi-based solutions provide better accuracy if compared to inertial navigation solutions, which rely on accelerometer, gyroscope and magnetometer sensors commonly available in Mobile Devices (MDs) [3].

Currently, two WiFi-based methods for localization exists, namely multilateration and fingerprinting [2], [4]. Multilateration estimates the position of MDs by computing the distances between Access Points (APs) and the reference MD comparing the strength of the signals (Received Signal Strength, RSS) transmitted by the APs and received by the MD. The main drawbacks of Multilateration are: i) low localization accuracy and ii) high energy consumption due to required heavy computation, which is a crucial issue when MDs are involved. FingerPrinting (FP) is a two-step procedure. The first step, carried out offline and also known as *training* phase, is the collection of information aimed at obtaining a spatio-temporal representation of the RSSs in a given area. The training is based on Reference Points (RPs), usually selected to cover the entire area of interest through a uniform grid, and on RSS measurements carried out by an MD and collected for each RP. The set of measured RSSs at a given RP is used to build the fingerprint for that RP.

Mobile cloud computing augments performance of mobile devices fostering task offloading [5]. Offloading stands for the capability of moving the load from one device to another [6] and can refer to both traffic offloading [7] or computation offloading [8]. In the second case, the tasks offloaded are executed remotely in the cloud or in the devices in the vicinity [9]. Offloading reduces processing and energy consumption of the mobile devices and is a valid strategy not only for energy saving, but can also for expanding data storage and extending computing capacity. This is especially useful for wearable devices [8]. Recently, fog computing has emerged as a new promising paradigm [10]. Fog computing extends mobile cloud computing capabilities towards the edge of the network and enables localization-aware services with low and predictable latency for geo-distributed applications. It is worth pointing out that, since this paper only presents preliminary results, we do not consider the impact of the obstacles and the lost of connection they could provide.

In this paper we improve and extend the *Smart* P-FP algorithm to be run in a distributed manner over mobile devices in a fog computing platform. For navigation, devices with low remaining battery charge offload to other devices in the vicinity the computation of the *Smart* P-FP algorithm. Being closer to the device initiating the process, the energy spent for data transmission is low. As a result, indoor navigation over fog computing platform is projected to become a sustainable and effective solution. Performance evaluation conducted with simulation highlights effectiveness of the proposed solution. Indeed, offloading becomes beneficial when two or more devices are in the vicinity and can contribute to relieve the computing load of the offloader. As reported in Section IV, power savings can be as high as 80% if compared with power spent with local computation.

II. Background and Motivations

This section provides a background on the fingerprint-based indoor positioning algorithms, with a particular emphasis on

the recently proposed version [11], [12] and on fog computing. Probabilistic-FingerPrinting (P-FP) positioning consists of two steps. The first step, also known as *training* is carried out offline and collects information to obtain a spatio-temporal representation of the RSSs from WiFi Access Points (APs) in a given area. The training is based on L Reference Points (RPs), uniformly distributed as a grid to cover the entire area of interest, and on the RSS measurements carried out by Mobile Devices (MDs) and collected for each RP. The set of measured RSSs, at a given RP, is used to build the fingerprint (FP) for such RP. More in detail, the mean value $\mu_{m,l}$ and the variance $\sigma^2_{m,l}$ of the m-th AP sensed at the l-th RP, $m \in [1, M]$ and $l \in [1, L]$, are computed and stored. The second step is called *positioning* and is performed online. The MD willing to locate itself, collects a RSS measure, the so called *observation vector* $\mathbf{o}$, which is compared with the reference FP of each RP. The estimated position corresponds to the coordinates, or to a combination of the coordinates, of the RPs whose fingerprints are the closest match with the RSS measures. Most positioning schemes base their action on the computation of the probability $p(\mathbf{o}|l)$ to have an observation vector $\mathbf{o}$ at a fixed RP l based on Gaussian probabilities. P-FP is of them: it computes a Gaussian-based probability $p(\mathbf{o}|l)$ entirely during the online phase [11]. It is identified as *Traditional* P-FP.

This paper exploits a recently proposed computation scheme for determining $p(\mathbf{o}|l)$, reported in [12]. Reducing the number of operations saves energy speeds up positioning. Such a scheme, called *Smart* P-FP advances *Traditional* P-FP in the way $p(\mathbf{o}|l)$ is computed and was tested over off-the-shelf Android OS smartphones [11], [12]. To reduce the number of operations, computation time and energy costs, *Smart* P-FP employs an algebraic factorization of the equations of the P-FP method. This allows to compute and store some necessary variables directly in the training phase, thus avoiding their computation during the online positioning phase.

Smart P-FP is a lighter algorithm with respect to its traditional version, still it could be computationally not negligible, especially when it is implemented over MDs that have limited resources.

When MDs exploit D2D communications and share computational load, they can save energy. Specifically, smartphone context-awareness through accelerometer and microphone readings is proved to guarantee considerable energy savings when the computation is shared by 3 or more devices [13].

Cisco proposed the *fog computing* concept to extend the traditional cloud computing paradigm at the edge of the network [10]. For such a reason, fog computing is also called *edge computing*. Fog computing is tailored to serve applications that are geo-distributed and require low latency and context awareness. As a result, it is envisaged to play an essential role in the framework of Internet of Things (IoT) [14]. Several research efforts are undergoing to assess sustainability and improve resource management of fog platforms for IoT applications [15], [16], [17], [18]. Fig. 1 illustrates a generic fog computing architecture. In the front-end, the mobile devices are IoT devices with various degree of computing, storage and

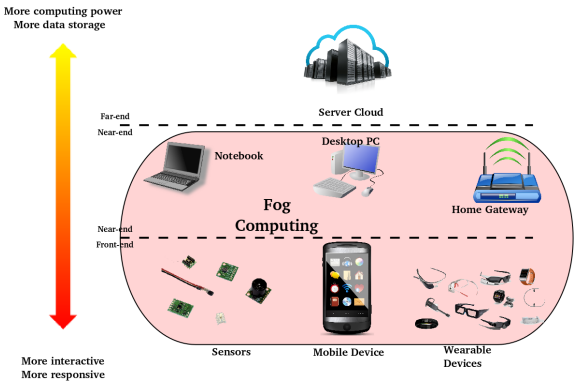


Figure 1. Fog computing architecture

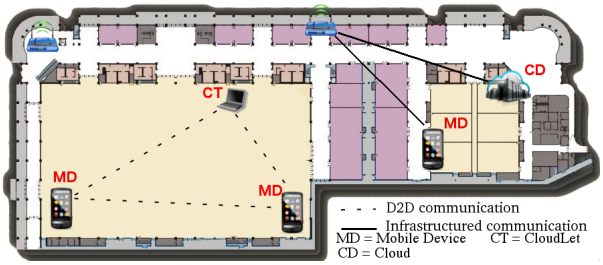


Figure 2. An example of a possible reference scenario

networking capabilities. CloudleTs (CTs) are typically local processing units such as notebook or desktop pcs used for temporary storage and processing [19]. They can also aggregate data to be delivered to the cloud in the far end, which provides centralization of functionalities and backup.

III. Indoor Localization meets Fog Computing

Along a day, many people visit public areas. As a result, exploiting fog computing paradigm and leveraging computational capacity of devices in the vicinity is an excellent solution to obtain localization while limiting battery drain at the same time. Fig. 2 illustrates an example of a possible reference scenario in which the architecture in Fig. 1 is employed. Users run the *Smart* P-FP algorithm introduced in Section II and detailed below. When their remaining battery charge is below a given threshold, they can demand the computation of the algorithm to other devices in the vicinity. For communications, Device-2-Device (D2D) technologies such as WiFi Direct can be employed. If no devices in the vicinity can help, the user performs computation offloading to the closest cloudlet or to the cloud. As the numerical results will show, merging a Smart P-FP approach with a fog computing platform allows to further save energy so extending the smartphone's battery lifetime.

A. Smart P-FP Algorithm

As said in Section II, most positioning schemes base their action on the computation of the probability $p(\mathbf{o}|l)$ to have an observation vector $\mathbf{o}$ at a fixed RP l based on Gaussian probabilities. The choice Gaussian PDF to model the RSSI in a fixed RP is supported by the natural symmetry of the

signal itself, as reported also in [2], [11] and references therein. *Traditional* P-FP computes the $p(\mathbf{o}|l)$ as follows:

$$p(\mathbf{o}|l) = \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi\sigma_{m,l}^2}} \mathrm{e}^{-\frac{\left(o_m - \mu_{m,l}\right)^2}{2\sigma_{m,l}^2}}, \forall l \in [1, L]\,. \quad (1)$$

Smart P-FP main idea is to re-structure the process required to obtain the $p(\mathbf{o}|l)$ value so to allow the computation and the storage of some quantities already in the training phase and therefore provide time and energy saving. It is based on some algebraic steps (not reported here for sake of brevity) which lead to introduce three new quantities:

$$\begin{cases} \alpha_l = \dfrac{\gamma_l}{2} \cdot \displaystyle\sum_{m=1}^{M} \frac{\mu_{m,l}^2}{\sigma_{m,l}^2}; \\ \beta_l = \left(\dfrac{1}{\sqrt{2\pi}}\right)^M \cdot \dfrac{1}{\sqrt{\frac{\gamma_l}{2}}}; \\ \gamma_l = \displaystyle\prod_{m=1}^{M} \sigma_{m,l}^2, \end{cases} \quad (2)$$

where α_l, β_l and γ_l are the elements of the following three vectors of length L: $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_l, \cdots, \alpha_L]$, $\boldsymbol{\beta} = [\beta_1, \cdots, \beta_l, \cdots, \beta_L]$ and $\boldsymbol{\gamma} = [\gamma_1, \cdots, \gamma_l, \cdots, \gamma_L]$, respectively. Since $\mu_{m,l}^2$ and $\sigma_{m,l}^2$ are already known after the *training* phase, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ can be computed without the knowledge of the observation vector $\mathbf{o}$. Exploiting the quantities reported in (2), the computation of $p(\mathbf{o}|l)$ with the *Traditional* algorithm can be re-written as follows:

$$p(\mathbf{o}|l) = \beta_l \mathrm{e}^{-\sum_{m=1}^{M} \frac{\mu_{m,l}^2 + o_m(o_m - 2\mu_{m,l}^2)}{2\sigma_{m,l}^2}}, \quad (3)$$

where the parameters $\mu_{m,l}^2$, $2\mu_{m,l}$, $2\sigma_{m,l}^2$ and β_l can be pre-computed and stored in the *training* phase. The computation in (3) is called *Smart*. It is important noticing that the result reported in (3) does not introduce any approximation at all.

B. Smart P-FP Algorithm over Fog Computing Platform

Algorithm 1 Offloading algorithm over fog computing platform

```
Require: struct[] D
Require: float[] J
Require: float[] P
  b =getBatteryLevel();
  if (b < β) then
    D = getDevicesInfo();
    if (D.length =≥ 3) then
      J = computeCostFunctions(D);
      for (i = 0; i < J.length; i++;) do
        P(i) = computeOffloadingPercentage(J(i));
      end for
      offloadData(P);
    end if
  else
    computePositionLocally();
  end if
```

Algorithm 1 details the steps required to distribute the computation load over the fog. The procedure requires four steps, namely *i) get devices info*, *ii) compute the cost function*, *iii) compute offloading percentages* and *iv) offload to the selected devices*. For localization, every MD with remaining battery charge below a threshold δ acquires information of devices in the vicinity related to their *cost function*. The authors of [13] have found that when enough devices are employed, offloading data over the fog is effective in terms of saved power. Starting from this result, we find out thanks to an analytical approach described in Section IV-A, that when at least 2 devices are available the fog approach is able to save power with respect to local computation. On the other hand, having less than 2 devices in the vicinity translates in offloading the computation to the CT.

Table I
INFORMATION EACH MD SHARES WITH THE FOG

	Battery Level	**Computation Load**	**Communication Interface**
Device 1	0.8	0.9	1
Device 2	0.2	0.3	1
Device 3	0.3	0.8	2
...	...	...	...
Device N	0.8	0.9	1

Get Devices info: When a MD has low battery level, *i.e.,* below a given threshold δ, it scans the surroundings, looking for other devices and demand computation of *Smart* FP algorithm. The devices within the fog emit beacon packets periodically, called metric vectors, to inform other devices in the vicinity about their current status of battery level, CPU load level and communication interface. The first two parameters assume real values in the range $[0, 1]$ where 1 denotes high availability of battery and CPU. The last parameter is an integer number defining the interface the MD exploits, namely 1 corresponds to BlueTooth (BT) and 2 corresponds to WiFi Direct. Table I contains some examples of the overall information contained within the metric vector.

Compute the cost function: Having information from devices in the vicinity, a MD can determine the *cost function*. Devices with high offloading potential are the best candidates contacted by the MD for offloading. Let T be the total number of MDs available and let $\mathbf{m_t}$ be the metrics vector for the t-th device, $t \in [1, T]$, containing all the information. Its single element $m_{t,i}$ with $i \in [1, 3]$ is the i-th metric for the t-th device (*e.g.,* $m_{4,1}$ is the battery level of the fourth founded device). Similarly, let $\mathbf{w} = [w_1, w_2, w_3]$ be the weight vector where each element w_i is the weight associated to each metric m_i. The *cost function* $J(\mathbf{m}, \mathbf{w})$ is therefore defined as follows:

$$J(\mathbf{m}, \mathbf{w}) = \sum m_i \cdot w_i = \mathbf{m} \cdot \mathbf{w},$$
$$\text{subject to } \begin{cases} 0 \leq w_i \leq 1; \\ \|\mathbf{w}\| = 1. \end{cases} \quad (4)$$

Having computed the offloading potential, the selected devices are the 2 with highest score in the set of candidates. It is worth noticing that to speed-up the identification step, in the set of

Table II
A NUMERICAL EXAMPLE OF THE OFFLOADING ALGORITHM PARAMETERS. THE EMPLOYED VALUE ARE $\mathbf{w} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$, $\epsilon = 0.5$ AND $L = 156$.

		Device 1	Device 2	Device 3
Device Info	Battery Level	0.7	0.3	0.9
	Computation Load	0.9	0.2	0.9
	Communication Interface	1	1	2
Math Functions	$J(\mathbf{m}, \mathbf{w})$	0.87	0.5	1.27
	p_n (%)	41%	13%	46%
RP for offloading	RP_{start}	1	63	83
	RP_{end}	62	82	156

candidate devices only those with offloading potential higher than a threshold ϵ are considered, i.e., $J(\mathbf{m}, \mathbf{w}) \geq \epsilon$.

Compute offloading percentages: Having selected at least 2 devices for offloading, it becomes necessary to divide the computation. The offloading percentages are computed exploiting the weighted mean of the metrics related to the battery level and the CPU availability. Being $N \leq T$ the number of devices for offloading, the corresponding offloading percentage p_n of the n-th device is determined as follows:

$$p_n = \frac{m_{n,1} \cdot w_1 + m_{n,2} \cdot w_2}{\sum_{n=1}^{N} m_{n,1} \cdot w_1 + m_{n,2} \cdot w_2} \cdot 100. \quad (5)$$

Offload to the selected devices: When determined the percentage of computation each selected device will perform, the last step practically distributes the load. Exploiting the *Smart*-FP $p(\mathbf{o}|l)$ formula in (3), we proposed to divide the overall load by assigning the number of Reference Points (RPs) that it must cover in computing $p(\mathbf{o}|l)$. According to the offloading percentages p_n and the total number of considered RPs L, we define r_n as the amount of RPs that the n-th selected device for offloading must cover. This parameter is defined as follows:

$$r_n = \lfloor\, p_n \cdot L \,\rfloor\,. \quad (6)$$

Let RP^i_{start} and RP^i_{end} the first and the last RP that the i-th device must consider for $p(\mathbf{o}|l)$ computation. The generic MD i computes the value of (3) starting from the RP RP^i_{start} and ending at the RP RP^i_{end}. In formula:

$$RP^i_{start} = \sum_{n=1}^{i} r_n; \quad RP^i_{end} = \sum_{n=1}^{i} r_n - r_i + 1. \quad (7)$$

For the sake of clarity, a complete numerical example is reported in Table II with parameters $\mathbf{w} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$, $\epsilon = 0.5$ and $L = 156$. The example shows the case where 3 MDs are selected for offloading. Device 1 has an offloading percentage of 41%, Device 2 of 13% and Device 3 equal to 46%. These three percentages correspond to the specific amount of RP that each of them must manage. Device 1 will compute the $p(\mathbf{o}|l)$ from the first RP to the 62-th. Device 2 will take care of RPs from the 63-th to the 82-th and Device 3 will manage RP from 83 to 156. When all the three devices have finished to compute their $p(\mathbf{o}|l)$, they send back their results to the device initiating the process. Such device can infer its location exploiting one

Table III
SIMULATION SETUP

PARAMETER	VALUE
Number of considered RPs	156
Number of considered APs	From 1 to 100
Number of simulation runs	10^6
P_{TX}	0.5 [mW]
P_P	0.1 [mW]
B_{RX}	250 [MBp]
D_{RX}	0.5 [Kbyte]
Indoor Environment Size	30 m × 30 m
Number of Users	200
User Arrival Rate	$\{2, 3, ..., 8\}$ user per minute
Threshold β	0.3
Threshold ϵ	$\{0.1, 0.2, ..., 1\}$

of the following methods:

NN approach: Denoting with $\mathbf{C}_l = (x_l, y_l)$ the coordinates of the l-th RP and with $\mathcal{L}$ the set of the RP indexes with cardinality $|\mathcal{L}| = L$, the NN approach estimates the MD position as follows:

$$\tilde{\mathbf{C}} = (\tilde{x}, \tilde{y}) = (x_{l_{max}}, y_{l_{max}})\,, \quad (8)$$

where:

$$(x_{l_{max}}, y_{l_{max}}) \text{ s.t. } l_{max} : \underset{l \in \mathcal{L}}{\operatorname{argmax}} \{p(\mathbf{o}|l)\}\,. \quad (9)$$

K-NN Approach: K-NN approach estimates the MD coordinates as:

$$\tilde{\mathbf{C}} = (\tilde{x}, \tilde{y}) \text{ s.t. } \tilde{x} = \frac{1}{K}\sum_{j=1}^{K} x_j \text{ and } \tilde{y} = \frac{1}{K}\sum_{j=1}^{K} y_j, \quad (10)$$

where:

$$(x_j, y_j) \text{ s.t. } j : \underset{l \in \mathcal{L}^k}{\operatorname{argmax}} \{p(\mathbf{o}|l)\}\,. \quad (11)$$

The set $\mathcal{L}^k \subset \mathcal{L}$ is defined as:

$$\mathcal{L}^k = \left\{\mathcal{L}^{k-1} \setminus \left\{l^{k-1}_{max}\right\} \text{ and } \mathcal{L}^1 = \mathcal{L}, \right. \\ \left. l^1_{max} = l_{max}\right\}, \forall k \in [1, K]\,. \quad (12)$$

L^k cardinality is $\left|\mathcal{L}^k\right| = L - k$.

KW-NN Approach: The KW-NN method requires the computation of the vector $\mathbf{\Phi}$ whose elements are the inverse of the probabilities $p(\mathbf{o}|l),\ \forall l \in [1, L]$.

$$\mathbf{\Phi} = \left[\frac{1}{p(\mathbf{o}|1)}, \frac{1}{p(\mathbf{o}|2)}, \cdots, \frac{1}{p(\mathbf{o}|l)}, \cdots, \frac{1}{p(\mathbf{o}|L)}\right]. \quad (13)$$

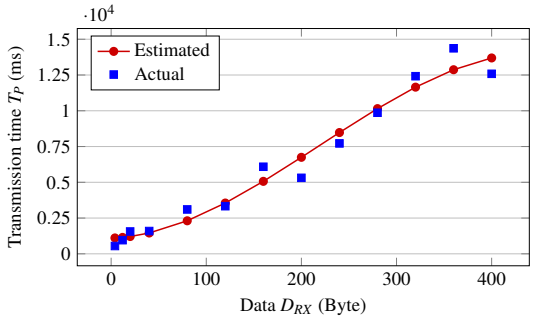
The KW-NN approach corresponds to the K-NN approach, but the average of the coordinates is weighted with the K correspondent weights of the vector $\bar{\mathbf{\Phi}}$.

$$\tilde{x} = \frac{\sum_{j=1}^{K} \phi_j x_j}{\sum_{j=1}^{K} \phi_j} \text{ and } \tilde{y} = \frac{\sum_{j=1}^{K} \phi_j y_j}{\sum_{j=1}^{K} \phi_j}, \quad (14)$$

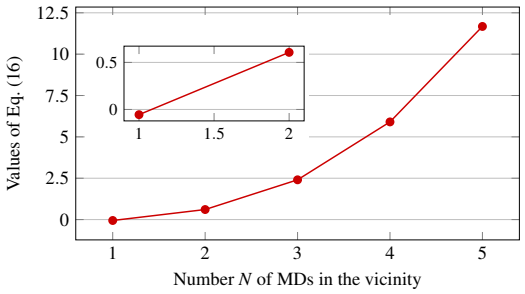
where the quantity ϕ_j represents the j-th component of the vector $\mathbf{\Phi}$.

IV. PERFORMANCE EVALUATION

To evaluate the performance of the proposed fog computing approach for indoor localization, we propose two analysis. The first study verifies energy gains the fog enables. The second study analyzes effectiveness of the policy used to



(a)



(b)

Figure 3. Benefit of fog computing. Analysis of transmission time vs amount of data to be offloaded (a) and power saving with increasing number of devices in the vicinity (b).

select candidates devices for computation offloading. Table III presents the details of the simulation setup.

A. Analysis of Energy Consumption

Exploiting the mathematical model proposed in [13], it is possible to compare the power spent to determine the position locally or through offloading. First, using the Least-Square (LS) approximation method, we estimate the time necessary for the processing as a function of the amount of data to be transferred during offloading. To achieve a robust estimation, we have performed 10^6 positioning runs. In Fig. 3(a) the squares are the actual values and the continuous line over the rounds is the obtained estimation of the offloaded data D_{RX} against the processing time T_P. To obtain an exploitable mathematical relationship between time and data, an LS third-degree polynomial interpolation is employed. The coefficients are reported in the following formula:

$$T_P = -2.91 \cdot 10^{-8} \cdot D_{RX}^3 + 0.191 \cdot 10^{-4} \cdot D_{RX}^2 + \\ + 1.70 \cdot 10^{-4} \cdot D_{RX} + 0.11 \quad (15)$$

Using the analytical model proposed in [13], we compare the power spent when the computation is performed locally or distributed over the fog platform. Let N be the number of MDs in the fog, P_{TX} and P_P the power (in W) spent for transmission and processing, respectively while B_{RX} is the transmission rate. From an energy point of view, it is convenient to offload computation if the inequality (16) holds. Note that the term T_P from (15) was included in addition to the basic model proposed in [13]. Table III lists the values of all parameter used. Equation (16) was solved numerically for several values of N. Fig. 3(b) plots the results obtained. As it is possible to

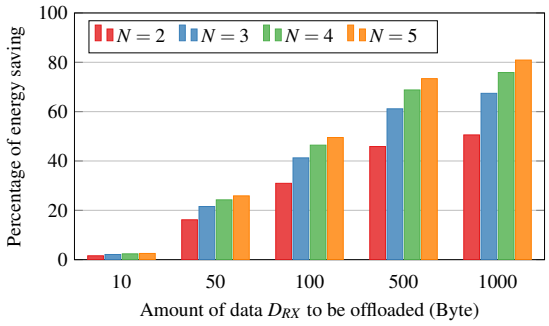


Figure 4. Percentage of power saved

see from the graph and from the related enlargement, having more than one fog device contributing to computation becomes beneficial of the offloader. This confirms the effectiveness of the proposed method: demanding the computation of the user's position within the fog environment is always convenient with at least 2 MDs are available.

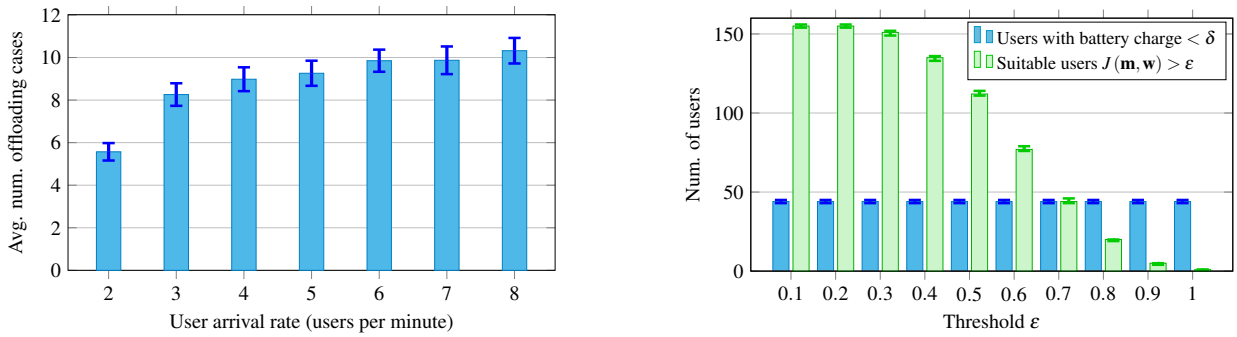
$$N^2(N-1)\left[P_{RX} \cdot \frac{D_{RX}}{B_{RX}}\right] + P_P \cdot D_{RX} \\ \left[1.3 \cdot 10^{-6}(N^3-1) - 3.24 \cdot 10^{-4} N(N^2-1) + \right. \\ \left. 0.38N^2(N-1)\right] > 0 \quad (16)$$

Fig. 4 analyzes the amount of power performing offloading saves in comparison with local computation. The graph plots the savings, in percentage, with the increase of N and for different amounts of data to be offloaded. The savings increase with the the amount of data offloaded and N. Having $D_{RX} > 500$ bytes, the power saved performed offloading is in the order of 50% with respect to perform *Smart* P-FP locally. When 5 other devices are available and $D_{RX} > 1000$ bytes, the power savings are above 80%.

B. Analysis of Candidate Device Selection Process

When the remaining battery charge of a MD is below the threshold δ, computation offloading takes place. For the experiment, δ is set to 0.3. The number of users is 200 located in an indoor environment of $30\text{m} \times 30\text{m}$. Users arrive following a Poisson distribution with mean $\{2-8\}$ users per minute. Each user holds an MD only, whose remaining battery charge and computing capacity are set at the beginning of the evaluation period uniformly between $[0, 1]$.

In the first experiment, we analyze the impact of user arrival rate for successful offloading. For the experiment, the threshold ϵ is set equal to 0.5. For each user with battery charge below δ, we count the number of successful offloading cases. Successful offloading occurs when 2 or more users in the vicinity have an offloading potential higher than ϵ and have arrived at most 1 minute before the reference user. The latter hypothesis allows to consider a worst case scenario with few other devices available. Fig. 5(a) shows that the average number of successful offloading increases with the increase of the user arrival rate. The error bars represent the confidence interval with 95% confidence level. The result is expected as having more users at disposal augments the chances to find at least 2 suitable devices for offloading.



(a) (b)

Figure 5. Analysis of device selection. Successful offloading with increasing user arrival rate (a) and threshold ϵ (b).

In the second experiment, we analyze the impact of the threshold ϵ for candidate selection. For the experiment, the user arrival rate is fixed to 2 users per minute. Similarly to the previous study, error bars represent the confidence interval with 95% confidence level. The number of users with low remaining battery charge remains constant during the evaluation period. Indeed, this parameter does not depend on ϵ. The number of potential candidates for offloading decreases with the increase of the threshold ϵ. Indeed, high values of ϵ make suitable for selection only the devices with high remaining battery charge and computing capacity. It is worth noting that values of $\epsilon < 0.7$ fully satisfy the demand for offloading of devices having battery charge below δ.

V. Conclusions

In this paper, we propose to distribute computation of the *Smart* P-FP algorithm over a fog. Devices with low remaining charge of battery exploit the computing capacity of other devices in the vicinity by offloading computation of the *Smart* P-FP algorithm. The results highlight the effectiveness of the proposal in two directions, namely energy and candidate selection process efficiency. With 2 or more devices, offloading is beneficial from a energy standpoint and the amount of energy saved can be as high as 80% if compared with local computation.

Acknowledgment

C. Fiandrino, D. Kliazovich and P. Bouvry would like to acknowledge the funding from National Research Fund, Luxembourg in the framework of ECO-CLOUD and iShOP projects.

References

[1] Ericsson ConsumerLab, "The indoor influence," *Consumer Insight Summary Report*, April 2015.

[2] S. He and S. H. G. Chan, "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 466–490, Firstquarter 2016.

[3] Y. Zhuang, Z. Syed, J. Georgy, and N. El-Sheimy, "Autonomous smartphone-based WiFi positioning system by using access points localization and crowdsourcing," *Pervasive and Mobile Computing*, vol. 18, pp. 118 – 136, 2015.

[4] C. Yang and H. r. Shao, "Wifi-based indoor positioning," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 150–157, March 2015.

[5] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 – 106, 2013.

[6] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 3, pp. 384–398, Sept 2015.

[7] C. Fiandrino, D. Kliazovich, P. Bouvry, and A. Y. Zomaya, "Network-assisted offloading for mobile cloud applications," in *IEEE International Conference on Communications (ICC)*, June 2015, pp. 5833–5838.

[8] C. Ragona, F. Granelli, C. Fiandrino, D. Kliazovich, and P. Bouvry, "Energy-efficient computation offloading for wearable devices and smartphones in mobile cloud computing," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[9] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *IEEE 8th International Conference on Cloud Computing (CLOUD)*, June 2015, pp. 9–16.

[10] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 169–186.

[11] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "Energy efficient WiFi-based fingerprinting for indoor positioning with smartphones," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 4639–4643.

[12] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "Smart probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices," *Pervasive and Mobile Computing*, pp. –, 2016.

[13] I. Bisio, F. Lavagetto, A. Sciarrone, T. Penner, and M. Guirguis, "Context-awareness over transient cloud in D2D networks: Energy performance analysis and evaluation," *Transactions on Emerging Telecommunications Technologies*, 2015.

[14] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. ACM, 2012, pp. 13–16.

[15] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[16] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for iot," in *IEEE 29th International Conference on Advanced Information Networking and Applications (AINA)*, March 2015, pp. 687–694.

[17] R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martínez, J. Serra, C. Verikoukis, and R. M. noz, "End-to-end sdn orchestration of iot services using an sdn/nfv-enabled edge node," in *Optical Fiber Communication Conference*. Optical Society of America, 2016, p. W2A.42. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2016-W2A.42

[18] V. Miliotis, L. Alonso, and C. Verikoukis, "Offloading with ifom: The uplink case," in *2014 IEEE Global Communications Conference*, Dec 2014, pp. 2661–2666.

[19] M. Chen, Y. Hao, Y. Li, C. F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: architecture and service modes," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 18–24, June 2015.