A user-centric mobility management scheme for high-density fog computing deployments

Zeineb Rejiba*, Xavier Masip-Bruin*, Eva Marín-Tordera*
*Advanced Network Architectures Lab (CRAAX), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{zeinebr, xmasip, eva}@ac.upc.edu

***Abstract*—The inherent mobility characterizing users in fog computing environments along with the limited wireless range of their serving fog nodes (FNs) drives the need for designing efficient mobility management (MM) mechanisms. This ensures that users' resource-intensive tasks are always served by the most suitable FNs in their vicinity. However, since MM decision-making requires control information which is difficult to predict accurately a-priori, such as the users' mobility patterns and the dynamics of the FNs, researchers have started to shift their attention towards MM solutions based on online learning. Motivated by this approach, in this paper, we consider a bandit learning model to address the mobility-induced FN selection problem, with a particular focus on scenarios with a high FN density. Following this approach, a software agent implemented within the user's device learns the FNs' delay performances via trial and error, by sending them the user's computation tasks and observing the perceived delay, with the goal of minimizing the accumulated delay. This task is particularly challenging when considering a high FN density, since the number of unknown FNs that need to be explored is high, while the time that can be spent on learning their performances is limited, given the user's mobility. Therefore, to address this issue, we propose to limit the number of explorations to a small subset of the FNs. As a result, the user can still have time to be served by the FN that was found to yield the lowest delay performance. Using real world mobility traces and task generation patterns, we found that it pays off to limit the number of explorations in high FN density scenarios. This is shown through significant improvements in the cumulative regret as well as the instantaneous delay, compared to the case where all newly-appeared FNs are explored.**

***Index Terms*—Fog computing, Edge computing, Fog Node selection, mobility management, multi-armed bandits**

I. Introduction

In the recent years, the landscape of consumers' applications has witnessed an increased popularity of a new class of applications, including Internet of Things (IoT) applications, augmented reality (AR) and virtual reality (VR), to name a few. An important characteristic of such applications is their strict Quality of Service (QoS) requirements, which cannot be successfully met when using the conventional cloud-based provisioning model. Moreover, these applications usually generate massive amounts of data that need to be sent over the core network for further processing in the cloud, which adds a significant burden on the underlying infrastructure.

Therefore, in order to cope with the afore-mentioned limitations, new computing paradigms have been proposed, namely fog [1] and edge [2] computing. These computing paradigms rely on the use of spare computational resources provided by distributed nodes at the edge of the network, generally referred to as fog nodes (FNs)[1] (or MEC servers in the multi-access edge computing terminology). Leveraging the ubiquity of such nodes and the ability of the users to access them using a one-hop wireless access, the required QoS levels mandated by highly-interactive applications can be met.

However, users of such applications could be on the move, and as a result, the fog nodes that serve the tasks generated by their applications should be appropriately selected to cope with their continuously-changing locations. That is why, mobility management (MM) is seen as one of the main research problems that need to be addressed in the fog computing context.

Many contributions have been recently proposed to address this problem. More specifically, the relevant literature reveals a noticeable trend towards adopting user-centric approaches that rely on embedding smartness and decision-making capabilities onto the user's device side (as envisioned in [3]). In line with this, the trend towards using online MM mechanisms (i.e. having no prior knowledge about the system parameters) is also gaining in popularity. Such a trend is motivated by the need to cope with the uncertainty characterizing the users' mobility, the FNs' availabilities and their capabilities. In fact, continuously requesting such information whenever a MM decision needs to be made would lead to a high control overhead. That is why, learning via *trial and error* was instead envisioned to support the decision making, at the cost of incurring occasional losses due to the learning process.

Even though the works that have been proposed in this context are promising, they are generally characterized with the following limitations. First, they consider a limited user mobility ([4], [5], [6]) and as such, they do not address the problems that may arise from longer mobility durations, such as the continuously-changing set of FNs. Second, they consider scenarios with a limited number of FNs, where the user can have enough time to learn the best one among them ([7], [8]), which may not always be the case. In fact, as will be shown in our collected data, a user usually has to learn the performances of a high number of FNs in a very short time frame, since the set of nearby FNs changes due to mobility.

Therefore, with such observations in mind, the aim of this work is to design a fog node selection scheme in a high-density fog deployment, while taking the effects of the user's mobility into account. To this end, we adopt a learning approach

[1]We will adopt the term Fog Node for the remainder of the paper.

inspired by the *many-armed bandit* learning model[9], where the number of FNs approaches the number of trials, for a fixed FN availability duration. This distinguishes our work from the rest of the literature that considers standard bandit learning with a number of trials that is much greater than the number of FNs. Then, in order to avoid the time-consuming exploration of the delay performances of the whole set of the newly-appeared FNs, we reduce the number of exploration candidates to a subset of FNs and then proceed to the exploitation of the FN offering the lowest delay. To evaluate this approach, we derive the task generation rate for a typical edge/fog application and use it along with a realistic mobility trace. The obtained results show significant reductions in the cumulative regret of the learning algorithm and the instantaneous delay perceived by the user.

The remainder of this paper is organized as follows: Section II provides an overview of the related works. Section III describes the considered system model and the research problem that we intend to address. In section IV, we present the details of our proposed solution. We then highlight the obtained results and discuss some implications of the proposed approach in sections V and VI, respectively. Finally, we conclude the paper.

II. Related work

Given the growing interest shown by the research community to mobility management in edge and fog computing scenarios, several contributions have been proposed to address this problem.

Some of these contributions are *network-centric*, i.e. the decision of selecting the target FN is implemented within a specific entity in the network. For instance, authors in [10] propose a task offloading scheme for ultra-dense edge computing environments with the goal of reducing the task duration, while keeping the device's energy consumption low. The proposed scheme is implemented within a controller attached to a macro base station having global information about the system. Similarly, authors in [11] propose a mobility-aware offloading algorithm which is envisioned to be implemented within the SDN controller. The algorithm decides the offloading size as well as the communication path for the offloaded task, using predicted near-future information about tasks and network conditions. Other works in this category have been extensively reviewed in [12].

In contrast to these network-centric approaches, there has been a noticeable trend towards using *user-centric* decision-making, where users' devices can host the "smartness" allowing them to perform advanced decision-making. Such a trend is coupled with using online learning approaches, since information about the considered system dynamics cannot be predicted accurately a-priori.

One relevant work in this direction is [4], where authors address the problem of selecting the optimal BS to serve a user's task in an ultra-dense MEC deployment, such that the delay and the device's energy consumption are minimized. An online solution based on the multi-armed bandit (MAB) theory is presented to solve the problem. In particular, a Volatile MAB approach is adopted to deal with the volatility of BSs. However, authors consider that the number of learning trials is large enough compared to the number of BSs whose performances need to be learnt. Focusing on a vehicular edge computing scenario instead, authors in [7] consider the problem of task offloading from one vehicle to neighboring serving vehicles (SeVs) with the goal of minimizing the overall delay. Similar to [4], a volatile MAB approach is used to deal with the SeVs' volatility, in addition to including load-awareness in the selection decision process. Subsequent work [8] considers task replication, where the same task can be processed simultaneously by multiple vehicles to improve reliability. In [13], authors consider a fog network where a task node needs to offload tasks to a helper node such that the long-term latency is minimized. Authors focused on the problem of the non-stationarity affecting the system parameters, and as such mobility-related aspects were not addressed.

Instead of relying on the MAB learning model, authors in [5] propose to use Q-learning for addressing mobility-management in a dense MEC environment. In the proposed solution, the user observes the current state, i.e. the serving BS and the channel conditions and decides on the target BS based on previously-observed task execution speeds. However, evaluation results have been carried out considering a small number of BSs and a limited mobility area. In [6], authors propose a user-centric computation offloading scheme for virtual edge computing systems based on double Q-learning. More specifically, offloading decisions are made based on the task queue state, the energy queue state and the channel quality, but with no specific focus on user mobility.

III. System model and problem formulation

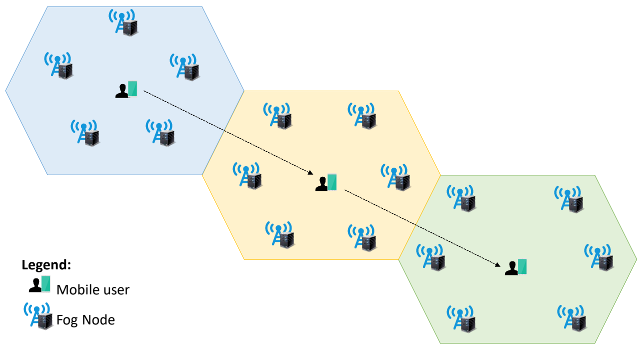


Fig. 1. System model

As shown in Fig. 1, we consider a representative user moving within a high-density fog computing deployment comprised of a set $\mathcal{A}$ of fog nodes. From the user's perspective, the set of visible FNs is time-varying due to the mobility. Therefore, if we denote by T_c the period characterizing the change of the visible FNs, the sets of FNs seen by the user can be represented as $\mathcal{A}_{T_c}, \mathcal{A}_{2T_c}, \ldots, \mathcal{A}_{nT_c}$. Naturally, given the limited range of the FNs, the transition from $\mathcal{A}_{iT_c}$ to $\mathcal{A}_{(i+1)T_c}$ can result in new FNs appearing for the first time, some FNs disappearing and others remaining available for two consecutive periods.

The considered user runs an application that continuously generates a sequence of computation-intensive tasks that need to be served by the encountered FNs (e.g. a location-based mobile AR game). We denote by M the number of tasks that the application generates within T_c.

An intelligent software agent is implemented within the user's device (or as part of the application) to select the most suitable FNs that should serve the user's tasks. In order to assess the quality of a given selection, we consider the total delay perceived by the user as a performance indicator. Such a delay is comprised of multiple components, as we detail next:

- The *transmission delay*, i.e. the delay for transmitting the task input data to FN a over the wireless channel. It can be calculated as follows:

$$d_{tr}(t,a) = \frac{N}{R_{t,a}} \tag{1}$$

  where N is the task's input data size and $R_{t,a}$ is the transmission rate.
  Given the channel bandwidth W, the transmit power P_{TX}, the channel gain to FN a $H_{t,a}$, the noise power σ^2 and the interference level $IF_{t,a}$, the transmission rate can be derived as follows:

$$R_{t,a} = Wlog_2\left(1 + \frac{P_{TX}H_{t,a}}{\sigma^2 + IF_{t,a}}\right) \tag{2}$$

- The *computation delay* is the delay taken by a FN to process a task with an input size I and a computational intensity N. It can be determined as follows:

$$d_{cmp}(t,a) = \frac{N * I}{f_{t,a}} \tag{3}$$

  where $f_{t,a}$ is the available computation capacity at FN a at time t.
- The *switching delay*. It refers to the delay incurred when the newly-selected FN is different from the previous one. It can be seen as the time that the newly-selected FN takes to start a container with the user's service[2].

Considering the aforementioned delay components, the overall delay observed by a user served by FN a at time t can be expressed as[3]:

$$d_{tot}(t,a) = d_{tr}(t,a) + d_{cmp}(t,a) + d_{swi}(t,a) \tag{4}$$

Our objective is then to find the optimal sequence of FNs that should serve the user's tasks in order to minimize the average delay over the total service duration T:

$$\min_{a\in\mathcal{A}} \frac{1}{T}\sum_{t=1}^{T} d_{tot}(t,a) \tag{5}$$

Since solving this problem requires having prior knowledge about time-varying information such as the FNs' availabilities and their expected delay performances, we address it using an online learning approach based on the MAB theory, as detailed in the following section.

IV. Proposed approach: Limited exploration for bandit-based FN selection (LimExp)

In this section, we first review the foundations of the bandit theory, in order to provide a better understanding of our approach. Then, we explain how our considered scenario differs from the standard bandit model and the issues that may arise in the resulting setup. Finally, we present the solution that we propose to address these issues.

A. MAB Foundations

In Multi-Armed Bandits, a learning *agent* is presented with multiple *actions* to choose from, each leading to receiving an initially-unknown *reward*. This process is similar to the way a gambler would have to choose among different slot machines in order to maximize its total reward. The agent follows a trial and error approach to learn the reward distributions of the different actions. Ideally, the agent's action selection strategy needs to balance *exploration*, i.e. selecting different actions to learn an accurate estimate of their rewards and *exploitation*, i.e. selecting actions having the best known reward so far. Upper Confidence Bound (UCB) [14] algorithms are commonly used in the literature to address this exploration-exploitation tradeoff. In UCB, each action a has an index i_a that combines its mean reward $\overline{r}(a)$ and a confidence term[4] that takes into account the current decision slot t and the number of times that action has been selected up to that decision slot $N_t(a)$:

$$i_a = \overline{r}(a) + \sqrt{\frac{log(\xi t)}{N_t(a)}} \tag{6}$$

The algorithm will then select the action having the highest index. The result is that actions having indices with low values and those which have been sufficiently selected in the past will be selected less often in the future.

In order to measure the efficiency of such bandit algorithms, the concept of the cumulative regret R_n, defined below, is generally used:

$$R_n = \sum_{i=1}^{n}(\mu_i^* - \mu_i) \tag{7}$$

where n is the n^{th} decision slot, whereas μ_i^* and μ_i denote the reward of the optimal action at the i^{th} decision slot and

[2]We assume that the container engine is already started at the new FN and that the considered service is popular in the considered area and as a result the container image is already downloaded. As a result, the delays that may result from these two processes are not considered.

[3]Similar to previous works[4], [5], we do not consider the downlink transmission delay in this paper.

[4]Also called padding function or exploration bonus.

the reward of the action selected at the i^{th} decision slot, respectively.

As it can be noted, the cumulative regret measures the difference between the agent's accumulated reward and the accumulated reward that could have been achieved if the optimal actions were selected, assuming their reward distributions were available a-priori.

When the MAB problem also involves a switching cost when the actions taken at two consecutive decision slots are different, the cumulative regret also includes the switching regret ([15], [4]), as follows:

$$R_n = \sum_{i=1}^{n}(\mu_i^* - \mu_i) + \left(C \sum_{i=2}^{n} \mathbb{1}\{a_i \neq a_{i-1}\} \right) \quad (8)$$

where C is the switching cost.

B. Beyond the standard MAB

Based on the above-defined bandit model, we define the following mappings between the different bandit components and our system components:

- Agent: The software agent implemented within the user's device in order to perform the FN selection decisions.
- Action: The FN to be selected to serve a user's task.
- Reward: The delay, which needs to be minimized.

Since we focus on the case of a user on the move, the set of visible FNs changes over time. This is usually referred to as a *sleeping*[16] or *volatile*[17] bandit problem in the literature. Such a problem is particularly challenging in our case, since the user has a very limited time to learn before new FNs become available and others disappear, thus disrupting the learning process.

The afore-mentioned volatile bandit model has been first used in the context of edge computing in [4]. In this work, authors used the concept of an *epoch*[5] as the duration in which the set of available MEC-enabled BSs remains unchanged. Within this epoch, the number of tasks that can be used for learning purposes is high compared to the number of BSs. This makes the exploration of each newly-appeared BS possible. However, our scenario is different in that we consider a high FN density, where the number of FNs may be greater than or equal to the number of tasks in a given epoch (as it can be seen in our collected data). In most cases, there will be a high number of new FNs that need to be explored in each epoch. Yet, if all such FNs are explored, there will be no trials left for exploitation. This situation can be approached to the *(infinitely) many-armed bandits*[9] extension of MABs, which is characterized by a large (or infinite) number of actions compared to the possible number of trials.

C. Proposed solution: Limited exploration for bandit-based FN selection (LimExp)

Inspired by the UCB-based solutions proposed for the infinitely many-armed bandits in [9], we propose to consider only a fixed number $\boldsymbol{K}$ of FNs for exploration and therefore avoid the exploration of all newly-appeared FNs in a given epoch. After those K FNs have been explored, the user can spend the remaining trials within the considered epoch connected to the FN having the best known delay performance. The rationale for our approach comes from the fact that, since we have a high number of FNs, we may end up with multiple FNs resulting in similar delay performances. Therefore, discarding some of them from the exploration process is not likely to degrade the performance. In addition, this results in a decreased probability of selecting an FN having a bad performance. Finally, limiting the number of explorations allows the user to avoid frequent switching from one FN to the other, thus ensuring a seamless task execution.

The proposed LimExp algorithm, depicted in Alg.1, is described next. For each epoch i, the agent retrieves the list of FNs in range $\mathcal{A}_i$. Then, it creates a list $\mathcal{B}$ of FNs which have not been explored yet. If the number of such FNs is lower than K, all of them are considered as exploration candidates. Otherwise, only a subset of $\mathcal{B}$ of size K is considered for exploration. Following this, and as long as there are tasks generated in the considered epoch, the agent will check whether the FN that will serve the current task should be selected via exploration or via exploitation. To this end, it checks the list of exploration candidates. If this list is empty, it will select the FN that resulted in the lowest delay in the previous trials, taking into account its corresponding confidence term (Line 15 in in Alg.1). Once the FN performs the task and sends the result back to the user, the agent will observe the delay $d_{tot}(a_{m,i})$ and update the accumulated delay for the chosen FN $z_{a_{m,i}}$[6] as well as the number of times it has been chosen up to the current decision slot $n_{a_{m,i}}$.

V. Evaluation results

In this section, we provide an overview of our data collection approach, which is followed by an analysis of the obtained results.

A. Data collection approach

1) Mobility data collection: In order to emulate the changing availabilities of the FNs from a mobile user's perspective, we used an Android mobile application to log the list of detected Wi-Fi access points as a user moves in the downtown area of the city of Vilanova I La Geltrù[7] where our lab is located. The user was walking for a duration of 30 minutes approximately. Such a duration may correspond to a common session length of a mobile augmented reality game such as Pokémon Go[18]. Within this duration, scans were performed every 30s, since as stated in the Android documentation[8], each foreground app is allowed to scan four times in a 2-minute period for Android 9 and later versions.

[5]We will also use this term to refer to the duration in which the set of FNs remains unchanged.

[6]Since the switching delay is not an indicator of the quality of a certain FN, it is subtracted from $d_{tot}(a_{m,i})$ in Line 18 of Alg.1.

[7]The city has a population of $\sim$ 66.000 residents.

[8]https://developer.android.com/guide/topics/connectivity/wifi-scan

Algorithm 1: LimExp

```
input: $K$: The number of FNs to explore
1  for $i \leftarrow 1$ to $n$ do
2      $A_i \leftarrow$ The set of FNs in range
3      $B \leftarrow get\_all\_unexplored(A_i)$
4      $exploration\_candidates \leftarrow \{\}$
5      if $|B| > 0$ then
6          if $|B| < K$ then
7              $exploration\_candidates \leftarrow B$
8          else
9              $exploration\_candidates \leftarrow sample(K, B)$
10         end
11     for $m \leftarrow 1$ to $M$ do
12         if $|exploration\_candidates| > 0$ then
13             $a_{m,i} = random(exploration\_candidates)$
               $exploration\_candidates \leftarrow$
               $exploration\_candidates \setminus \{FN_{selected}\}$
14         else
15             $a_{m,i} = \arg\min_{a \in A_i}(\frac{z_a}{n_a} - \sqrt{\frac{\xi log(m+M*(i-1))}{n_a}})$
16         end
17         Observe delay $d_{tot}(a_{m,i})$
18         $z_{a_{m,i}} \leftarrow z_{a_{m,i}} + (d_{tot}(a_{m,i}) - d_{swi}(a_{m,i}))$
19         $n_{a_{m,i}} \leftarrow n_{a_{m,i}} + 1$
20     end
21 end
```

After completing the data collection phase, we performed a basic data analysis to extract general statistics from the obtained trace. More specifically, we found that a single scan resulted in the detection of 33 APs on average, while the minimum number of APs per scan was 19 and the maximum was 55. Based on this trace, we defined two scenarios to emulate different FN densities, as described below:

- The first scenario, corresponds to a **medium FN density**, where all APs having SSIDs representing freely-accessible Wi-Fi networks (i.e. corresponding to public city facilities) are considered as having FN capabilities. This results in an average of 10 FNs per scan, where 3 FNs are new compared to the previous scan. The resulting distribution of FNs in this scenario is depicted in Fig. 2
- The second scenario, corresponds to a **high FN density** (i.e. the case that we focus on in this paper), where we consider all APs from the previous scenario in addition to 50% of the remaining ones. It is worth noting that the remaining APs have SSIDs indicating a private Wi-Fi network (generally starting with an internet service provider name and could correspond to corporate/home networks). So, for the purposes of this work, we consider that only half of them are upgraded by their owners to have FN capabilities. We found that in this case, 21 FNs are detected on average in each scan, 9 out of which are new compared to the previous scan. The corresponding distribution of FNs in this case is shown in Fig. 3.

2) Task generation rate: For the purposes of this work, we considered *object recognition* (OR) as a typical resource-intensive task that may require fog capabilities. In fact, object recognition is an essential component of mobile augmented reality applications, which are typical use cases for fog/edge computing[19].

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Task input size N	1 MBits
Task computation intensity I	$2640 * N$ [19]
Switching cost (e.g. container start-up time)	50ms[20]
Bandwidth W	20MHz
Transmit power P_{TX}	0.5 W
Channel gain $H_{t,a}$	$127 + 30log(d)$, where d is the user-FN distance[4]
Noise power σ^2	$2.10^{-13}W$
Computation capacity $f_{t,a}$	Uniformly distributed within $[1, 25]$ GHz [4]
T_c	$30s$ (in line with the scan interval in the mobility trace)
UCB constant ξ	8 [16]

In order to determine the rate at which such tasks would be generated, we used an Android-based Image labeling application provided by Firebase ML Kit[9]. This application continuously sends requests to annotate the contents of live video frames obtained from the smartphone camera. OR algorithms provided by Google's Cloud Vision API are invoked to serve the application's requests. The task generation rate can then be derived from the API's dashboard. In fact, we found that the *Image Annotation* method was invoked by the mobile application 29 times (on average) in a 30s interval. Therefore, we consider $M = 29$ to represent the number of tasks generated within a period $T_c = 30s$.

B. Obtained results

Using the collected mobility trace and the derived task generation pattern, simulations have been conducted in a custom simulator built in Python using the models defined in Section III and the parameters shown in Table I. We compare our algorithm to the Auer algorithm[16] which was originally proposed for sleeping bandits. As opposed to our algorithm, Auer performs explorations of all newly-appeared FNs. We also compare to an Oracle algorithm having exact prior knowledge about the FNs' availabilities and their corresponding delay distributions. The presented results were averaged over 50 runs.

Fig. 4 depicts the cumulative regret (as defined in (8)) obtained by Auer and LimExp for different values of K. As it can be seen, the limited number of explorations performed in LimExp leads to a significant regret reduction compared to Auer. Choosing $K = 4$ in the considered high FN density scenario yields the lowest regret. However, when $K = 3$, the regret increases (while still being much lower than the one obtained by Auer), which may be attributed to the fact that some FNs having very good delay performances were missed (not explored) when K was small.

For reference purposes, we show in Fig. 5 the cumulative regret obtained by the same algorithms in the medium density scenario (for the distributions shown in Fig. 2). As it can be

[9]https://firebase.google.com/docs/ml-kit/label-images

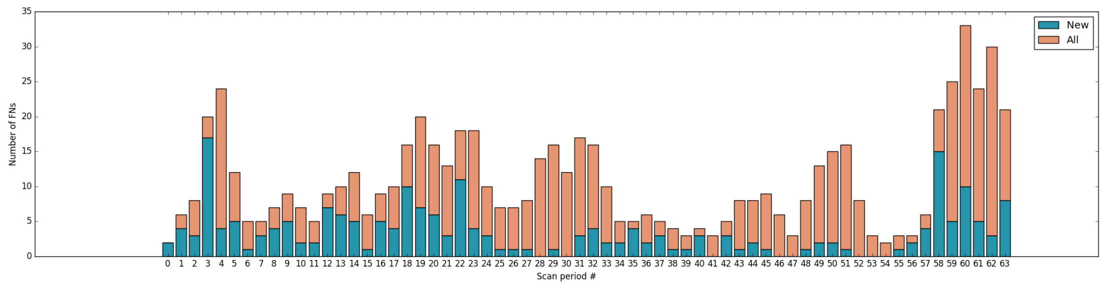


Fig. 2. FN distributions - Medium density case

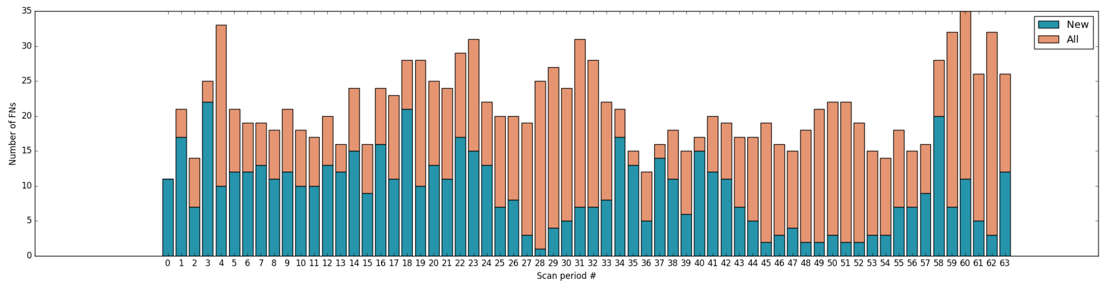


Fig. 3. FN distributions - High density case

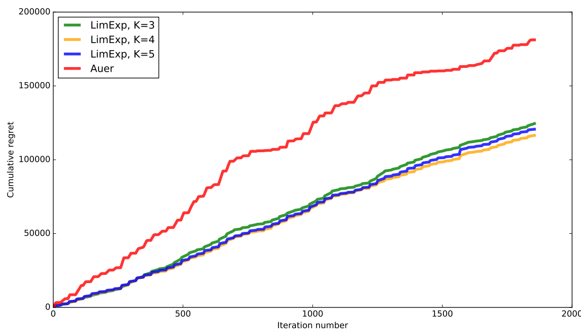


Fig. 4. Cumulative regret - High density case

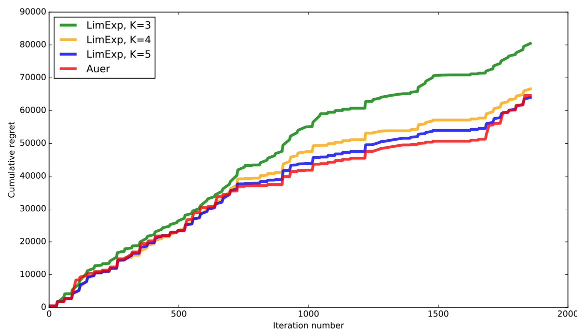


Fig. 5. Cumulative regret - Medium density case

seen, since the number of newly-appeared FNs in each epoch is low, it is more beneficial to explore them all (as done in Auer), thus leading to a lower regret. The gap between Auer and LimExp for K=3, indicates that the latter has repeatedly missed an FN with an optimal performance in multiple epochs.

To further highlight the benefits achieved by our proposed approach, we depict in Fig. 6 the instantaneous delay within a epoch where a high number of newly-appeared FNs has been observed. In this case, while Auer spends most of the time exploring the delay performances of the new FNs, thus leaving only a few trials for exploiting the best FN, LimExp spends only K trials performing an exploration phase, then it starts exploiting the best FN that it found. We note that even though this best FN is actually sub-optimal with regards to the one selected by the Oracle, it leads to a much better delay performance compared to the costly explorations in Auer. We also note that in this considered epoch, the best FN has been discovered in (a) previous epoch(s) and not in the current one. As a result, although the explorations performed in the current epoch may seem wasteful since they did not reveal an FN having a better performance, they are necessary for balancing exploration and exploitation.

Finally, in Fig. 7, we show the total number of FN switches performed by each approach for the whole simulated duration. Such FN switches highly depend on the number of FNs to explore. Then, as expected, since Auer explores every new FN, this results in the highest number of switches. However, reducing the number of explorations in LimExp considerably reduces the number of switches, thus providing a seamless

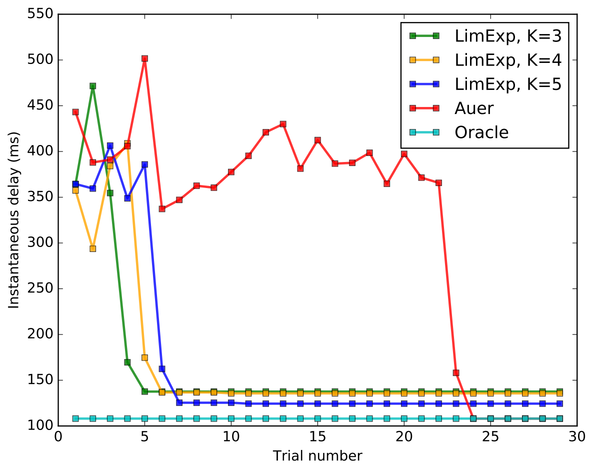


Fig. 6. Instantaneous delay

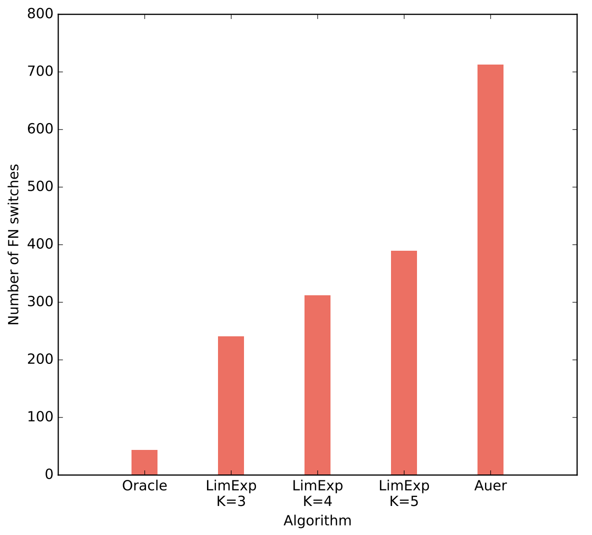


Fig. 7. Total number of FN switches

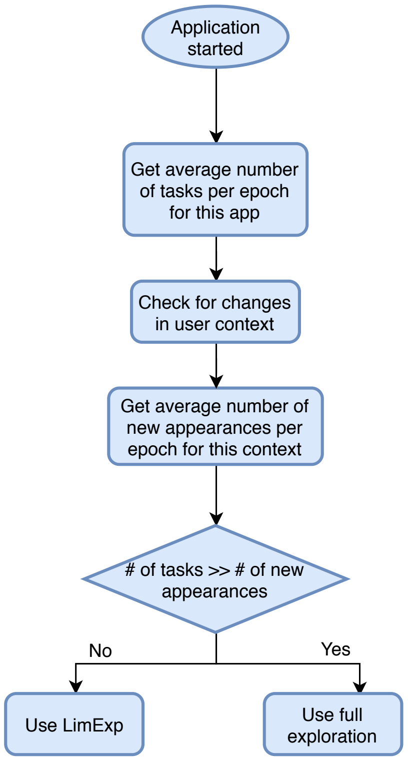
experience to the user.

VI. Discussion

In the following, we discuss some of the questions that may arise from the adoption of our proposed approach, namely with regards to the definition of the right context to use LimExp and the definition of the ideal value of K.

A. *When to use LimExp?*

As shown previously in Fig. 4, using LimExp when the number of newly-appeared FNs is high improves the learning performance significantly. In contrast, Fig. 5 reveals that limiting explorations (as done in LimExp) when the number of newly-appeared FNs is low does not lead to a performance enhancement. Thus, to avoid any potential losses due to the improper use of LimExp, context-awareness mechanisms should be implemented within the agent, as illustrated in the tentative scheme in Fig. 8. More specifically, when the application requiring FN capabilities starts, the agent will check previous execution statistics for this app and determines the average number of tasks that it is likely to generate within an epoch. Then, the agent will check if there are any changes in the user context (i.e. changes in the location due to mobility) and retrieves the average number of new FN appearances for the detected context. In case the average number of tasks is much larger than the number of new FN appearances, FNs will be fully explored (as done in Auer). In the opposite case, LimExp will be used.



Application started
Get average number of tasks per epoch for this app
Check for changes in user context
Get average number of new appearances per epoch for this context
# of tasks >> # of new appearances
No
Yes
Use LimExp
Use full exploration

Fig. 8. Envisioned flow to determine when to use LimExp

B. *What is the ideal value of K?*

As it can be noticed, LimExp uses a defined number K of FNs for exploration. Therefore, it is important to define K appropriately in order not to affect the learning performance. In fact, a low value of K may lead to missing optimal fog nodes, whereas a large K is almost equivalent to the full exploration setting. According to [9], K can be calculated based on a parameter β that characterizes the distributions of the suboptimal actions. Indeed, a small β corresponds to a high chance of selecting a good action, therefore, there is no need to select many actions (i.e. a low K is sufficient). [9] has also provided the theoretical regret bounds corresponding to different values of β. Then, leveraging these theoretical results, the agent in our case can determine the value of K to be used in LimExp by inferring the characteristics of the distributions of suboptimal FNs based on historical executions.

VII. Conclusion

In this paper, we studied the mobility management problem in a high-density fog computing environment. More specifically, we proposed a user-centric fog node selection scheme where a mobile user learns the best FN to serve its tasks using the multi-armed bandit approach. The key aspect that distinguishes our study from the rest of the literature is that we consider learning epochs where the number of newly-appeared FNs is close to the number of tasks that can be used to learn. As a result, inspired by the *many-armed* bandit variant, we proposed to consider only a fixed subset of FNs for exploring their performance. The rest of the time is spent connected to the best known FN. To support our results, we collected realistic user mobility data and task generation patterns and we found that limiting explorations when the FN density is high significantly improves the performance. As a future work, we will investigate ways to extend our scheme with smart service and computation caching at the FNs' side in order to obtain additional delay reductions.

Acknowledgment

This work was supported by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund under contract RTI2018-094532-B-100 and TEC2015-66220-R (MINECO/FEDER), and by the H2020 EU mF2C project Ref. 730929.

References

[1] 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 - MCC '12*, p. 13, ACM, 2012.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[3] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[4] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[5] J. Wang, K. Liu, M. Ni, and J. Pan, "Learning based mobility management under uncertainties for mobile edge computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[6] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, 2018.

[7] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-Based Task Offloading for Vehicular Cloud Computing Systems," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, may 2018.

[8] Y. Sun, J. Song, S. Zhou, X. Guo, and Z. Niu, "Task Replication for Vehicular Edge Computing: A Combinatorial Multi-Armed Bandit Based Approach," *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, dec 2018.

[9] Y. Wang, J.-Y. Audibert, and R. Munos, "Algorithms for infinitely many-armed bandits," in *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2009.

[10] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[11] F. Yu, H. Chen, and J. Xu, "DMPO: Dynamic mobility-aware partial offloading in mobile edge computing," *Future Generation Computer Systems*, vol. 89, pp. 722–735, 2018.

[12] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656.

[13] Z. Zhu, T. Liu, S. Jin, and X. Luo, "Learn and Pick Right Nodes to Offload," *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, dec 2018.

[14] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[15] R. Agrawal, M. Hedge, and D. Teneketzis, "Asymptotically efficient adaptive allocation rules for the multiarmed bandit problem with switching cost," *IEEE Transactions on Automatic Control*, vol. 33, no. 10, pp. 899–906, 1988.

[16] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma, "Regret bounds for sleeping experts and bandits," *Machine learning*, vol. 80, no. 2-3, pp. 245–272, 2010.

[17] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, "Social network search as a volatile multi-armed bandit problem," *HUMAN*, vol. 2, no. 2, pp. pp—-84, 2013.

[18] A. B. O. de Gortari, "Empirical study on Game Transfer Phenomena in a location-based augmented reality game," *Telematics and Informatics*, vol. 35, no. 2, pp. 382–396, 2018.

[19] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398–401, 2017.

[20] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers," *IEEE wireless communications*, vol. 24, no. 3, pp. 48–56, 2017.