On-demand Service Deployment Strategies for Fog-as-a-Service Scenarios

Arash Bozorgchenani, *Member, IEEE,* Daniele Tarchi, *Senior Member, IEEE,* and Walter Cerroni, *Senior Member, IEEE*

***Abstract*—Service deployment at the network edge is a promising area that has been studied recently in the literature. In this work we have investigated a Fog-as-a-Service scenario, where multiple Server Fog Nodes (SFNs) can serve multiple Client Fog Nodes (CFNs) by exploiting different service deployment models, i.e., SaaS, PaaS, and IaaS, in a flexible way. The system has been modeled as a Size-Constrained Weighted Set Cover Problem aiming at maximizing the amount of satisfied CFNs exploiting a heterogeneous service deployment architecture, while minimizing the service completion time in a computation offloading scenario. In the simulation results section, we analyze the performance of different methods in terms of percentage of CFNs' offloading requests satisfaction and offloading delay.**

***Index Terms*—Fog Computing, Service Deployment, Computation Offloading, Optimization, Weighted Set Cover Problem**

I. Introduction

FOG Computing can be seen as an extension of the Cloud Computing paradigm toward the network edge considering an intermediate layer between the users and the cloud aiming at reducing the latency while keeping the advantages of the latter [1]. Cloud-based services are historically organized in three main models, named Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS), each one involving different approaches, technologies and levels of flexibility. When moving from a centralized cloud architecture to a distributed edge architecture, a proper service model deployment policy becomes of paramount importance for coping with users requests while respecting their requirements [2], [3]. Some works have considered SaaS, PaaS or IaaS model deployment in edge networks [4]–[6]. Game-theoretic approaches have also been considered for resource allocation in cloud and fog environments [7], [8]. However, all the previous studies focused on a specific service model or application scenario, not taking advantage of the full flexibility offered by a joint adoption of the different service models. To this aim, we propose a Fog-as-a-Service (FogaaS) approach where the Fog Computing layer is able to select the proper models to be deployed in order to meet the user requests, while keeping the overall delay low. Such a user-centric approach is relevant with respect to the vision of future B5G/6G systems [9].

We consider a fog environment, where some nodes can work as Server Fog Nodes (SFNs), running applications and services based on requests from Client Fog Nodes (CFNs). In particular, in this letter we focus on computation offloading applications, one of the main classes of services enabled by fog computing [10]; the approach can be easily extended to other kinds of applications. We advocate the possibility of deploying the requested applications flexibly by leveraging on the presence of multi-purpose SFNs, which are able to implement any of the SaaS, PaaS and IaaS models. The problem is formulated as a Weighted Set Cover Problem (WSCP), where each CFN can be served by one SFN implementing one of the aforementioned models, whose selection depends on the overall CFNs requests. We intend to design a mechanism that aims to respond to all CFNs requests at the edge, and jointly minimize the offloading delay through a proper service model deployment. To this aim, our main contribution in this work is introducing both an optimal and a heuristic model deployment solution with different solution spaces. In the simulation results, we have proved that the proposed heuristic solution guarantees a complete CFNs coverage at the edge while minimizing the offloading delay.

II. System Model and Problem Formulation

We focus on a scenario composed by two types of nodes, namely CFNs and SFNs. While the CFNs are the nodes performing an offloading request for a given application type, the SFNs work as computing nodes, where the requested application can be deployed through one of the SaaS, PaaS or IaaS models; SaaS is able to only serve the CFNs requesting a given application, PaaS is able to serve all the CFNs whose requested application can be deployed on a specific platform, and IaaS is able to change at run-time both platform and application, hence being capable of responding to any request. We define N as the number of CFNs and i as the index for a specific CFN, while we consider to have S multi-purpose SFNs, shown with index s, aiming at remotely processing the requests received from the CFNs. Let us consider to have M applications, where m identifies a specific application. Moreover, let us assume there are at most P possible platforms, where p identifies a specific platform, i.e., a specific Operating System or set of libraries able to execute a code specifically written for that platform.

D. Tarchi and W. Cerroni are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy (e-mail: daniele.tarchi@unibo.it; walter.cerroni@unibo.it).

A. Bozorgchenani was with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy. He is now with the Department of Computing and Communications, Lancaster University, the UK (e-mail: a.bozorgchenani@Lancaster.ac.uk).

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.

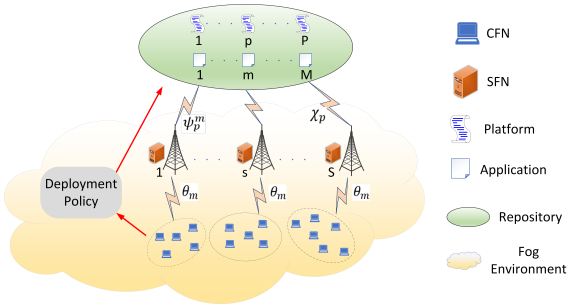

Fig. 1. FogaaS Network Architecture.

The architecture of the considered scenario is depicted in Fig. 1, where it is possible to notice that the CFNs can be served by different SFNs. Applications and platforms provided by the FogaaS network are supposed to be stored in a repository. A deployment policy is considered, aiming at selecting the proper service models to be deployed, with the goal of a complete CFNs request coverage. For a better clarity, some of the symbols used throughout the paper have been added in the figure.

Let us define $\mathbb{Q}_p^m$ as an indicator function that takes the value of 1 if the m-th application can be executed on the p-th platform. We now define the set of platforms on which the m-th application can be executed as $\mathcal{P}_m = \{p = 1, \ldots, P | \mathbb{Q}_p^m = 1\}$ and the set of applications that can be executed on p-th platform as $\mathcal{M}_p = \{m = 1, \ldots, M | \mathbb{Q}_p^m = 1\}$ which are non-empty sets, since, otherwise, the application to platform association would be meaningless.

We suppose that every CFN can set-up one and only one computation offloading request for a given application, hence we can identify with N_m the number of CFNs requesting the m-th application, where $N = \sum_{m=1}^{M} N_m$. If we define with CFN_i^m the generic i-th CFN when requesting the m-th application, it is possible to define the set of CFNs requesting the same application as $\{CFN_i^m\}_{i=1,\ldots,N_m}$.

By considering that in SaaS one SFN can only serve CFNs having the same application request, it is possible to define the collection of sets whose CFNs can be served with the SaaS model as:

$$\mathcal{S}_{SaaS} = \bigcup_{m=1,\ldots,M} \left(\{CFN_i^m\}_{i=1,\ldots,N_m} \right) \tag{1}$$

Similarly, the CFNs can be grouped depending on the platform that can serve them. It is worth to be noticed that there exists certain number of CFNs requesting applications that can be executed on the p-th platform. Hence, it is possible to define $\mathcal{S}_{PaaS}$ as the collection of all the possible sets grouping the CFNs whose application requests can be served by a specific platform through the PaaS model:

$$\mathcal{S}_{PaaS} = \bigcup_{p=1,\ldots,P} \left(\wp \left(\bigcup_{m \in \mathcal{M}_p} \{CFN_i^m\}_{i=1,\ldots,N_m} \right) \right), \tag{2}$$

representing the union of all the platforms' powersets[1] calculated over all the sets of CFNs whose application can be supported by the p-th platform. This corresponds to have the set of all the possible groups in which the CFNs can be organized depending on the requested application and the platform supporting it.

Finally, the collection of the sets that can be served with the IaaS model includes the powerset of all the possible combinations of sets composed by the CFNs requesting a given application and the supporting platform, hence:

$$\mathcal{S}_{IaaS} = \wp \left(\bigcup_{m \in \mathcal{M}_p, p=1,\ldots,P} \{CFN_i^m\}_{i=1,\ldots,N_m} \right) \tag{3}$$

IaaS allows the highest flexibility in composing the set of CFNs, by mixing CFNs with different platforms and application requests. However, the higher flexibility is obtained at the cost of a higher deployment time. The collection of the sets defining all the possible grouping that can be used for serving all the CFNs can be defined as the union of the three previously defined sets, i.e., $\mathcal{S} = \mathcal{S}_{SaaS} \cup \mathcal{S}_{PaaS} \cup \mathcal{S}_{IaaS}$. Let us focus for simplicity on the CFN requesting the m-th application and its associated SFN.

In case of SaaS, the m-th application is considered to be already deployed on the SFN. Hence, the CFN has to send only the input data θ_m, depending on the requested offloading application, and after the execution, the results, ϑ_m, will be eventually sent back. This means that the offloading completion time for any CFN requesting the application m using the SaaS model is:

$$T_{SaaS}^m = T_{tx}^{\theta_m} + T_{proc}^{\theta_m} + T_{rx}^{\vartheta_m} \tag{4}$$

where $T_{tx}^{\theta_m} = L(\theta_m)/R_s$ and $T_{proc}^{\theta_m} = O(\theta_m)/\eta_s$ are the input data transmission and processing times for the m-th application, respectively, while $T_{rx}^{\vartheta_m} = L(\vartheta_m)/R_s$ is the time required to have the result back. Let us denote by $L(\cdot)$ the task size to be offloaded/received to/from the sth SFN, R_s the data rate between the ith CFN[2] and the sth SFN, $O(\cdot)$ the number of operations to process a task, and η_m the computational capability of the sth SFN.

In case the application is not deployed directly through SaaS, a PaaS model can be used. In this case the platforms are supposed to be already deployed on the SFNs, while the application code ψ_p^m for executing a specific application m over the platform p, i.e., $m \in \mathcal{M}_p$, can be downloaded at request from the repository. In this case, the offloading completion time becomes:

$$T_{PaaS}^m = T_{tx}^{\psi_p^m} + T_{init}^{\psi_p^m} + T_{SaaS}^m + T_{prop}^{Re} \tag{5}$$

where $T_{tx}^{\psi_p^m} = L(\psi_p^m)/R_{Re}$, $T_{init}^{\psi_p^m}$ and T_{prop}^{Re} are the m-th application code transmission time, the related initialization time for a given p-th platform, and the propagation delay from SFN to repository, respectively, and R_{Re} represents the data rate for accessing the repository.

Finally, in case of IaaS, the system should be able to deploy at run-time the requested platform and application for any CFN, if not already deployed. In the worst case when

[1]The powerset $\wp(\cdot)$ of a set $\mathcal{S}$ refers to the set of all the possible combinations of subsets composing $\mathcal{S}$.

[2]More specifically, the requested application is generated by the ith CFN, however, for the sake of simplicity of notation we omit the CFN index.

both platform and application requested by a CFN must be deployed, the offloading completion time becomes:

$$T^m_{IaaS} = T^{\chi_p}_{tx} + T^{\chi_p}_{init} + T^m_{PaaS} + T^{Re}_{prop} \tag{6}$$

where $T^{\chi_p}_{tx} = L(\chi_p)/R_{Re}$ and $T^{\chi_p}_{init}$ are the times needed to download the image and initialize a virtual machine or container, χ_p, that provides the p-th platform. In addition, the time needed to transmit and initialize the platform-specific application code and the parameter transmission and execution time are considered.

The optimal service deployment can be modeled as an optimization problem with the goal to guarantee full CFN request coverage while minimizing the overall offloading time. Among all possible combinations for covering all CFNs, we define $C_\pi = \{C_1, ..., C_S\}$ as a feasible grouping solution considering policy π. C_π is composed of maximum S sets of CFNs connected to the S SFNs, on each of which which one of the service models is implemented.

Definition 1: A solution C_π is considered feasible if it meets the following conditions:

$$\mathbf{C1}: i \in C^+_\pi \implies i \notin C^-_\pi \tag{7a}$$

$$\mathbf{C2}: \sum_{s=1}^{S} |C_s| = N, \tag{7b}$$

where C^+_π is one subset from the solution set C_π, and C^-_π is the complement of C^+_π w.r.t. C_π, i.e. $C^-_\pi = C_\pi \setminus C^+_\pi$. The first condition assures that a CFN does not belong to two groups, and the second condition guarantees that all CFNs are covered through the selected policy.

The goal of the problem is finding a policy π for grouping the CFNs and deploying proper service models in the network to jointly minimize the offloading delay and cover all the CFNs. We define $\Phi(\pi)$ as an $[N \times S]$ allocation matrix where each element represents the allocation of each i-th CFN to s-th SFN, when the policy π is selected. Moreover, we define $T(\pi)$ as a $[N \times S]$ delay matrix for policy π where each element represents the delay for completing the offloading request of the i-th CFN to the allocated SFN, having implemented one of the service models exploiting (4), (5) or (6). The considered problem can be formulated as an Integer Linear Program as:

$$\mathbf{P1}: \underset{\pi}{\operatorname{argmin}} \{\operatorname{tr}[(T(\pi) \odot \Phi(\pi)) \cdot J_{S\times N}]\}, \tag{8}$$

w.r.t., $\mathbf{C1} - \mathbf{C2}$, where $\odot$ refers to the Hadamard product of the matrices, $J_{S\times N}$ is the all-one matrix with size $[S \times N]$, and $\operatorname{tr}(\cdot)$ stands for the trace. Eq. (8) allows to find the policy π for deploying the proper models on the SFNs, where the offloading delay, corresponding to the sum of the delay of all the allocated models meeting the feasibility conditions (7) (i.e., full CFN coverage), is minimized[3].

III. Problem Transformation and Solutions

The problem can be modeled as a size-constrained WSCP, where the goal is optimizing the set selection for covering all the elements, while considering a predefined number of SFNs (i.e., a constrained number of sets) whose offloading time acts as a weight for the set selection. In this section we reformulate **P1** as a WSCP where this transformation enables to group the CFNs constrained by the number of SFNs.

A. WSCP-based service model deployment

Let us define $\hat{T}$ as a $[N\times(M+P+\Lambda)]$ delay matrix related to each possible combination in $\hat{\Phi}$, which is a $[N \times (M+P+\Lambda)]$ binary allocation matrix, defined as below:

$$\hat{\Phi} = \begin{pmatrix} \hat{\phi}^1_1 & \cdots & \hat{\phi}^M_1 & \hat{\phi}^{M+1}_1 & \cdots & \hat{\phi}^{M+P}_1 & \hat{\phi}^{M+P+1}_1 & \cdots & \hat{\phi}^{M+P+\Lambda}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{\phi}^1_N & \cdots & \hat{\phi}^M_N & \hat{\phi}^{M+1}_N & \cdots & \hat{\phi}^{M+P}_N & \hat{\phi}^{M+P+1}_N & \cdots & \hat{\phi}^{M+P+\Lambda}_N \end{pmatrix} \tag{9}$$

Each column in $\hat{\Phi}$ represents one of the possible SaaS, PaaS, or IaaS models to be deployed. Since each CFN can request one application at a time, in each row only one of the elements between column 1 and M takes the value of 1, i.e., $\sum_{k=1}^{M} \hat{\phi}^k_i = 1, \forall i$. Moreover, in each row the number of 1s between columns $M+1$ to $M+P$ depends on the number of platforms supporting the application requested by the i-th CFN; hence, we have $\sum_{p=1}^{P} \hat{\phi}^{(M+p)}_i = |\mathcal{P}_m|, \forall i$. On the other hand, Λ is the total number of possible IaaS deployments, corresponding to the cardinality of the powerset $\mathcal{S}_{IaaS}$.

The optimal solution results in selecting S out of $(M+P+\Lambda)$ possible combinations. Since Λ corresponds to the number of possible sets in (3), being composed by all the possible combinations of CFNs irrespective to their requested application and platform, it is possible to derive the cardinality of the IaaS sets as[4] $\Lambda = \sum_{l=2}^{M+P} \binom{M+P}{l}$. The solution space[5] of the problem reults to be equal to $\left(\frac{(M+P+\Lambda)!}{(M+P+\Lambda-S)!\cdot S!}\right)$ where S deployments out of $(M+P+\Lambda)$ should be selected.

The WSCP is NP-complete, hence, it cannot be solved in a polynomial time. Moreover, the solution space for the derived problem grows exponentially w.r.t. the number of applications and platforms leading to a non-tractable solution space. However, the NP problems, even though require exponential time, still grow slowly enough allowing solutions for problems of a useful size [11]. In order to solve the problem we propose two possible solutions: an optimal solution applied to a reduced solution space, and a heuristic solution applied to the full solution space.

B. Solutions to the problem

1) Reduced Space Optimal Solution: In this case we consider to restrict the solutions to SaaS and PaaS models. The allocation matrix $\hat{\Phi}$ can be simplified considering the deployment of SaaS and PaaS for the WSCP. Let us define $\bar{T}$ as a $[N \times (M+P)]$ delay matrix, and $\bar{\Phi}$ as a $[N \times (M+P)]$ binary allocation matrix as below:

$$\bar{\Phi} = \begin{pmatrix} \bar{\phi}^1_1 & \cdots & \bar{\phi}^M_1 & \bar{\phi}^{M+1}_1 & \cdots & \bar{\phi}^{M+P}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{\phi}^1_N & \cdots & \bar{\phi}^M_N & \bar{\phi}^{M+1}_N & \cdots & \bar{\phi}^{M+P}_N \end{pmatrix} \tag{10}$$

[3]Note that $\operatorname{tr}(A_{M\times N} \cdot J_{N\times M})$ corresponds to sum all the elements of A.

[4]The deployment of the IaaS is needed only if the CFNs request at least two applications that are supported by two different platforms, otherwise SaaS and PaaS is sufficient, hence going back to the solutions modeled in the first $M+P$ columns.

[5]It should be noted that not all the solutions in this space respect the feasibility conditions defined in Def.1.

where $\bar{\phi}_i^x$ shows one element of the matrix. Hence, the number of 1s in each row of $\bar{\Phi}$ is $|\mathcal{P}_m|$+1, representing all the platforms that support the m-th requested service plus the requested m-th service. Consequently, the total number of 1s in $\bar{\Phi}$ for all CFNs equals $(|\mathcal{P}_m|+1) \cdot N$.

Now we map the optimization problem to a WSCP, where the goal is optimizing the set selection policy π for covering all the CFNs. In other words, the set selection policy chooses the models that should be deployed on the SFNs to cover all the CFNs in S groups by transforming the allocation matrix $\bar{\Phi}$ to a matrix with S columns representing the selected models for deployment on S SFNs. Let us define $F_X(\pi)$ as a linear transformation matrix with policy π, mapping matrices $\bar{\Phi}$ and $\bar{T}$ to two matrices of size $[N \times S]$. The optimization problem can now be formulated in the following way:

$$\mathbf{P2}: \underset{\pi}{\operatorname{argmin}} \left\{ \operatorname{tr} \left[\left(F_{\bar{T}}(\pi) \odot F_{\bar{\Phi}}(\pi) \right) \cdot J_{S \times N} \right] \right\}, \tag{11}$$

w.r.t., $\mathbf{C1} - \mathbf{C2}$. We can define the set of CFNs in the group $C_s \in C_\pi$ as $C_s = \left\{ i | \tilde{\phi}_i^s = 1, \forall i \right\}$, where $\tilde{\phi}_i^s$ is an element of transformed binary matrix $F_{\bar{\Phi}}(\pi)$. The goal is finding the optimal policy π based on which the linear transformation function selects the best CFNs grouping and model deployment on the SFNs among the feasible solutions to minimize the offloading delay. The solution space of the problem is $\left(\frac{(M+P)!}{(M+P-S)! \cdot S!} \right)$, which is smaller w.r.t. the original $\hat{\Phi}$.

2) Full Space Heuristic Solution: In this section we propose a heuristic for full coverage of CFNs' requests at the edge while minimizing the delay. Our proposed XaaS-based model Deployment Policy (XaDeP) exploits all the three service models, where X stands for any of the three models. To this aim, enlightened by the Concise Weighted Set Cover (CWSC) algorithm [12], our heuristic leverages the number of covered CFNs, i.e., $|C_\iota|$, and the coverage latency for the model selection to be deployed on the S SFNs. We denote the latency of the $|C_\iota|$ covered CFNs by model ι with T_ι, as defined in (4) or (5). Let us define the input $\Theta = \{CFN\}_{i=1}^N$, as the set of all uncovered CFNs, and in the output the indexes of the selected models, $\bar{\mathcal{S}}$, and the group of covered CFNs with the policy π (i.e., selected by the XaDeP algorithm), C_π. We also define $\mathcal{V} = M + P$, as the set of all the SaaS and PaaS models. The algorithm iteratively calculates T_ι and $|C_\iota|$ for all SaaS and PaaS models and selects those maximizing $|C_\iota|/T_\iota$. The selected models are removed from $\mathcal{V}$ and added to $\bar{\mathcal{S}}$, and similarly, the set of covered CFNs are removed from Θ and added to C_π. This rule allows to select at each iteration the model that covers the maximum number of uncovered CFNs while minimizing the cost in terms of offloading time. In case the CFNs are not fully covered at the end of the previous procedure, the algorithm deploys IaaS model to cover the remaining CFNs. This is made possible thanks to the flexibility introduced by the IaaS model, at the cost of a higher deployment time. The pseudo-code of the algorithm has been provided in Alg. 1.

When deploying the model for the last SFN, if **C2**, (7b), is respected the algorithm stops and returns $\bar{\mathcal{S}}$ and C_π (lines 13-14), otherwise, the XaDeP algorithm includes also the Λ IaaS models in (3) and, among them, selects the one that respects **C2** and minimizes the delay (lines 16-21). Thanks to this approach, XaDeP first tries to cover all the CFNs using SaaS and PaaS models, hence minimizing the latency, and if (7b) is not respected, exploits the IaaS model for covering the remaining CFNs.

Algorithm 1 XaDeP Algorithm

1: **Input:** M, P, S, Λ, Θ
2: **Output:** $\bar{\mathcal{S}}$, C_π
3: **for** s=1 to $S-1$ **do**
4: **for** ι=1 to $\mathcal{V}$ **do**
5: Calculate $|C_\iota|$; Calculate T_ι using Eq. (4) or (5), $\forall$ $|C_\iota|$
6: **end for**
7: $\bar{\mathcal{S}} \longleftarrow \operatorname{argmax}_\iota \frac{|C_\iota|}{T_\iota}$; $C_\pi \longleftarrow C_\iota$; $\mathcal{V} - \iota$; $\Theta - C_\iota$
8: **end for**
9: **for** ι=1 to $\mathcal{V}$ **do**
10: Calculate $|C_\iota|$; Calculate T_ι using Eq. (4) or (5), $\forall$ $|C_\iota|$
11: **end for**
12: Select the last model using $\operatorname{argmax}_\iota \frac{|C_\iota|}{T_\iota}$
13: **if C2** holds with the last model **then**
14: $\mathcal{V} - \iota$; $\Theta - C_\iota$; $\bar{\mathcal{S}} \longleftarrow \iota$; $C_\pi \longleftarrow C_\iota$; return $\bar{\mathcal{S}}$ and C_π
15: **else**
16: **for** each ι in Λ **do**
17: **if C2** holds with ι **then**
18: Calculate T_ι using Eq. (6)
19: **end if**
20: **end for**
21: $\mathcal{V} - \iota$; $\Theta - C_\iota$; $\bar{\mathcal{S}} \longleftarrow \operatorname{argmin}_\iota T_\iota$; $C_\pi \longleftarrow C_\iota$; return $\bar{\mathcal{S}}$ and C_π
22: **end if**

IV. Numerical Results

For the simulation results, we have set the number of SFNs to S=3, and the number of platforms to P=5. We have set the task size θ (either θ_m or θ_c) uniformly distributed in the range [1 5] MB, a small-sized task-result of $\theta/10$ MB, 10 GFLOPS per MB for $O(\theta)$, 150 GFLOPS for the η_s, while R_s and R_{Re} are maximum 200 Mb/s and 150 Mb/s, with a log-distance path loss model, as in [10]. Regarding the PaaS and IaaS models, the size of ψ and χ are uniformly distributed in the range [1 5] MB and [5 10] MB, respectively, and their initialization time uniformly distributed in [0 0.2] s and [0.5 2] s, respectively. These values are based on internal measurements performed on a Docker based virtualization infrastructure, and are comparable with other in literature [10]. We are analyzing the performance of the following solutions:

- Optimal Solution: The optimization problem **P2** is solved using standard solver *CPLEX* considering the reduced space in Sec. III-B1, labeled *Opt.* in the figures.
- XaDeP: Our proposed heuristic is applied where IaaS is also considered for full coverage of the CFNs' requests, labeled *XaDeP* in the figures.
- CWSC: The heuristic algorithm in [12] is applied considering to deploy only SaaS and PaaS resulting in partial covering of the CFNs, labeled *CWSC* in the figures.
- SaaS: Only the SaaS model is considered for deployment at the SFNs, labeled *SaaS* in the figures. This is a benchmark for the proposed FogaaS model.
- PaaS: Only the PaaS model is considered for deployment at the SFNs, labeled *PaaS* in the figures. This is a benchmark for the proposed FogaaS model.

In order to have a fair comparison among all algorithms in terms of service delay, we have considered that the uncovered CFNs at the edge are able to access to a remote cloud. To this aim, we include a CFN-Cloud offloading delay as $T^{cloud} = T_{tx}^{\theta_c} + T_{proc}^{\theta_c} + T_{rx}^{\vartheta_c} + T_{prop}^{c}$ where θ_c is the task size to be

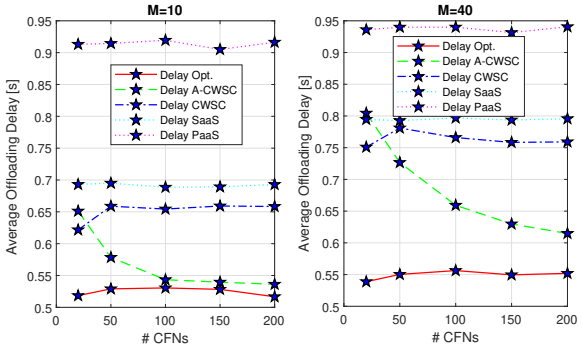

Fig. 2. Average per-CFN offloading delay.

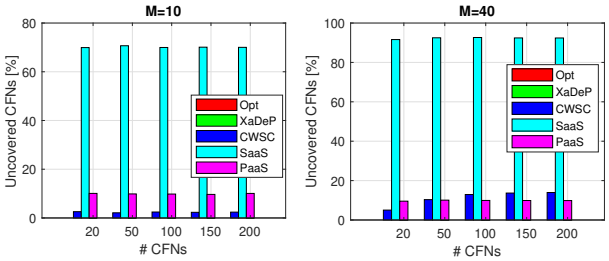

Fig. 3. Average Percentage of Uncovered CFNs at edge.

offloaded/processed to/in the cloud. The propagation time to cloud and repository are set T^{c}_{prop} = 0.2 s and T^{Re}_{prop} = 0.1 s, respectively, while the data rate to the cloud is supposed to be 100 Mb/s. Thus, the offloading delay is $T(\pi)+T^{cloud}$, whether the task is offloaded to the cloud or edge. It is worth to be mentioned that both optimal and XaDeP solutions are able to cover all the CFNs at the edge, thus for these solutions T^{cloud} can be neglected. The results have been obtained by averaging over 1000 rounds, each representing random CFN requests.

Figs. 2 and 3 depict the average per-CFN offloading completion time (labeled *Delay*), and the average percentage of uncovered CFNs, respectively, for different numbers of applications and CFNs. CFNs are supposed to be randomly placed in an area covered by the three SFNs. Moreover, Table I shows the Overall Delay (O.D.) of each algorithm, corresponding to the sum of per CFN offloading delay (in Fig. 2) and the execution Time Complexity (T.C.) for different number of CFNs and applications, obtained with a dual core Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz. For M=10, the optimal solution has the best performance in terms of offloading delay while ensuring a full CFN coverage. However, the offloading delay minimization is overshadowed by T.C. whose value depends on the considered hardware when solving the problem for every run. This high O.D. in Tab. I, is due to the T.C. required by the optimal solution which grows exponentially as the number of CFNs rises. This makes our XaDeP heuristic as the alternative approach in the long run to guarantee a full coverage of the CFNs with an offloading delay which is the closest to the optimal one. Observing Tab. I we can see that, except for the case of 20 CFNs, our proposed XaDeP outperforms the other algorithms. This is because, after deploying the IaaS model at the edge, applications/platforms are downloaded once from the repository, hence higher the CFNs more the gain due to the reduced offloading delay at the edge. Based on our observations, despite SaaS and PaaS have the smallest T.C., SaaS leaves the largest percentage of

TABLE I
OVERALL DELAY FOR M = 10 AND M = 40 EXPRESSED IN SECONDS.

	20 CFN		**50 CFN**		**100 CFN**		**150 CFN**		**200 CFN**	
App. / Ben.	10	40	10	40	10	40	10	40	10	40
Opt	1.38	6.3	1.68	18.2	2.77	45.1	4.61	63.5	9.1	107
XaDeP	0.65	0.8	0.57	0.72	0.54	0.66	0.54	0.63	0.53	0.61
CWSC	0.62	0.75	0.65	0.78	0.65	0.76	0.65	0.75	0.65	0.75
SaaS	0.69	0.79	0.69	0.79	0.68	0.79	0.68	0.79	0.69	0.79
PaaS	0.91	0.93	0.91	0.94	0.91	0.94	0.90	0.93	0.91	0.94

uncovered CFNs, and PaaS has the largest offloading delay; hence, since the uncovered CFNs have to access the cloud, the O.D increases. Similar performance can be observed for the case of M = 40. In the end, the proposed Optimal approach has the best performance in terms of offloading delay. However, with higher number of CFNs or applications, its T.C. rises. The XaDeP on the other hand, guarantees a full coverage with the lowest O.D. thanks to the flexibility that it offers for offloading to the network edge.

V. CONCLUSION

In this work, we have studied a FogaaS architecture able to deploy flexibly different applications at the edge upon CFNs requests. To the best of our knowledge this is the first study formulating this problem for minimizing the offloading delay, and guaranteeing a full coverage of the CFNs in the form of WSCP. In numerical results we have analyzed the trade-off between the CFNs' requests coverage and service delay arising by the different service model deployments.

REFERENCES

[1] T. Chiu, A. Pang, W. Chung, and J. Zhang, "Latency-driven fog cooperation approach in fog radio access networks," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 698–711, Sep./Oct. 2019.

[2] I. Lera, C. Guerrero, and C. Juiz, "Availability-aware service placement policy in fog computing based on graph partitions," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3641–3651, Apr. 2019.

[3] R. Moallemi, A. Bozorgchenani, and D. Tarchi, "An evolutionary-based algorithm for smart-living applications placement in fog networks," in *2019 IEEE Globecom Workshops (GC Wkshps)*, Waikoloa Village, HI, USA, Dec. 2019.

[4] A. Bonadio, F. Chiti, and R. Fantacci, "Performance analysis of an edge computing SaaS system for mobile users," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2049–2057, Feb. 2020.

[5] C. Mouradian, F. Ebrahimnezhad, Y. Jebbar, J. K. Ahluwalia, S. N. Afrasiabi, R. H. Glitho, and A. Moghe, "An IoT platform-as-a-service for NFV based-hybrid cloud/fog systems," *IEEE Internet Things J.*, 2020, early access.

[6] S. Shaik and S. Baskiyar, "Resource and service management for fog infrastructure as a service," in *2018 IEEE International Conference on Smart Cloud (SmartCloud)*, 2018, pp. 64–69.

[7] F. Zafari, K. K. Leung, D. Towsley, P. Basu, and A. Swami, "A game-theoretic framework for resource sharing in clouds," in *2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2019, pp. 8–15.

[8] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.

[9] S.Dang, O.Amin, B. Shihada, and M. Alouini, "What should 6G be?" *Nature Electronics*, vol. 3, no. 1, pp. 20–29, 2020.

[10] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, May 2020, early view.

[11] C. A. Shaffer, *Data Structures and Algorithm Analysis in Java*, 3rd ed. Dover Publication, 2013.

[12] L. Golab, F. Korn, F. Li, B. Saha, and D. Srivastava, "Size-constrained weighted set cover," in *2015 IEEE 31st International Conference on Data Engineering*, Seoul, South Korea, Apr. 2015, pp. 879–890.