

Multilayer Resource-aware Partitioning for Fog Application Placement

Zahra Najafabadi Samani, Nishant Saurabh, Radu Prodan
Institute of Information Technology, University of Klagenfurt, Austria
zahra,nishant,radu@itec.aau.at

Abstract—Fog computing emerged as a crucial platform for the deployment of IoT applications. The complexity of such applications require methods that handle the resource diversity and network structure of Fog devices, while maximizing the service placement and reducing the resource wastage. Prior studies in this domain primarily focused on optimizing application-specific requirements and fail to address the network topology combined with the different types of resources encountered in Fog devices. To overcome these problems, we propose a multilayer resource-aware partitioning method to minimize the resource wastage and maximize the service placement and deadline satisfaction rates in a Fog infrastructure with high multi-user application placement requests. Our method represents the heterogeneous Fog resources as a multilayered network graph and partitions them based on network topology and resource features. Afterwards, it identifies the appropriate device partitions for placing an application according to its requirements, which need to overlap in the same network topology partition. Simulation results show that our multilayer resource-aware partitioning method is able to place twice as many services, satisfy deadlines for three times as many application requests, and reduce the resource wastage by up to 15 – 32 times compared to two availability-aware and resource-aware state-of-the-art methods.

Index Terms—Fog computing, application placement, resource partitioning, resource wastage, deadline satisfaction.

2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

I. INTRODUCTION

The Cloud-assisted Internet of Things (IoT) emerged as an essential accelerator of the fourth industrial revolution [1]. However, the rapid growth of IoT applications with multiple services makes it challenging for Clouds to satisfy their low-latency real-time requirements [2] and deadline constraints. To mitigate these challenges, *Fog computing* [3] emerged as a crucial platform consisting of a large number of heterogeneous and geographically distributed resources, hierarchically split between the Cloud and the IoT resources. This enables the Fog to address latency-centric constraints posed by the Cloud through systematic placement [4] of services across resources closer to the IoT devices. However, the non-uniform Fog network composed of heterogeneous devices (e.g., routers, switches, gateways) significantly varies in terms of processing speed, network bandwidth and storage capacity [2], which

induces service availability and deadline fulfillment challenges for time-critical IoT applications.

To solve IoT application placement issues in Fog, there is a need to research strategies that increase its resource utilization, while fulfilling diverse application requirements. Prior researches leveraged evolutionary multi-objective optimization algorithms [5], and linear programming [6], [7] models to identify cost-efficient IoT application placement strategies. Such approaches primarily focused on optimizing energy consumption, network usage and response time of IoT service requests, but limited to application specific requirements without considering resource dependencies among interrelated services. Other works [8]–[12] improved upon these studies and explored resource partitioning at a network topology level without considering other resources of Fog devices, such as processing speed, memory or storage sizes. Moreover, these methods failed to address the resource wastage, while optimizing the utilization of capacity-constrained Fog resources.

We approach this problem using a *multilayer resource-aware partitioning* method that handles the resource diversity and the interconnection structure of Fog devices to minimize the resource wastage and optimize the application service placement. To address the resource diversity, we model the heterogeneous Fog infrastructure as a multilayer graph comprising the network topology and heterogeneous devices with different CPU speed, memory and storage capacities. To optimize the application placement, we split the Fog devices in overlapping partitions with respect to the network topology and different resource types. This enables our method to target the correct set of resources for an application and improve its availability. Afterwards, the placement involves two steps.

a) *Feature partition selection*: matches the requested application services with partitions based on their resource requirements, which need to overlap in the same network topological partition underneath.

b) *Service placement*: maps services onto the appropriate Fog devices in the elected partitions, such that all application services reside in the same network partition.

Extensive simulation experiments demonstrate that our multilayer resource-aware partitioning method is able to place up to twice as many services, satisfy deadlines for up to three times as many application requests, and reduce the Fog resource wastage by up to 15 – 32 times compared to two related state-of-the-art methods [1], [10].

The paper is organized in seven sections. Section II summarizes the related work. Section III presents the model underneath our method, including the multilayered Fog representation and its key layer partitioning method in section IV. Section V describes multilayer Fog placement method, including two feature partition selection and service topology optimization algorithms. Section VI presents the experimental results and Section VII concludes the paper.

II. RELATED WORK

This section revisits the recent related works on Fog application placement across the following categories.

a) Application-aware placement: Huang et al. [7] proposed an integer programming model to optimize the energy cost by merging neighboring services onto a single device in a multi-hop Fog network. Oueis et al. [4] formulated a resource allocation model in Fog to jointly optimize power consumption and latency by clustering resources and assigning each cluster to a requested application service. Velasquez et al. [6] designed an integer linear programming placement for IoT services across a Cloud-Edge infrastructure that optimizes service latency. Naha et al. [13] proposed a resource provisioning algorithm for deadline-based application placement in Fog to optimize processing time, processing cost and network delay. These approaches primarily focused on optimizing application-specific requirements without considering the network topology and resource heterogeneity.

b) Topology-aware placement: Filiposka et al. [11] designed a community-based Fog management that exploits distributed hierarchical clustering to reduce latency and optimize service migration. Similarly, Asensio et al. [12] proposed a distributed control and management approach that groups Fog resources based on their network topology to minimize latency and energy consumption. In contrast, Sun et al. [5] modeled the resource placement in Fog as a multi-objective execution and latency optimization that computes different service clusters for an application. These approaches utilized clustered Fog resources based on network topology without considering the infrastructure heterogeneity.

c) Resource-aware placement: Shooshtarian et al. [9] proposed a two-phase allocation method that hierarchically represents Fog resources and performs local clustering in each layer to optimize resource utilization and network delay. Taneja et al. [1] proposed a resource-aware application mapping approach to optimize resource utilization in Fog. Similarly, Stefanic et al. [14] proposed a subgraph pattern matching approach for application placement that maps the multi-tier application graph onto the Fog infrastructure to improve resource utilization. Nevertheless, these three approaches only consider different Fog resources characteristics and ignore the network topology structure of Fog devices. Contrarily, Lera et al. [10] proposed a greedy approach for application placement that optimizes availability and latency by partitioning Fog resources in hierarchical clusters based on their connectivity. However, it fails to minimize resource

wastage by ignoring Fog resource characteristics during the greedy-based assignment.

III. MODEL

This section presents a formal model essential to this work.

A. Resource infrastructure

A *Fog infrastructure* consists of three layers:

1) *Cloud layer:* represents a data center with high-performance computing resources.

2) *Fog network layer:* $F = (D, N)$ lies between the Cloud and the end-users, and provides close proximity computational and storage services on top of two resource sets, according to the architecture proposed by the OpenFog consortium [3]:

a) Physical devices: $D = \{d_1, d_2, \dots, d_n\}$, modeled as a triplet of resources $d_i = (R_{i1}, R_{i2}, R_{i3})$, where R_{i1} represents the speed of a CPU core in millions of instructions (MI) per second, R_{i2} represents the memory size in GB, and R_{i3} represents the storage size of a device d_i in TB.

b) Network connections: $N = \{n_{ij} | (d_i, d_j) \in D \times D\}$ between a subset of devices, where a network connection $n_{ij} = (BW_{ij}, LAT_{ij})$ depends on the bandwidth BW_{ij} and the latency LAT_{ij} between the devices d_i and d_j .

3) *Client layer:* consists of a set of users $\mathcal{U} = \{u_1, \dots, u_q\}$, including sensor and actuator client devices that request Fog resources for placing their applications. We do not consider user mobility in this work.

B. Application model

We model an *application* $A = (\mathcal{S}, M, \theta, u)$ requested by a user u as a directed graph of *services* $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ interconnected through request *messages* M . Every service $s_i \in \mathcal{S}$ has a triplet of resource demands $s_i = (r_{i1}, r_{i2}, r_{i3})$, where r_{i1} is the workload in MI, r_{i2} is the required memory size, and r_{i3} is the storage size.

We model each *request message* $m_{ij} \in M$ based on its size SZ_{ij} , its source $s_i \in \mathcal{S}$, and its destination service $s_j \in \mathcal{S}$: $m_{ij} = (SZ_{ij}, s_i, s_j)$. A user $u \in \mathcal{U}$ triggers the application execution via an *initial request message* m_{ui} to service s_i .

The application also has a completion *deadline* that requires the completion of all its services.

C. Multilayer Fog model

The Fog network layer represents the topological interconnection of the physical devices and does not capture their heterogeneous resources. To better handle resource diversity for application placement, we model the Fog across four layers $L = \{l_0, l_1, l_2, l_3\}$ representing relationships among its devices based on the network topology and three resource types.

(i) *Network layer* l_0 corresponds to the Fog network layer modelled in Section III-A.

(ii)–(iv) *CPU layer* l_1 , *memory layer* l_2 , and *storage layer* l_3 indicate similarities among the Fog devices according to their CPU speed, memory size, and storage size.

We model the Fog as a *fully interconnected multilayer graph* $G = (D, \mathcal{E}, L)$, where D is the set of Fog devices replicated across all four layers L , and $\mathcal{E} = \{E_{ll'} | \forall l, l' \in [0, L]\}$ is the set of weighted bidirectional graph edges of two types.

a) *Inter-layer edges*: $E_{ll'} = \{(d_i, d_i) \in D \times D\}$ connect each device d_i in the layer $l \in L$ with the corresponding device d_i in all the other layers $l' \in L, l \neq l'$. They indicate the connection between different resource characteristics of the same device and uncover the relation among resource types;

b) *Intra-layer edges*: $E_{ll} = \{(d_i, d_j) \in D \times D | i \neq j\}$ connect two Fog devices inside one layer $l \in L$ using a weight function $w^{(l)} : E_{ll} \rightarrow \mathbb{R}$, representing their *similarity score*:

$$w_{ij}^{(l)} = \frac{1}{1 + d_l(d_i, d_j)},$$

where $d_l(d_i, d_j)$ is the Euclidean distance between their resource characteristics in layer $l \geq 1$: $d_l(d_i, d_j) = R_{il} - R_{jl}$. The Fog devices with a similarity score of 1 are exactly similar with respect to a resource R_l .

D. Problem statement

We introduce a number of definitions introducing our problem statement of placing an application $A = (\mathcal{S}, M, \theta, u)$ in a Fog environment $F = (D, N)$.

1) *Application placement*: is a function $\mu : \mathcal{S} \rightarrow D \cup \emptyset$, where $\mu(s_i) = d_j$ satisfies the constraints for placing each service $s_i = (r_{i1}, r_{i2}, r_{i3})$ on a Fog device $d_j = (R_{j1}, R_{j2}, R_{j3})$: $\frac{r_{i1}}{R_{j1}} \leq \theta$, $r_{i2} \leq R_{j2}$ and $r_{i3} \leq R_{j3}$. An *invalid placement* $\mu(s_i) = \emptyset$ indicates that there exist no device in D that satisfies the service constraints.

2) *Execution time*: of a service s_i is the ratio between its workload r_{i1} and the speed R_{i1} of the underlying hosting device $d_j = \mu(s_i)$: $ET_{i,j} = \frac{r_{i1}}{R_{j1}}$.

3) *Transmission time*: of a message $m_{ij} \in M$ of size SZ_{ij} between two devices d_i and d_j is: $T_{ij} = LAT_{ij} + \frac{SZ_{ij}}{BW_{ij}}$, where LAT_{ij} is the latency and BW_{ij} is the bandwidth of a network connection $n_{ij} \in N$.

4) *Response time*: of a service $s_i \in \mathcal{S}$ running on the device $d_j = \mu(s_i)$ is the sum between the maximum response time RT_{pq} of its predecessors s_p , including its request message transmission time T_{qj} , where $\mu(s_p) = d_q$ (or T_{uj} , if initial message request), and its execution time ET_{ij} :

$$RT_{i,j} = \begin{cases} T_{uj} + ET_{ij}, & \exists m_{ui} \in M; \\ \max_{m_{pi} \in M} \{RT_{pq} + T_{qj}\} + ET_{ij}, & \exists m_{pi} \in M \wedge s_p \in \mathcal{S}. \end{cases}$$

5) *Application response time*: is the maximum response time of all its services $s_i \in \mathcal{S}$ placed on devices $d_j = \mu(s_i)$:

$$RT_A = \max_{s_i \in \mathcal{S}} \{RT_{ij}\}.$$

6) *Deadline fulfilment*: requires that the response time of the application placement satisfies the deadline θ : $RT_A < \theta$.

IV. FOG MULTILAYER PARTITIONING

This section describes the multilayer Fog partitioning architecture, its design phases and the corresponding algorithm.

A. Architecture design

Figure 1 depicts the architecture design for the Fog multilayer resource partitioning in five phases: resource extraction, multilayer generation, layer partitioning, graph compression and feature partitioning.

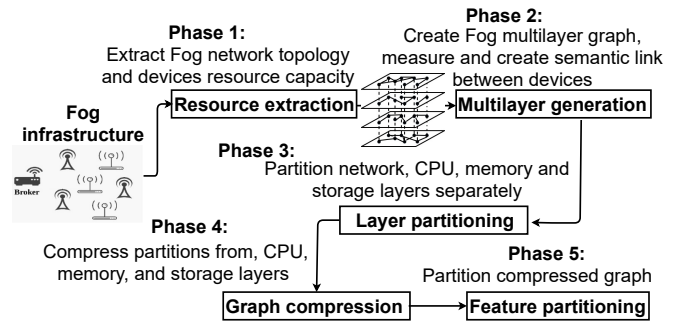


Fig. 1: Fog multilayer partitioning architecture workflow.

1) *Resource extraction*: identifies the infrastructure characteristics based upon different resource types (e.g. R_{i1}, R_{i2}, R_{i3} representing speed of CPU cores in MIPS, memory size, and storage size) and reveals complete information about the Fog platform required for the correct application placement.

2) *Multilayer graph generation*: models the Fog infrastructure as a multilayer graph in three steps:

a) *Placement of Fog devices*: in four layers based on their topological and semantic features.

b) *Similarity score between Fog devices*: based on different resource types.

c) *Intra-layer Fog device linking*: based on topological features and similarity scores, as defined in Section III-C.

3) *Layer partitioning*: splits each Fog multilayer graph layer $l \in L$ in a set $\mathcal{P}(l)$ of disjoint partitions that cluster the Fog devices based on their resource types (see Section III-C).

(i) *Network layer l_0 partitioning* clusters the highly interconnected Fog devices based on their network connections N .

(ii)–(iv) *CPU l_1 , memory l_2 , and storage l_3 layer partitioning* cluster Fog devices with similar CPU speed (R_{i1}), memory size (R_{i2}), and storage size (R_{i3}).

4) *Graph compression*: groups the disjoint CPU, memory and storage layer partitions in a high-level compressed graph representation associated to a similar resource type.

5) *Feature partitioning*: splits the compressed graph in disjoint partitions such that each feature partition is a cluster of similar Fog devices across overlapping resource types.

B. Modularity background

We use the *modularity* [15] metric $Q \in [-1, 1]$ to measure the connectivity strength of Fog multilayer graph partitions:

$$Q = \frac{1}{2W} \cdot \sum_{l \in L} \sum_{l' \in L} \sum_{d_i \in l} \sum_{d_j \in l'} \left\{ \left(w_{ij}^{(l)} - \frac{\sigma_i^l \cdot \sigma_j^l}{2W_l} \right) \cdot \delta_{ll'} + \Delta_{ij} \cdot E_{ll'}^j \right\} \cdot \lambda_{ij}, \text{ where :}$$

• $\sigma_i^{(l)} = \sum_{d_j \in l} w_{ij}^{(l)}$ is the connectivity strength of the Fog device d_i with the other devices in layer l ;

• $\sigma_j^{(l)} = \sum_{d_i \in l} w_{ij}^{(l)}$ is the connectivity strength of the Fog device d_j with the other devices in layer l ;

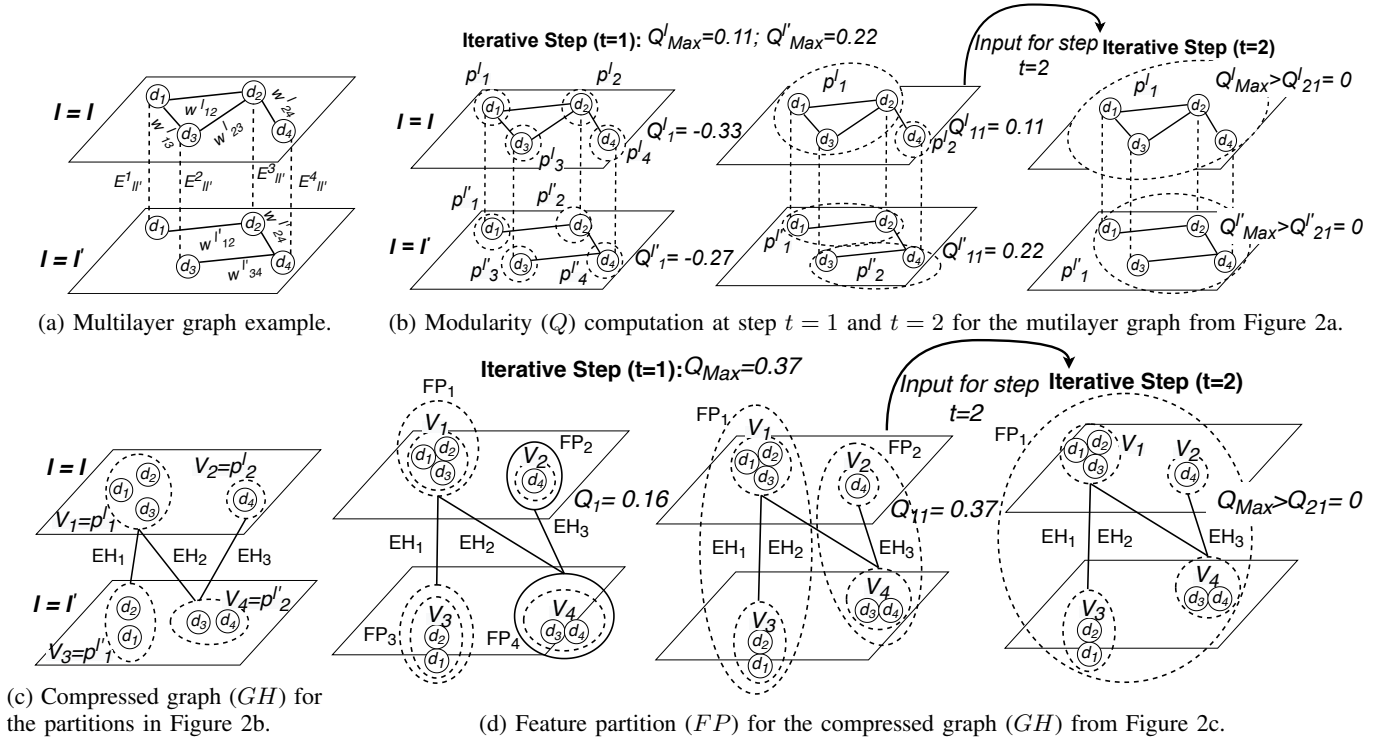


Fig. 2: Fog multilayer graph, modularity computation, and Fog multilayer partitioning example.

- $W_l = \sum_{i=0}^D \sum_{j=0}^D w_{ij}^{(l)}$ is the total sum of the link weights between D Fog devices in each layer $l \in L$;
- $W = \sum_{l \in L} W_l$ is the total sum of the link weights between Fog devices, $\forall l \in L$;
- $E_{ll'}^{(j)}$ indicates the number of interlayer edges of the Fog device d_j from layer l to layer l' ;
- $\delta_{ll'}$ is equal to 1 if $l = l'$ and 0 otherwise;
- Δ_{ij} is equal to 1 if $i = j$ and 0 otherwise;
- λ_{ij} is equal to 1 if device d_i and d_j belong to the same partition, otherwise 0.

a) $Q \leq 0$: represents low quality partitions of disassortative Fog devices with sparse connections among them.

b) $Q > 0$: represents high quality topological partitions with better connectivity strength among densely-connected Fog devices. Hence, the goal is to find a set of partitions in a multilayer graph with highest modularity ($Q_G \rightarrow 1$).

C. Layer partitioning

We apply the Louvain clustering technique [16] that utilizes the modularity metric [17] to obtain high quality partitions with densely connected Fog devices in each layer. We define the modularity of the obtained partitions as the difference between the number of edges within partitions and the expected number of edges over all pair of Fog devices. The Louvain algorithm applies two phases in each iterative step $t > 0$ until achieving partitions with the maximum modularity.

1) *Phase 1*: considers first each Fog device d_i in layer l as a single partition and calculates its modularity Q_i . Afterwards, it

considers all neighbouring devices d_j of d_i (i.e. $(d_i, d_j) \in E_{ll}$ and $(d_j, d_i) \in E_{ll}$) and calculates the modularity of the new possible partition Q_{ij} . If the gain in modularity is positive (i.e. $Q_{ij} - Q_i > 0$), we place d_i and d_j in the same partition. We repeat this step sequentially for all Fog devices in layer l until no further modularity gain is possible.

2) *Phase 2*: takes the set of partitions in each layer l from the first phase and considers each partition as a node to build a new network in each layer. The links between the Fog devices in the same partition represent self-loops, while the links between Fog devices across partitions denote edges between nodes. The new network and the maximum modularity act as input to the next iterative step $t + 1$ starting with phase 1.

3) *Example*: Figure 2a shows a weighted multilayer graph $G = (D, \mathcal{E}, L)$ with two layers $L = \{l, l'\}$. Each layer has four devices $D = \{d_1, d_2, d_3, d_4\}$ and the following intra-layer sets of edges: $E_{ll} = \{(d_1, d_2), (d_1, d_3), (d_2, d_3)\}$ and $E_{l'l'} = \{(d_1, d_2), (d_3, d_4)\}$. The Louvain technique requires two iterative steps to find the partitions with the maximum modularity in each layer in this example.

a) *Step $t = 1$* : shown in Figure 2b creates first a set of four partitions $\mathcal{P}(l) = \{p_1, p_2, p_3, p_4\}$ in layer l and another $\mathcal{P}(l') = \{p'_1, p'_2, p'_3, p'_4\}$ in layer l' . Each partition in $\mathcal{P}(l)$ and $\mathcal{P}(l')$ consists of a single Fog device (d_1, d_2, d_3 and d_4) with the modularities $Q_{\max} = -0.33$ and $Q'_{\max} = -0.27$. Afterwards, it considers all neighboring devices to obtain two new partition sets $\mathcal{P}(l) = \{p_1, p_2\}$ and $\mathcal{P}(l') = \{p'_1, p'_2\}$ with a positive modularity gain (i.e. $Q_1 = 0.11 > Q_{\max}$, $Q'_{11} = 0.22 > Q'_{\max}$), indicating a better connectivity strength

of the devices in each layered partition. We therefore select the partitions $\mathcal{P}(l)$ and $\mathcal{P}(l')$, update the maximum modularities (i.e. $Q_{\max} = Q_1$, $Q'_{\max} = Q'_1$), and consider the partitions in p_1, p_2, p'_1, p'_2 as nodes in a new network over the two layers.

b) *Step $t = 2$* : displayed in Figure 2b starts from the partitions p_1, p_2, p'_1, p'_2 with the maximum modularities Q_{\max} and Q'_{\max} obtained in the step $t = 1$, and considers the neighboring nodes to obtain the single partitions in each layer: $\mathcal{P}(l) = \{p_1\}$ and $\mathcal{P}(l') = \{p'_1\}$ with the modularities $Q_2 = Q'_2 = 0$. As the maximum modularities from the step $t = 1$ are positive for both layers, we consider them as the highly connected topological output.

D. Graph compression

Graph compression shrinks the disjoint partitions from the CPU, memory and storage layers and provides a high-level intermediate representation of partitions associated to similar resource types. The compressed graph merges all the similar resources inside a partition in a single node with an average capacity. This automates the computation of overlapping partitions without detailed analysis of individual Fog devices.

A *compressed graph* $G_P = (V_P, E_P)$ corresponding to a multilayer graph $G = (D, \mathcal{E}, L)$ consists of two sets:

a) *Layer partition set*: is the union of the partitions in the CPU (l_1), memory (l_2), and storage (l_3) layers: $V_P = \bigcup_{l \in L} \mathcal{P}(l)$.

b) *Inter-layer partition edges*: represent connections between a partition p in layer l and a partition p' in layer $l' \neq l$, such that there is at least one inter-layer edge $(d_i, d_j) \in E_{ll'}$ in the original graph G between a device $d_i \in p \in \mathcal{P}(l)$ and a device $d_j \in p' \in \mathcal{P}(l')$:

$$E_P = \{(p, p') \in \mathcal{P}(l) \times \mathcal{P}(l') \mid \forall l \neq l' \in L \\ \wedge \exists (d_i, d_j) \in E_{ll'} \wedge d_i \in p \wedge d_j \in p'\}.$$

1) *Example*: Figure 2c illustrates a compressed graph that represents the four partitions in the l and l' layers in Figure 2b as nodes: $V_P = \{p_1, p_2, p'_1, p'_2\}$. Similar to the nodes, we compress the edges between two partitions into: $E_P = \{(p_1, p'_1), (p_1, p'_2), (p_2, p'_2)\}$.

E. Feature partitioning

We define a *feature* of a layer partition $p \in V_P$ as a triplet $F_p = (R_{p1}, R_{p2}, R_{p3})$ with average CPU speed, memory and storage sizes across all Fog devices in p .

Feature partitioning splits a compressed graph $G_P = (V_P, E_P)$ in a set $\mathcal{P}(G_P)$ of disjoint *feature partitions* (exhibiting similar features) by applying t iterative steps of the Louvain clustering technique as in the layer partitioning to achieve a maximum modularity (see Section IV-C).

1) *Example*: Figure 2d shows the two iterative steps of feature partitioning of the compressed graph G_P in Figure 2c.

a) *Step 1*: initially creates a set of four feature partitions $\mathcal{P}(G_P) = \{FP_1, FP_2, FP_3, FP_4\}$ with the modularity $Q_{\max} = 0.16$, where each feature partition FP_k consists of a single layer partition (i.e. p_1, p_2, p'_3 and p'_4). Afterwards, it considers neighbouring partitions to obtain a new feature

partition set $\mathcal{P}(G_P) = \{FP_1, FP_2\}$ with a positive modularity gain $Q = 0.37 > Q_{\max}$, which becomes the maximum modularity $Q_{\max} = Q$ at this step.

b) *Step 2*: starts from the feature partition set $\mathcal{P}(G_P)$ with the maximum modularity from the first step and considers each feature partition $FP_k \in \mathcal{P}(G_P)$ as a node of the new network (similar to layer partitioning). Afterwards, it iteratively checks the neighbouring feature partitions of each feature partition $FP_k \in \mathcal{P}(G_P)$ and obtains a single partition FP_1 with a lower modularity $Q = 0$. Hence, we select the feature partition set $\mathcal{P}(G_P) = \{FP_1, FP_2\}$ with the maximum modularity from the first step as the final output.

F. Multilayer resource partitioning algorithm

The *multilayer resource partitioning algorithm* clusters Fog devices based on their network connections, CPU, memory, and storage resource characteristics. Algorithm 1 receives a Fog multilayer graph with four layers L , a set of Fog devices D and their underlying CPU, memory, and storage resources, and the inter- and intra-layer edges \mathcal{E} as input. Initially, lines 1–2 initialize five empty sets corresponding to the partitions in the network (l_0), CPU (l_1), memory (l_2) and storage (l_3) layers, as well as the feature partitions. Thereafter, line 3 performs the network layer partitioning $\mathcal{P}(l_0)$ that clusters densely connected Fog devices in the same partition. Similarly, lines 4–6 partition the Fog devices in the CPU, memory, and storage layers and store them in $\mathcal{P}(l_1)$, $\mathcal{P}(l_2)$, and $\mathcal{P}(l_3)$ (see Section IV-C). Line 7 creates a compressed graph $G_P(V_P, E_P)$ using inter-layer edges between the CPU, memory, and storage layers, where $V_P = \{\mathcal{P}(l_1), \mathcal{P}(l_2), \mathcal{P}(l_3)\}$ (see Section IV-D). Afterwards, lines 9–13 compute the feature triplet F_P of each partition $p \in V_P$ as the average CPU speed, memory, and storage sizes of their Fog devices $d_i \in p$. Line 14 performs feature partitioning of the compressed graph partitions with similar features, as presented in Section IV-E. Finally, line 15 returns the feature partition set $\mathcal{P}(G_P)$ and the set of partitions in the network layer $\mathcal{P}(l_0)$.

V. MULTILAYER FOG APPLICATION PLACEMENT

This section describes the application placement workflow in multilayer Fog partitions, its design, and the underlying feature partition selection and service placement algorithms.

A. Multilayer Fog placement workflow

The multilayer Fog placement of application requests $A = (\mathcal{S}, M, \theta, u)$ coming from end-users has two phases.

a) *Feature partition selection*: maps each service $s \in \mathcal{S}$ of the requested application to an appropriate feature partition FP_k composed of layer partitions $p \in V_P$ with the feature F_p similar to the resource demand of the service s .

b) *Service placement*: allocates a Fog device $d = \mu(s)$ to each service $s \in \mathcal{S}$ in the selected feature partitions, such that its selected Fog devices exist in the same network layer partition. This enables placing interrelated services of the same application across highly connected Fog devices.

Algorithm 1: Multilayer resource partitioning.

Input : $G = (D, \mathcal{E}, L)$: Fog multilayer graph
 $L = \{l_0, l_1, l_2, l_3\}$: Fog layers
 $D = \{d_i | d_i = (R_{i1}, R_{i2}, R_{i3})\}$: set of Fog devices
 $\mathcal{E} = \{E_{l,l'} | \forall l, l' \in [0, L]\}$: inter- and intra-layer edges
Output: $\mathcal{P}(G_P)$: feature partition set
 $\mathcal{P}(l_0)$: network layer partition set

```
1  $\mathcal{P}(l_0) \leftarrow \emptyset$ ;  $\mathcal{P}(l_1) \leftarrow \emptyset$ ;  $\mathcal{P}(l_2) \leftarrow \emptyset$ ;  $\mathcal{P}(l_3) \leftarrow \emptyset$ 
2  $\mathcal{P}(G_P) \leftarrow \emptyset$ 
3  $\mathcal{P}(l_0) \leftarrow \text{layerPartition}(D, E_{l_0l_0}, l_0)$ 
4  $\mathcal{P}(l_1) \leftarrow \text{layerPartition}(D, E_{l_1l_1}, l_1)$ 
5  $\mathcal{P}(l_2) \leftarrow \text{layerPartition}(D, E_{l_2l_2}, l_2)$ 
6  $\mathcal{P}(l_3) \leftarrow \text{layerPartition}(D, E_{l_3l_3}, l_3)$ 
7  $G(V_P, E_P) \leftarrow \text{graphCompress}(\mathcal{P}(l_1), \mathcal{P}(l_2), \mathcal{P}(l_3), E_{l_1l_2}, E_{l_1l_3}, E_{l_2l_3})$ 
8  $fList \leftarrow \emptyset$ ;
9 forall  $p \in V_P$  do
10    $R_{p1} \leftarrow \text{avg}_{\forall d_i \in p} \{R_{i1}\}$ ;  $R_{p2} \leftarrow \text{avg}_{\forall d_i \in p} \{R_{i2}\}$ ;  $R_{p3} \leftarrow \text{avg}_{\forall d_i \in p} \{R_{i3}\}$ 
11    $F_p \leftarrow (R_{p1}, R_{p2}, R_{p3})$ 
12    $fList \leftarrow fList \cup F_p$ 
13 end
14  $\mathcal{P}(G_P) \leftarrow \text{featurePartition}(G_P(V_P, E_P), fList)$ 
15 return  $(\mathcal{P}(G_P), \mathcal{P}(l_0))$ 
```

B. Service fitness

An application $A = (\mathcal{S}, M, \theta, u)$ has a set of resource requirements for each service $s_i = (r_{i1}, r_{i2}, r_{i3})$ in terms of CPU speed, memory size and storage capacity required for a successful execution. Hence, it is imperative to place each service $s_i \in \mathcal{S}$ across the Fog devices of a feature partition $d \in FP_k$ composed of layer partitions $p \in FP_k$ with the feature F_p that satisfies these requirements. Additionally, the service placement requires the Fog devices in close proximity of the user location u to simultaneously satisfy application latency and deadline constraints θ .

We define the *fitness* of a service $s_i \in \mathcal{S}$ for a feature partition FP_k requested by an user u as follows:

$$Fit(FP_k, s_i, u) = \alpha \cdot \max_{\forall p \in FP_k} \{Sim(p, s_i)\} + \beta \cdot \left(\frac{1}{1 + \min_{\forall d_j \in FP_k} \{T_{uj}\}} \right), \text{ where :}$$

a) $\max_{\forall p \in FP_k} \{Sim(p, s_i)\}$: is the maximum similarity between the partition with the feature $F_p = (R_{p1}, R_{p2}, R_{p3})$ in feature partition FP_k and the resource demand of the service $s_i = (r_{i1}, r_{i2}, r_{i3})$, computed using the Euclidean distance between the feature F_p and the resource demand s_i in all three dimensions, normalized in the $[0, 1]$ interval;

b) $\min_{\forall d_j \in FP_k} \{T_{uj}\}$: is the minimum transmission time of a message M_{uk} of size SZ_{uk} between the source (user) u and the destination Fog devices d_j in the feature partition FP_k ;

c) α, β : are weighting factors for the similarity and transmission times in the service fitness calculation.

C. Feature partition selection algorithm

The *feature partition selection algorithm* places the services of requested applications to appropriate feature partitions composed of Fog devices that satisfy the application deadline θ and individual service resource demands.

Algorithm 2: Feature partition selection.

Input : $AS = \{A | A = (\mathcal{S}, M, \theta, u)\}$: set of applications;
 $D = \{d_i | d_i = (R_{i1}, R_{i2}, R_{i3})\}$: set of Fog devices;
 $\mathcal{P}(G_P)$: feature partition set;
 $\mathcal{P}(l_0)$: network layer partition set;
 $\mathcal{T} = \{T_{ui} | (u, d_i) \in \mathcal{U} \times D\}$: message M_{ui} transmission times;
Output: $\mu List[AS]$: array of $\mu List[A]$ service placements, $\forall A \in AS$;

```
1  $AS \leftarrow \text{sortApp}(AS, \theta)$ 
2 forall  $A = (\mathcal{S}, M, \theta, u) \in AS$  do
3    $\mu List[A] \leftarrow \text{selectFP}(\mathcal{S}, u)$ 
4 end
5 return  $\mu List$ 
6 ;
7 Function  $\text{selectFP}(\mathcal{S}, u)$  :
8   forall  $s \in \mathcal{S}$  do
9      $fpRank \leftarrow \emptyset$ 
10    forall  $FP \in \mathcal{P}(G_P)$  do
11       $fpRank \leftarrow \text{insert}(fpRank, Fit(FP, s, u))$ 
12       $dMatrix[FP] \leftarrow \text{sortDev}(FP, \mathcal{T})$ 
13    end
14     $\mu(s) \leftarrow \text{placeService}(P(l_0), fpRank, dMatrix, s)$ 
15  end
16 return  $\mu$ 
```

Algorithm 2 takes as input a set of requested applications AS (including their services, resource demands and deadlines), a set of Fog devices D , the feature partition set $\mathcal{P}(G_P)$, the network partition set $P(l_0)$ (both computed by Algorithm 1) and the message transmission times \mathcal{T} between user and Fog devices. Firstly, line 1 sorts the applications to prioritize the placement of those with the lowest deadline. Lines 2–4 select the appropriate feature partitions for all applications $A \in AS$ by placing all their services $S \in A$ on Fog devices that satisfy their resource demands in the proximity of the requesting users. The algorithm returns an array of placement functions $\mu List[AS]$ in line 3. $\forall A \in AS$.

To select the feature partition (line 7) that satisfies the resource demand of each service $s_i \in \mathcal{S}$ (line 8), lines 10–13 iterate through each feature partition $FP \in \mathcal{P}(G_P)$, calculate its fitness to service $s \in \mathcal{S}$ (see Section V-B), and insert it in a $fpRank$ list in descending fitness order (line 11). Line 12 sorts the Fog devices $d \in FP$ in ascending order based on the transmission time T_{ui} and stores them in a two dimensional array $dMatrix$ to enable their placement in user proximity. Line 14 invokes a service placement function (see Algorithm 3) that maps each service $s \in \mathcal{S}$ of the same application across the selected feature partitions with Fog devices in the same network layer partition. The function returns the Fog device $\mu(s)$, which updates the application placement function μ returned by the algorithm in line 16.

D. Service placement algorithm

Algorithm 3 places a service s on a Fog device $\mu(s)$ in the same network layer partition as all other services of the same application. The input arguments to Algorithm 3 are 1) a network layer partition set $P(l_0)$, 2) a feature partition set $\mathcal{P}(G_P)$ ranked based on fitness, 3) a sorted list of Fog devices based on their transmission time in each feature partition $FP \in \mathcal{P}(G_P)$, and 4) the placement service s_i (line 1). First, lines 2–3 extract the network layer partition of the first service placement $\mu(s_1)$ in p_1 (if available). Lines 4–12 iterate through all the feature partitions $FP \in \mathcal{P}(G_P)$ in descending

Algorithm 3: Service placement.

Input: $\mathcal{P}(l_0)$: network layer partition set
 $\mathcal{P}(G_P)$: fitness-ranked feature partition set
 $dMatrix[\mathcal{P}(G_P)]$: sorted Fog devices, $\forall FP \in \mathcal{P}(G_P)$
 $s_i = \{r_{i1}, r_{i2}, r_{i3}\}$: service to place

```

1 Function placeService( $\mathcal{P}(l_0), \mathcal{P}(G_P), dMatrix, s_i$ ):
2   if  $s_i \neq s_1$  then
3      $p_1 \leftarrow \text{getNetPartition}(\mu(s_1), \mathcal{P}(l_0))$ 
4     forall  $FP \in \mathcal{P}(G_P)$  do
5        $dList \leftarrow dMatrix[FP]$ 
6       forall  $d_j = (R_{j1}, R_{j2}, R_{j3}) \in dList$  do
7          $p_i \leftarrow \text{getNetPartition}(d, \mathcal{P}(l_0))$ 
8         if  $p_1 = p_i \wedge \frac{r_{i1}}{R_{j1}} \leq \theta \wedge r_{i2} \leq R_{j2} \wedge r_{i3} \leq R_{j3}$  then
9            $\mu(s_i) \leftarrow d_j$ 
10          return  $\mu(s_i)$ 
11        end
12      end
13       $\mu(s_i) \leftarrow \emptyset$ 
14    return  $\mu(s_i)$ 

```

order of their fitness. Afterwards, line 5 extracts the set of Fog devices $dList$ in each feature partition FP sorted by the transmission times to the requesting user. To place the service s_i onto a Fog device, lines 6–11 iterate through each device $d_j \in dList$ and line 7 extracts its network layer partition in p_i . If this partition is the same as p_1 and the device d_j meets the resource constraints of the service s_i , line 9 performs the placement. If no service placement on the same network partition is possible, line 13 assigns an invalid device. Finally, lines 10 and 14 return the service placement result.

VI. EVALUATION

We present first our experimental setup, then analyze the results of our multilayered partitioning method against two related resource [1] and availability-aware [10] methods.

A. Experimental setup

We simulated a Fog infrastructure using the YAFS [18] simulator on an Intel® Core^(TM) i7-8650U server at 1.90 GHz running Ubuntu 18.04 (x86_64) operating system with 16 GB of DDR4 RAM memory. We simulated the Fog infrastructure (e.g. devices, network topology), applications, and clients using similar configuration settings as two previous studies [1], [10] for a fair comparison.

1) *Fog infrastructure*: We simulated a Fog infrastructure as a bidirectional graph based on a Albert-Barbasi random network [19] of 100 devices, where 25 devices with the lowest betweenness centrality [20] represent gateways at the edge of the network. We represented the Cloud data center though an additional device with the highest betweenness centrality computed using the Python `networkx` module. We simulated resource characteristics (e.g. cores, CPU speed, memory, storage) of each Fog device using a uniform random distribution within the range specified in Table Ia. Finally, we configured the bandwidth and latency across the Fog network as specified in Table Ic, similar to previous studies [1], [10].

2) *Applications*: We simulated an application A as a directed graph using the Python `networkx.Gn_Graph` module, where the nodes correspond to the application services. We generated a number of services for each application A in

TABLE I: Experimental setup.

(a) Fog device.		(b) Application.	
Parameters	Range	Parameters	Value
CPU cores	10 to 25	Services	2 to 10
CPU speed	20 MI/s to 60 MI/s	Deadline	300 ms to 50 000 ms
Memory size	10 GB to 25 GB	Memory size	1 GB to 6 GB
Storage size	10 TB to 25 TB	Storage size	1 TB to 6 TB
		Message size	1500 kB to 4500 kB
		Workload	20 MI to 60 MI

(c) Network.		(d) Client.	
Parameters	Value	Parameters	Value
Latency	5 ms	Request rate	1.557 ms
Bandwidth	75 000 B ms ⁻¹		

the range between two and ten, and their resource requirements (i.e. CPU speed, number of cores, memory and storage size) using a uniform random distribution, as described in Table Ib. Additionally, we defined the dependencies between two services in terms of request messages of sizes between 1500 kB to 4500 kB, which generated a workload in the destination service in the range 20 MI to 60 MI.

3) *Client*: We configured the client layer similar to previous studies [1], [10] such that end-users connected to gateway devices in Fog layer request random applications every 1.557 ms, as specified in Table Id.

4) *Evaluation metrics*: We identified five performance objectives to optimize the multilayer application Fog placement μ for a set of applications AS .

a) *Placement success rate*: is the ratio between the number of successfully placed services \mathcal{S}_μ and the total number of services \mathcal{S} of all applications $A = (\mathcal{S}, M, \theta, u) \in AS$: $\frac{|\mathcal{S}_\mu|}{|\mathcal{S}|}$, where $\mathcal{S}_\mu = \{s \in A | \mu(s) \neq \emptyset\}$ and $|\mathcal{S}|$ and $|\mathcal{S}_\mu|$ represent the cardinality of the two service sets.

b) *Resource wastage*: is the remaining percentage between total resource units consumed by the placed services to the total CPU, memory and disk resource units of the devices:

$$1 - \frac{\sum_{A \in AS} \sum_{s_i \in \mathcal{S}_\mu} \max \left\{ \frac{r_{i1}}{CPU}, \frac{r_{i2}}{RAM}, \frac{r_{i3}}{Disk} \right\}}{\sum_{d_j \in D} \max \left\{ \frac{R_{j1}}{CPU}, \frac{R_{j2}}{RAM}, \frac{R_{j3}}{Disk} \right\}}.$$

We define a *resource unit* as a triplet: $(CPUcore, RAM, Disk) = (1, 1 \text{ GB}, 1 \text{ TB})$. The resource units consumed by a service or device is the maximum of these three unit components.

c) *Deadline satisfaction*: represents the ratio between the total number of applications that fulfill their deadline $A_\theta = \{A \in AS | RT_A < \theta\}$ and the total number of applications $|AS|$ requested for placement: $\frac{|A_\theta|}{|AS|}$.

d) *Hop distance*: indicates the proximity of the placed services to the requesting users. A hop distance of 0 indicates a service placement at the Fog gateway device. We compute a hop distance histogram across all application services, aiming to increase the number of placements at a low hop distance.

5) *Evaluation scenarios*: Similar to the study in [10], we divide our evaluation into two parts, summarized in Table II.

TABLE II: Evaluation scenarios.

Evaluation	Scenario	App.	Service	Users	App. Requests	Service Requests
Service placement	SMALL	10	63	29	29	204
	MEDIUM	20	129	65	65	440
	LARGE	30	179	98	98	537
Deadline fulfilment	D-SMALL	10	63	29	84429	573458
	D-MEDIUM	20	129	65	161597	933944
	D-LARGE	30	179	98	216038	1163571

a) *Service placement*: evaluates the placement success rate, resource wastage and hop distance for three placement scenarios (i.e. SMALL, MEDIUM, and LARGE) with different randomly generated application sets, number of services, users, application requests and corresponding service requests. Every user randomly requests one application for placement.

b) *Deadline satisfaction*: evaluates three scenarios (i.e. D-SMALL, D-MEDIUM, D-LARGE) in a reliable and faulty Fog infrastructure across a simulation period of 2000 s. We randomly introduced failures across the Fog devices every 20 s in the faulty simulation, similar to [10]. All users select a random application every 1.557 s until the complete simulation.

6) *Related work*: We compare our multilayer resource-aware partitioning approach with two state-of-the-art methods:

a) *Availability-aware placement*: with improved application deadline satisfaction in presence of device failures [10].

b) *Resource-aware placement*: with optimized resource usage and reduced latency and application response time by applying fractional selectivity model [1].

B. Service placement

Figure 3 shows the placement success rate for the three evaluation scenarios.

a) *SMALL*: The multilayer and availability-aware methods performed similarly by placing 200 and 202 services with a similar success rate of 0.98 and 0.99, respectively. The resource-aware approach placed only 161 services with a lower success rate of 0.78.

b) *MEDIUM* and *LARGE*: The availability-aware approach placed 221 and 241 services with a success ratio of 0.50 and 0.44, while the resource-aware approach placed only 188 and 172 services with a low placement ratio of 0.42 and 0.32, respectively. In contrast, the multilayer method outperformed them by placing 419 and 407 services with 0.95, respectively 0.75 placement success rates.

c) *Summary*: The multilayer method is able to place more services with the increasing number of requests by mapping their individual resource demands to unique Fog device resource characteristics (i.e. CPU, RAM, storage). In contrast, the availability-aware approach prioritizes a set of

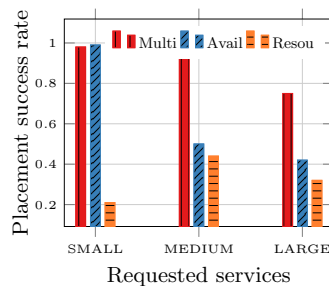
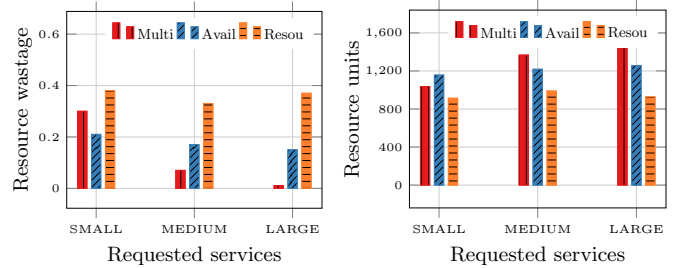


Fig. 3: Service placement.



(a) Resource wastage.

(b) Resource unit consumption.

Fig. 4: Resource wastage.

dependent services by placing them onto the same Fog device if it satisfies their joint resource demands only. While resource-aware approach place dependent services across different Fog devices if they satisfy their joint bandwidth requirements.

C. Resource wastage

Figure 4a compares the resource wastage ratio for the multilayer, availability-aware and resource-aware placement approaches across the simulated Fog infrastructure (100 Fog devices) with 1483 resource units. Figure 4b explains the results through the resource units consumed by each placement.

a) *SMALL*: The multilayer method placed 200 services with a resource wastage of 0.30, while the availability-aware approach performed slightly better by placing 202 services with a resource wastage of 0.21. The resource-aware approach performed the worst by placing only 161 services with a high resource wastage of 0.38. The multilayer and availability-aware methods with similar placement success rate consumed 1036, respectively 1157 resource units. In comparison, the resource-aware approach consumed only 913 resource units for a low placement success rate.

b) *MEDIUM*: The availability-aware and resource-aware approaches placed 221 services with a resource wastage of 0.17 and 0.33, respectively. The multilayer method outperformed them and placed 419 services with a low resource wastage of 0.07. The multilayer method consumed 1367 resource units with a higher placement success rate compared to the availability-aware and resource-aware approaches consuming only 1218, respectively 989 resource units.

c) *LARGE*: Similarly, the multilayer method performed the best by placing 407 services across 100 devices with a very low resource wastage of 0.011. In contrast, the availability and resource-aware approaches placed 241 and 172 services with a high resource wastage of 0.15, respectively 0.37. The multilayer method consumed 1466 resource units with increasing requests due to its higher placement success rate. In contrast, the availability-aware and resource-aware approaches consumed only 1257, respectively 928 resource units.

d) *Summary*: The multilayer maximizes the placement success rate for increasing requests and thus, consume more resource units of the Fog infrastructure, and reduces the resource wastage upto 15 times compared to availability-aware and upto 32 times to resource-aware method.

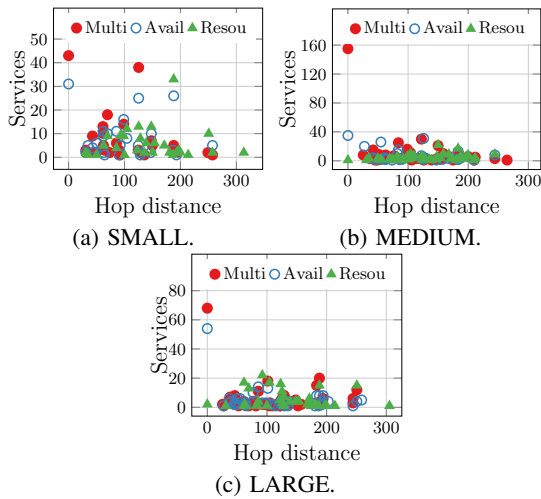


Fig. 5: Hop distance.

D. Hop distance

Figure 5 compares the three approaches based on the hop distance from the user to the placed services.

a) *SMALL*: Figure 5a shows that the multilayer and resource-aware approaches placed 43, respectively 31 services at the zero hop distance, while the availability-aware approach did not place any services at the zero hop distance. Additionally, the multilayer method placed only one service at the maximum hop distance of 257, while the availability-aware approach placed five services at a maximum hop distance of 257. In contrast, the resource-aware approach performed worst and placed two services at a maximum hop distance of 313.

b) *MEDIUM*: Figure 5b shows that the multilayer method placed 155 services at the zero hop distance. In comparison, both resource and availability-aware approaches placed only 35, respectively 1 services at zero hop distance. Moreover, the multilayer method only places one service at the maximum hop distance of 264, the availability-aware and resource-aware placed 8, and 9 services at a maximum hop distance of 244, respectively.

c) *LARGE*: Figure 5c shows that the multilayer method performed again the best and placed 68 services at zero hop distance and used a maximum hop distance of 250. In contrast, the availability-aware and resource-aware approaches placed 54 and 2 services at the zero hop distance, and used a maximum hop distance of 258 and 303, respectively.

d) *Summary*: We conclude that the multilayer method places a larger number of services across the Fog devices in close proximity to users compared to the related methods. The advantage comes from considering the transmission time between users and the Fog devices and by placing the requested services across highly connected devices.

E. Deadline satisfaction

1) *Reliable Fog infrastructure*: Figure 6a shows the cumulative deadline satisfaction ratio in the three 2000 s long simulation scenarios, specified in Table II. Each user requests

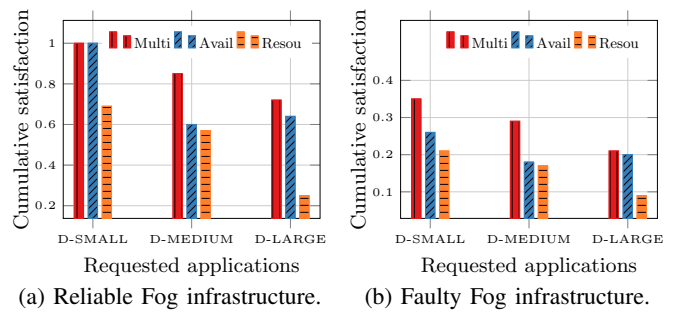


Fig. 6: Deadline satisfaction.

a random applications every 1.557 s (see Table Id) until the complete simulation period.

a) *D-SMALL*: The multilayer and availability-aware methods fulfilled the deadlines of all applications, while the resource-aware approach had a lower cumulative deadline satisfaction rate of 0.69.

b) *D-MEDIUM*: The availability-aware and resource-aware approaches fulfilled deadlines with a cumulative satisfaction rate of 0.60 and 0.57, respectively. The multilayer method outperformed both approaches with a high cumulative deadline satisfaction rate of 0.85.

c) *D-LARGE*: The multilayer method satisfied deadlines with a high rate of 0.72. The availability-aware approach exhibited a lower rate of 0.64, while the resource-aware approach performed worst with a rate of 0.25 only.

d) *Summary*: We conclude with two observations.

(i) The multilayer and availability-aware methods fulfill deadlines with high cumulative satisfaction rate compared to the resource-aware approach that does not consider application deadline as a placement constraint. In contrast, both multilayer and availability-aware methods prioritize applications based on their deadline for optimized placement.

(ii) The multilayer method has a better deadline satisfaction rate with increasing application requests compared to the availability-aware approach due to its ability to place dependent services of the same application across highly connected Fog devices in the same network partitions. This optimizes the latency between dependent services and fulfills deadline for more applications with a better satisfaction rate.

2) *Faulty Fog infrastructure*: Figure 6b evaluates the cumulative deadline satisfaction rate in the three scenarios for faulty Fog infrastructures. Similar to the evaluation of reliable Fog infrastructures, we show the cumulative satisfaction rate in the three 2000 s long simulation scenarios, where users requested random applications every 1.557 s. We introduced faults by randomly failing a Fog device every 20 s, such that all devices are not reachable at the end of simulation period of 2000 s.

a) *D-SMALL*: The availability-aware and resource-aware approaches fulfilled deadlines with a cumulative satisfaction rate of 0.29, respectively 0.21, in the presence of randomly failing Fog devices. The multilayer method performed slightly better with a cumulative satisfaction rate of 0.35.

b) *D-MEDIUM*: All three approaches fulfilled the deadlines with a lower cumulative satisfaction rate than *D-SMALL*. However, the multilayer method performed better with a deadline cumulative satisfaction rate of 0.26. In contrast, both availability-aware and resource-aware approaches exhibited a low deadline satisfaction rate of 0.18 and 0.17, respectively.

c) *D-LARGE*: The resource-aware approach performed worst and fulfilled deadlines with a very low cumulative satisfaction rate of 0.09, while the multilayer and application-aware methods fulfilled deadlines with a better cumulative satisfaction ratio of 0.23 and 0.20, respectively.

d) *Summary*: We draw two observations in Figure 6b.

(i) The resource-aware placement does not consider application deadline or device failures and exhibits poor cumulative deadline satisfaction rate compared to the multilayer and availability-aware methods across faulty Fog infrastructures.

(ii) The multilayer placement achieves a slightly higher cumulative deadline satisfaction rate compared to the availability-aware approach, which places dependent services on the same Fog devices and the others across weakly connected devices prone to failures. In contrast, the multilayer method places all application services across highly connected Fog devices in the same network layer partition and therefore, exhibits better deadline satisfaction rate even upon faults.

VII. CONCLUSIONS

We introduced a new resource-aware method for Fog application placement, which represents heterogeneous Fog devices as a multilayer network graph and partitions them with respect to the network topology and resource characteristics. The new multilayer method places a multi-service application structured as a directed graph in two steps. The first step matches the requested applications with feature partitions based on their requirements, which overlap in the same topological partition. The second step places the services to Fog devices in the elected partitions closest to the end-users. We evaluated the multilayer resource-aware placement against two availability-aware and resource-aware state-of-the-art approaches. The results indicate that our method is able to place twice as many services, satisfy deadlines for three times as many application requests, and reduce the resource wastage by 15 – 32 times compared to the related methods.

We plan to extend our multilayer resource-aware placement method to dynamically consider changing application and Fog infrastructure characteristics.

ACKNOWLEDGEMENTS

a) *European Union's Horizon 2020*: research and innovation programme funded this work through two grants:

- ARTICONF: “Smart Social Media Ecosystem in a Blockchain Federated Environment” (grant 825134);
- DataCloud: “Enabling the Big Data Pipeline Lifecycle on the Computing Continuum” (grant 101016835).

b) *Carinthian Agency for Investment Promotion and Public Shareholding (BABEG)*: supported the use case application and Fog infrastructure simulation.

REFERENCES

- [1] M. Taneja and A. Davy, “Resource aware placement of iot application modules in fog-cloud computing paradigm,” in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 1222–1228.
- [2] C.-H. Hong and B. Varghese, “Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms,” *ACM Computing Survey*, vol. 52, Sep. 2019.
- [3] “IEEE standard for adoption of openfog reference architecture for fog computing,” pp. 1–176, 2018.
- [4] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, “Small cell clustering for efficient distributed fog computing: A multi-user case,” in *IEEE 82nd Vehicular Technology Conference*, 2015, pp. 1–5.
- [5] Y. Sun, F. Lin, and H. Xu, “Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II,” *Wireless Personal Communications*, vol. 102, no. 2, pp. 1369–1385, 2018.
- [6] K. Velasquez, D. P. Abreu, M. Curado, and E. Monteiro, “Service placement for latency reduction in the internet of things,” *Annals of Telecommunications*, vol. 72, pp. 105–115, 2017.
- [7] Z. Huang, K.-J. Lin, S.-Y. Yu, and J. Y. Jen Hsu, “Co-locating services in iot systems to minimize the communication energy cost,” *Journal of Innovation in Digital Ecosystems*, vol. 1, pp. 47–57, 2014.
- [8] H. P. Sajjad, F. Rahimian, and V. Vlassov, “Smart partitioning of geo-distributed resources to improve cloud network performance,” in *IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015, pp. 112–118.
- [9] L. Shoosharian, D. Lan, and A. Taherkordi, “A clustering-based approach to efficient resource allocation in fog computing,” in *International Symposium on Pervasive Systems, Algorithms and Networks*. Springer, 2019, pp. 207–224.
- [10] I. Lera, C. Guerrero, and C. Juiz, “Availability-aware service placement policy in fog computing based on graph partitions,” *IEEE Internet of Things Journal*, vol. 6, pp. 3641–3651, 2019.
- [11] S. Filiposka, A. Mishev, and K. Gilly, “Community-based allocation and migration strategies for fog computing,” in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.
- [12] A. Asensio, X. Masip-Bruin, R. Durán, I. de Miguel, G. Ren, S. Daijavad, and A. Jukan, “Designing an efficient clustering strategy for combined fog-to-cloud scenarios,” *Future Generation Computer Systems*, vol. 109, pp. 392–406, 2020.
- [13] R. K. Naha, S. Garg, A. Chan, and S. K. Battula, “Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment,” *Future Generation Computer Systems*, vol. 104, pp. 131–141, 2020.
- [14] P. Štefanič, P. Kochovski, O. F. Rana, and V. Stankovski, “Quality of service-aware matchmaking for adaptive microservice-based applications,” *Concurrency and Computation: Practice and Experience*, 2020.
- [15] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, “Community structure in time-dependent, multiscale, and multiplex networks,” *Journal of Science*, vol. 328, pp. 876–878, 2010.
- [16] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 10, 2008.
- [17] M. E. J. Newman, “Modularity and community structure in networks,” *Proceedings of the National Academy of Sciences*, vol. 103, pp. 8577–8582, 2006.
- [18] I. Lera, C. Guerrero, and C. Juiz, “YAFS: A simulator for IoT scenarios in fog computing,” *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019.
- [19] A.-L. Barabási, “Scale-free networks: A decade and beyond,” *Journal of Science*, vol. 325, pp. 412–413, 2009.
- [20] L. C. Freeman, “A set of measures of centrality based on betweenness,” *Journal of Sociometry*, vol. 40, pp. 35–41, 1977.