A Dynamic Service Placement in Fog Infrastructure

Mayssa Trabelsi[a], Nadjib Mohamed Mehdi Bendaoud[b] and Samir Ben Ahmed[c]
LIPSIC Laboratory, Faculty of Sciences of Tunis, University of Tunis El Manar, Tunis 2092, Tunisia

Keywords: Internet of Things (IoT), Fog Computing, Service Placement, Real-Time Application, Quality-of-Service (QoS), iFogSim.

Abstract: The Internet of Things (IoT) is a key technology that improves the connectivity between applications and devices over different geographical locations. However, IoT devices, particularly those used for monitoring, have stringent timing requirements that Cloud Computing might not be able to satisfy. Fog computing, which uses fog nodes close to IoT devices, can solve this problem. In this paper, we propose a Dynamic Service Placement (DSP) algorithm for Fog infrastructures. DSP's objective is to dynamically place the services emitted by applications one at a time and in real time on Fog nodes. The algorithm chooses the fog node with the least response time over the infrastructure and dynamically places the incoming service in it. The algorithm is implemented in the iFogSim simulator, and its performances were evaluated and compared to other algorithms. DSP showed very encouraging results, as it proceeded to minimize the average response times and the application placement rate, thus lowering the infrastructure usage and energy consumption.

1 INTRODUCTION

The Internet of Things (IoT) has become a highly important aspect in peoples daily lives and businesses. IoT is the network of smart devices such as sensors, actuators, vehicles, smartphones, and cameras. These devices are interconnected and communicated in order to generate massive amounts of data that need to be stored, processed, analyzed, and represented to obtain valuable information meeting users' needs (Tavousi et al., 2022). However, most IoT devices have limitations in terms of computing power, storage capacity, battery power, and bandwidth. Hence, more powerful devices are required to process and store the services requested by an IoT device and manage the enormous amounts of generated data, such as cloud computing and fog computing (Tavousi et al., 2022) (Azizi et al., 2019).

The Cloud of Things (CoT) refers to integrating IoT with Cloud Computing, which enhances network performance by transferring application services to the cloud data centers to execute and store. Nevertheless, IoT applications' geographically distributed nature differs from the centralized nature of cloud data centers. As IoT devices are usually far away from the cloud data centers, high communication latency and high network bandwidth consumption are inevitable. Hence, cloud computing is not feasible for delay-sensitive IoT applications requiring Real-Time (RT) service. To overcome these limitations (network bandwidth consumption, geographic distribution, high service delay, and privacy-sensitive applications), Cisco proposed a new paradigm called fog computing (Bonomi et al., 2012). The primary purpose of the Fog is to extend the Cloud resources and services at the edge of the network.

Fog computing provides computational and storage resources at the network edge devices closer to the data generation sources; thus, it provides the services for RT IoT applications (Natesha and Guddeti, 2021). It can support latency-critical IoT applications requiring short response times. In fact, using Fog computing can allow for a drastic reduction of the overall network latency (Naas et al., 2017). Furthermore, it improves the quality of service (QoS), such as latency and response time, of IoT applications, particularly for delay-sensitive ones (Khosroabadi et al., 2021). Any device capable of computing, storing, and connecting to the network can be considered a fog node (FN), such as smart gateways, personal computers, switches, routers, and local servers. Fog computing faces new challenges and difficulties that need more research, and attention (Azizi et al., 2019) (Khos-

[a] https://orcid.org/0000-0002-1723-9486
[b] https://orcid.org/0000-0002-1442-6323
[c] https://orcid.org/0000-0002-4642-2108

Trabelsi, M., Bendaoud, N. and Ben Ahmed, S.
A Dynamic Service Placement in Fog Infrastructure.
DOI: 10.5220/0011852400003464
In *Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2023)*, pages 444-452
ISBN: 978-989-758-647-7; ISSN: 2184-4895

roabadi et al., 2021). For example, the placement of services is an optimization problem in fog computing in which services should be placed in the fog infrastructure more efficiently. This problem is referred to as the Service Placement Problem (SPP) (Khosroabadi et al., 2021).

This paper proposes a Dynamic Service Placement algorithm (DSP) in a real-time fog infrastructure. Our algorithm aims to solve the problem of dynamic service placement for latency-critical IoT applications in hierarchical Fog infrastructure in order to achieve high QoS for IoT users. The algorithm is implemented in the iFogSim simulator (Gupta et al., 2017) and compared with random and first-fit policies together with the iFogSim built-in cloud-only policy.

The major contribution of this paper can be summarized as follows.

- A framework for placing services dynamically and in real-time in a hierarchical fog infrastructure is developed.
- An Integer Linear Programming (ILP) formulation is presented for the SPP problem subject to constraints such as application deadlines, requirements, and fog nodes characteristics.
- This paper proposes an efficient heuristic algorithm that dynamically assigns a placement for a service in real-time in the most suitable fog node. The algorithm selects the best fog device that minimizes the response time and meets the deadline of the processed application.
- The novelty of this work lies in the ability of the framework to dynamically process tasks emitted by applications one at a time and in real-time. The algorithm proceeds to optimize the placement of the services dynamically in the fog infrastructure as and when a new task is emitted. The proposed algorithm offers high performances even when the complexity of the process increases with a large number of delay-sensitive applications.

The remainder of the paper is organized as follows. In Section 2, the related works are reviewed. Section 3 describes the system architecture. In Section 4, we formulate the problem and describe our solution. We evaluate the proposed solutions in Section 5 and conclude in Section 6.

2 RELATED WORK

In this section, we review and discuss some recent related works that have been proposed for solving the problem of placing services (SPP) in the fog environment, focusing on their objective functions and solutions to the SPP.

In (Khosroabadi et al., 2021), authors proposed a heuristic algorithm, dubbed as ”a clustering of fog devices and requirement-sensitive services first” (SCATTER), based on fog node clustering to solve the SPP. This algorithm, which has promising results, is based on QoS metrics in terms of application response time, network usage, average application loop delays, and energy consumption. In (Farzin et al., 2022), authors proposed a flexible and scalable platform called FLEX for the SPP in multi-Fog and multi-Cloud environments. The service placement problem is formulated as an optimization problem and solved by the heuristic algorithm with the aim of delay and cost minimization. In (Tran et al., 2019), Tran et al. proposed a novel approach to task placement on fog computing made efficient for IoT applications that can enhance the performance of IoT services in terms of response time, cost, and energy. In (Tavousi et al., 2022), authors developed a fuzzy approach to classify IoT applications based on their characteristics. Furthermore, they proposed a heuristic algorithm to place applications on the virtualized computing resources. Cost and resource usage are the performance metrics for evaluating the proposed approach. The problem is formulated using Mixed Integer Linear Programing (MILP). In (Azizi et al., 2019), authors introduced an efficient heuristic algorithm, called Most Delay-sensitive Application First (MDAF), to solve the SPP in fog-cloud computing environments. The proposed algorithm placed the most delay-sensitive application services closer to the IoT devices. This later work was extended in (Hassan et al., 2020) where the authors proposed a service placement policy, called MinRE, to provide high QoS for IoT services and low energy consumption for fog service providers. In addition, they proposed two heuristic-based algorithms to solve the problem efficiently. The first one tried to provide high QoS for critical services in terms of response time, while the second focused on the fog environment’s energy efficiency. In (Nezami et al., 2021), authors studied two optimization objectives for IoT service placement. They formulated a decentralized global and local load-balancing problem to minimize the cost of deadline violation, service deployment, and unhosted services. In (Natesha and Guddeti, 2021), authors developed a docker and containers-based two-level fog infrastructure to provide the resources. Furthermore, they formulated the service placement problem as a multiobjective optimization problem for minimizing service time, cost, and energy consumption. The multiobjective problem is solved using the Elitism-based Genetic Algorithm (EGA). In

(Herrera et al., 2021), the authors proposed a framework called Umizatou that combines three optimization problems, the Decentralized Computation Distribution Problem (DCDP), the Fog Node Placement Problem (FNPP), and the SDN Controller Placement Problem (CPP) that aimed at minimizing response time. Umizatou is a holistic deployment optimization solution based on Mixed Integer Linear Programming.

In this work, we propose a dynamic service placement algorithm for real-time application in a Fog Infrastructure. The main goal of this algorithm is to minimize the response time and meet the deadline of the processed application. Different from state-of-the-art approaches, the dynamic algorithm that we propose is the better solution for service placement in delay-sensitive IoT applications.

3 SYSTEM ARCHITECTURE

The architecture of the Fog Infrastructure system is shown in Fig. 1. It presents a hierarchical structure. We discuss the components and how they interact with each other below.

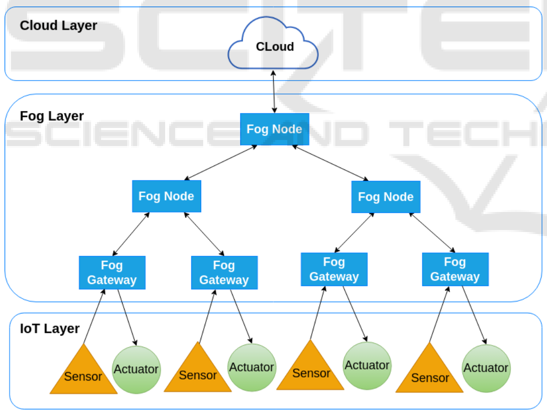

Figure 1: Fog computing architecture.

3.1 IoT Layer

The IoT devices are distributed in different geographical locations and include end-point devices such as sensors and actuators. In IoT devices, the sensors send their service requests to the fog gateway node, which sends them to the fog node for further processing and filtering. The actuators provide the results of service execution for users.

3.2 Fog Layer

The fog layer is the middle layer between IoT devices layer and the cloud layer. Any device with the capability of hosting application modules in the fog layer is referred to as fog node, such as routers, switches, local servers, and base station. These fog nodes are computing nodes that can host virtual machines, given their resource capacity. On each virtual machine, some applications are responsible for running the services requested by IoT devices. The fog gateways are the fog devices that connect sensors to the network, such as Wi-Fi access points, cellular base stations, and home switches, which are located near IoT devices (Gupta et al., 2017).

3.3 Cloud Layer

The cloud environment is a set of large-scale data centers and servers. IoT applications without deadline requirements are placed and run on servers located at cloud data centers. Although the cloud offers good performances on high-demand applications, the communication latency could be very high due to the geographical distances between the cloud and IoT devices.

4 SERVICE PLACEMENT IN FOG INFRASTRUCTURE

Our architecture is inspired by (Naas et al., 2017). This paper formulated a data placement problem and proposed an exact solution using integer programming. The hierarchical fog architecture used in (Naas et al., 2017) is suitable for our suggested dynamic service placement.

In this section, we describe the proposed architecture and formulate the problem of placing services in a hierarchical fog infrastructure.

4.1 Proposed Architecture Description

Our proposed architecture is presented in Fig.2. The sensors are responsible for sending requests to the gateways, which will send the requests to the other fog nodes, depending on the physical connections between the nodes. The choice of the fog node which will receive and execute the request sent by the gateway devices is managed by our proposed framework. The proposed framework consists of a heuristic algorithm called Dynamic Service Placement (DSP).

From a closer perspective, the sensors collect raw data from the environment, creating sensing tasks.

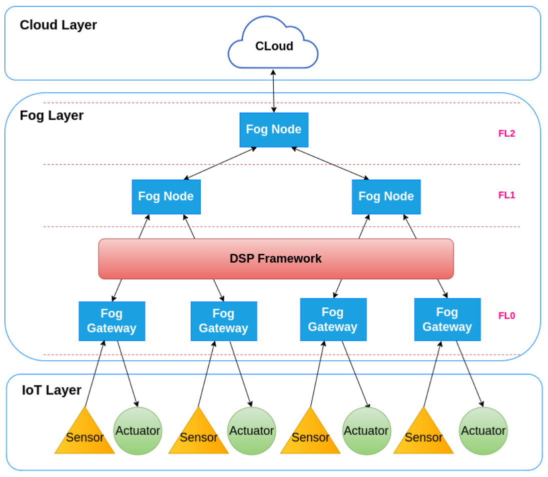

Figure 2: Proposed System Architecture.

The sensor sends the task to the fog node through their connected gateways, to a fog node at a higher level, or to the cloud server. The fog node sends the results back to the actuator specific to the IoT device which initially sent the task, as illustrated in Fig 3.

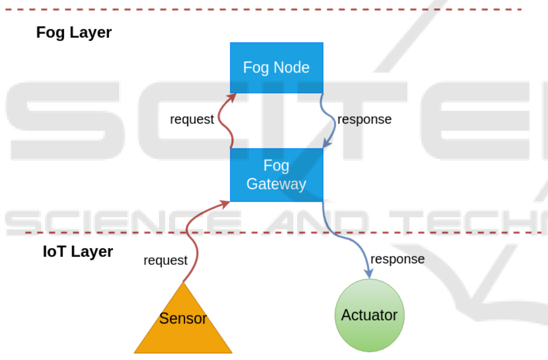

Figure 3: Sensor and Fog Node communication.

The formulation of SPP is discussed in detail within the following subsections.

4.2 Problem Statement

Suppose the total number of all computing devices in fog and cloud layer is m, represented by the set $F = \{F_1, F_2, \dots, F_m\}$. Each Fog device F_j has different resources, such as CPU and RAM. Therefore, F_j^{cpu} and F_j^{ram} represent the capacity of the CPU, measured in a million instructions per second (MIPS), and RAM, measured in Megabytes (MB), of the Fog device, respectively.

Let A represent a set of n IoT applications that can be requested by IoT devices. Each application $A_i \in A$ consists a set of independent tasks $A_i = \{t_{i,1}, t_{i,2}, \dots, t_{i,|A_i|}\}$, where each A_i has a different number of tasks, and $t_{i,h}$ represents the h-th task of application A_i. Each task $t_{i,h}$ requires the number of resources, such as CPU (in a million instruction per second – MIPS) and RAM (in Megabytes – MB), that is denoted as $t_{i,h}^{cpu}$ and $t_{i,h}^{ram}$, respectively. Throughout this paper, the terms service, and task are used interchangeably.

In this work, we attempt to place tasks on computational nodes to reduce application response time. Next, we model the objective functions using ILP.

4.3 Problem Formulation

We consider the binary decision variable $x_{i,h}^j$ to indicate that the task $t_{i,h}$ on which fog device F_j is located.

$$x_{i,h}^j = \begin{cases} 1, & \text{if task } t_{i,h} \text{ is placed on node } F_j \\ & (\forall t_{i,h}, \forall F_j \in \text{F}) \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

Any latency-sensitive application A_i should satisfy the following Eq. (2), where RT_{A_i} is the average response time of application A_i, and D_{A_i} is the deadline for A_i application users determine that.

$$RT_{A_i} \leq D_{A_i}, \forall A_i \in A \quad (2)$$

The response time of application A_i is defined as the time interval between the IoT application request by the respective IoT device and the moment it receives the result (Apat et al., 2022).

The response time of each task of A_i is composed of three parts: communication delay, processing delay, and queuing delay. Therefore, the response time $RT_{t_{i,h}}^j$ of $t_{i,h}$ can be calculated as Eq. (3).

$$RT_{t_{i,h}}^j = TC_{t_{i,h}}^j + TE_{t_{i,h}}^j + TQ_{t_{i,h}}^j, \forall t_{i,h} \quad (3)$$

where, $TC_{t_{i,h}}^j$ is the sum of the two ways communication link delays between the IoT device and the computational node where the task of application A_i is placed. $TE_{t_{i,h}}^j$ is the processing execution delay for $t_{i,h}$. $TQ_{t_{i,h}}^j$ is the queuing delay for $t_{i,h}$.

The average response time of an IoT application can be calculated using Eq. (4). Average response time is obtained by dividing the total response time of an application over the number of tasks $|A_i|$.

$$RT_{A_i} = \frac{1}{|A_i|} \sum_{h=1}^{|A_i|} RT_{t_{i,h}}^j \quad (4)$$

4.4 The Objective Function

Our main goal is to solve the service placement problem in the Fog infrastructure to minimize the response time of the tasks, as stated by the following equation.

$$Min \sum_{i=1}^{n} \sum_{h=1}^{|A_i|} RT_{t_{i,h}}^{j} \quad (5)$$

Our optimization model involves the following five constraints.

$$\sum_{i=1}^{n} \sum_{h=1}^{|A_i|} t_{i,h}^{cpu} \times x_{i,h}^{j} \leq F_j^{cpu}, \forall F_j \in F \quad (6)$$

$$\sum_{i=1}^{n} \sum_{h=1}^{|A_i|} t_{i,h}^{ram} \times x_{i,h}^{j} \leq F_j^{ram}, \forall F_j \in F \quad (7)$$

$$\sum_{j=1}^{m} x_{i,h}^{j} = 1, \forall t_{i,h} \quad (8)$$

$$x_{i,h}^{j} \in \{0,1\} \quad (9)$$

$$RT_{A_i} \leq D_{A_i}, \forall A_i \in A \quad (10)$$

Constraints 6 and 7 indicate that the total CPU and RAM of all tasks placed on each computational node should not exceed their capacity. Constraint 8 ensures that each task is hosted on only one computational node. Constraint 9 specifies the domain of the decision variable. Finally, constraint 10 provides that the deadline for each application must be met.

The proposed model for the dynamic service placement problem is an Integer Linear Programming (ILP) problem. As a result, offering an accurate solution to this problem on a large scale is almost inapplicable (Salaht et al., 2020).

In the next section, we propose an efficient heuristic algorithm for solving this problem.

4.5 Proposed Algorithm

In this work, we describe our proposed heuristic algorithm called Dynamic Service Placement (DSP) in Fog infrastructure. Our framework takes place after fog gateway nodes. Its purpose is to minimize the response time for each task sent from the gateway devices. The main idea of DSP is to calculate and place in real-time the delay-sensitive application services on fog devices with the lowest response time. This ensures that the application deadline is met, hence satisfying the global constraint in Eq. (2). Based on the architecture Fig. 2, each IoT device (which has a sensor and an actuator) is directly connected to a specific fog gateway. Therefore, when a sensor is transmitting a task, it is automatically placed on the corresponding fog gateway. Then, the DSP framework runs the algorithm to find the best host for the task to be placed on. Once a task is deployed successfully, the node uses some of its computation power to execute it. Then it processes the request and sends back the result to the actuator. The pseudocode of the proposed algorithm is presented in Algorithm 1.

Algorithm 1: Dynamic Service Placement algorithm.

```
    Input: $t_{i,h}$ , $FogDevicesList\ F$
    Output: $t_{i,h} \longrightarrow F$
L1:= $\emptyset$ , L2:= $\emptyset$;
for each $F_j \in \mathcal{F}$ do
    if $t_{i,h}^{cpu} \leq F_j^{cpu}$ && $t_{i,h}^{mem} \leq F_j^{ram}$ then
        $L1 := L1 \cup F_j$
    end if
end for
if $L1 \neq \emptyset$ then
    for each $F_j \in L1$ do
        calculate $RT_{t_{i,h}}^{j}$ using eq.(3);
        $L2 := L2 \cup RT_{t_{i,h}}^{j}$ ;
    end for
    place $t_{i,h}$ on $F_j \in L2$ with the minimum $RT_{t_{i,h}}^{j}$ ;
end if
```

The DSP algorithm is explained in the following paragraphs:

The algorithm receives as input a task $t_{i,h}$ from an application A_i and the list of the available computational nodes in the architecture F.

The process begins in line 1 by generating two empty lists, $L1$ and $L2$. Then, for each task emitted by a sensor to a fog gateway, the algorithm will begin by iterating over all the computational nodes F. If a computational node F_j can satisfy the requirements of a task $t_{i,h}$, the node F_j is added to the list $L1$ (line 2-6).

Afterward, if the $L1$ list is not empty, the algorithm will iterate over $L1$. For each computational node F_j in $L1$, the algorithm calculates the response time of task $t_{i,h}$ on that node. The response time $RT_{t_{i,h}}^{j}$ is then added to $L2$ (line 7-11). Finally, the minimum response time is calculated from list $L2$, and task $t_{i,h}$ will be placed on the corresponding node F_j.

5 EVALUATION

To evaluate our dynamic service placement algorithm, it is important to examine its performance according to the QoS criteria, such as application response times, energy consumption, network usage, and the number of service placements at each infrastructure level.

The proposed DSP algorithm has been implemented on the iFogSim simulator (Gupta et al., 2017). In addition, the performance of DSP has been com-

Table 1: Characteristics of Applications.

Application name	Number of instructions (MI)	Memory requirements (MB)	Deadline (ms)	Frequency (ms)
App1	60000	256	4000	50
App2	150000	256	4500	60
App3	350000	256	5000	55
App4	180000	256	5000	70
App5	80000	256	4000	100
App6	150000	256	3500	52
App7	100000	256	4000	65
App8	300000	256	4000	150

pared with other benchmark algorithms such as First-Fit (F.F), Random, and Cloud-Only strategies. The First-Fit strategy (Tavousi et al., 2022) consists of placing the service on a computational node of the first level of the infrastructure; this node has to meet the requirements of the service. Otherwise, the service is sent to a higher level until it finds a computational node that satisfies the service's requirements. If no node in the fog infrastructure can host the service, it will be placed in a data center located in the cloud. On the other hand, the random strategy, as its name suggests, will randomly select a computational node to place a service (Tavousi et al., 2022). Meanwhile, the cloud-only strategy will have all the application services running on cloud data centers (Azizi et al., 2019).

5.1 Experiment Settings

The experiments were performed on a PC Intel Core i7-2670 CPU 2.20 GHz *8, 3.7 GB RAM, and Ubuntu 20.04.5 LTS.

The simulated infrastructure is modeled according to the architecture illustrated in Fig. 2. Eight IoT applications are considered, and each application has one sensor with a distinct periodic frequency $Freq_i$. All the applications A_i have different deadlines D_{A_i}, they emit a task each $Freq_{A_i}$. The required resources for each application are presented in Table 1. Moreover, There are eight fog gateways (Fog Level 0 (FL0)), as each application sensor is connected to a fog gateway. So then, four fog devices are present in the first level (FL1) of the infrastructure, followed by two in level 2 (FL2), and finally, one cloud server.

Furthermore, the computational devices in each level of the infrastructure have their own characteristics, RAM and CPU (see Table 2).

Table 2: Characteristics of computational devices.

Level	Node	Number of node	CPU (MIPS)	RAM (MB)
0	Fog gateway	8	250	256
1	Fog device	4	2800	1000
2	Fog device	2	4000	4000
3	Cloud server	1	44800	40000

Table 3 shows the latency in (ms) between the different devices and nodes, such as the cloud, fog nodes, sensors, and actuators.

Table 3: Nodes communication latencies.

Network link	Latency (ms)
Fog node-cloud	3000
Fog node-Fog node	500
Fog node-Fog Gateway	50
Sensor-Fog Gateway	10
Actuator-Fog Gateway	10

5.2 Analysis of the Results

To evaluate the performance of the service placement algorithm, the difference between the application deadline and response time ($D_{A_i} - RT_{A_i}$) will be determined. The results of algorithms performance based on ($D_{A_i} - RT_{A_i}$) is shown in Table 4. The First-Fit algorithm violates application A2 deadline at 5578.68 ms, application A3 deadline at 12044.15 ms, application A4 deadline at 12262.31 ms, application A7 deadline at 3220.99 ms, and application A8 at 4642.75 ms.

Meanwhile, the cloud-only algorithm violates all the application deadlines in 7183.43, 7192.10, 7190.91, 7183.74, 7184.92, 7185.25, and 7123.42 ms. Also, In the random algorithm, the deadlines for all applications are violated in 10494.98, 9290.20, 9907.91, 9515.52, 10371.82, 9317.8, and 10568.67 ms. However, no application deadline is violated in DSP; this shows how effective the proposed algorithm is compared to other algorithms, as it managed to successfully meet all the application deadlines.

Fig. 4 offers a better illustration of the average response time of all the applications and shows the deadline curve to better measure the performance of the algorithms.

Table 4: Algorithms performance comparison.

	Average Response Time (ms)				$D_{A_i} - RT_{A_i}$ (ms)			
	DSP	Cloud-Only	Random	F.F	DSP	Cloud-Only	Random	F.F
App1	2930.96	7183.43	10494.98	4722.38	1069.04	-3183.43	-6494.98	-722.38
App2	2899.09	7192.1	9290.20	5578.68	1600.91	-2692.1	-7591.61	-4790.2
App3	3798.41	7190.91	9907.91	12044.15	1201,59	-2190.91	-4907.91	-7044.15
App4	3785.32	7183.74	9515.52	12262.31	1352.4	-2993.12	-7029.42	-7323.56
App5	2264.64	7184.92	10371.83	3340.04	1735.36	-3184.92	-6371.83	659.96
App6	2205.33	7185.25	9317.8	3220.99	1294.67	-3685.25	-5817.8	279.01
App7	2151.98	7183.42	10568.67	4642.75	1848.02	-3183.42	-6568.67	-642.75
App8	2309.94	7203.69	9050.45	4885.57	1690.06	-3203.69	-5050.45	-885.57

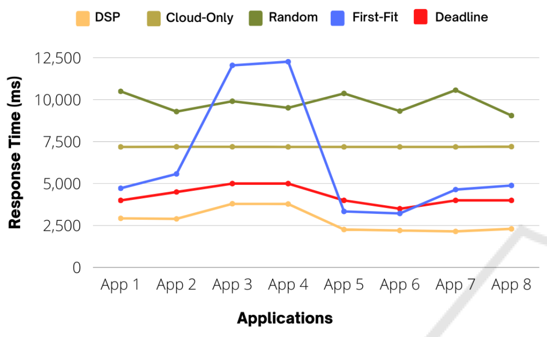

Figure 4: Average Response time of applications.

Fig. 5 shows the number of services placed on each level of the infrastructure. Overall, 1200 tasks have been sent from sensors in the simulation. All of the services are placed on the cloud when using a cloud-only algorithm. Random and first-fit algorithms have placed, respectively, 172 and 50 services in the cloud. This is much higher than DSP, which only placed one service in the cloud. Moreover, DSP has placed 704 tasks in fog level 1 (FL1) and 495 tasks in fog level 2 (FL2), which is higher than both first-fit and random algorithms.

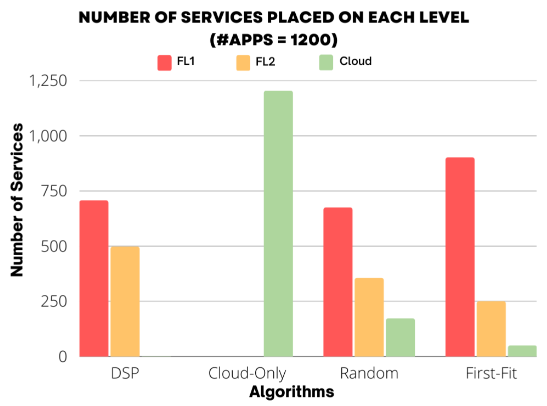

Figure 5: Number of services placed on each level of the infrastructure (# Apps = 1200).

The observation in the previous paragraph is confirmed in Table 5, where a placement rate for each algorithm in the different infrastructure levels is given. We can clearly see that DSP (58.66% FL1 and 41.25% FL2) has placed 4.08% more services in the two first levels (FL1 and FL2) than first-fit (75% FL1 and 20.83% FL2). DSP has also placed 14.25% more services than random (56.08% and 29.58%) in FL1 and FL2 combined. It is also clear that DSP has the least services placed on the cloud (0.08%) compared to random (14.33%) and first-fit (4.16%).

Table 5: Service placement rate per infrastructure level.

	Placement (%)		
Algorithms	FL1	FL2	Cloud
DSP	58.66	41.25	0.08
Cloud-Only	0	0	100
Random	56.08	29.58	14.33
F.F	75	20.83	4.16

Fig. 6 shows clearly that none of the applications have been placed in the cloud by DSP with the exception of Application 4, which has placed 64 tasks in level 1, 77 tasks in level 2 and 1 task in the cloud. The other applications are predominantly placed by DSP in the FL1 and FL2, with a higher placement advantage for FL1 (704 applications placed) over FL2 (495 applications placed).

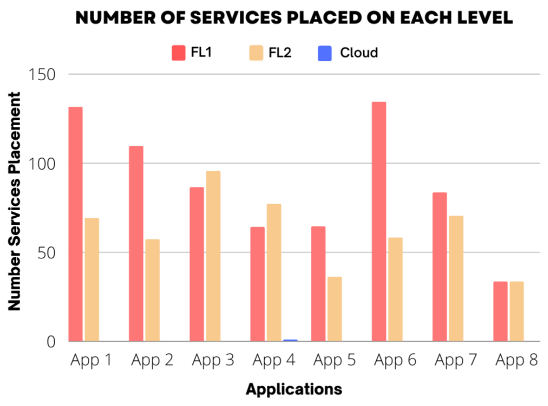

Figure 6: Number of services placement of each application on each level.

Energy consumption is the sum of energy consumed by fog nodes and the cloud for task placement. Fig. 7 presents the total energy consumption for all computational resources in DSP, cloud-only, random, and first-fit algorithms. When using the DSP algorithm, less energy is consumed by fog nodes (FL1 = 4.14 MJ and FL2 = 1.90 MJ) compared to the first fit (FL1 = 4.22 MJ and FL2 = 1.90 MJ) and random (FL1 = 4.09 MJ and FL2 = 2.01 MJ) algorithms. Meanwhile, cloud-only consumption is the lowest in both FL1 and FL2 compared to the other algorithms.

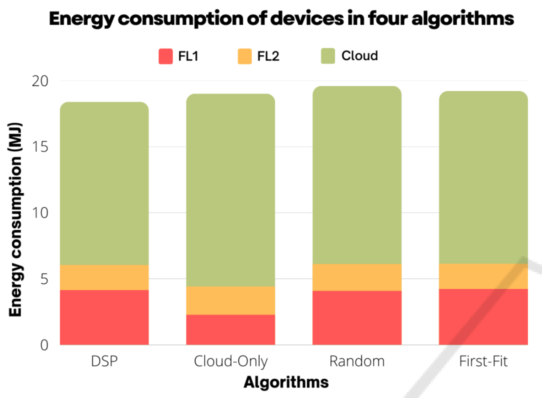

Figure 7: Energy consumption of fog devices for each algorithm.

Finally, Fig. 8 illustrates how low the network usage is when using DSP compared to other algorithms. This shows how efficient DSP is when it comes to network bandwidth and data transfer efficiency.

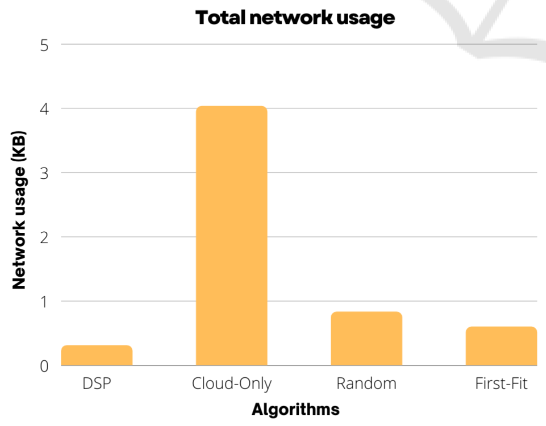

Figure 8: Total Network Usage.

6 CONCLUSION

In this paper, we have proposed a Dynamic Service Placement (DSP) algorithm in a real-time fog infrastructure. We formulated the DSP problem using Integer Linear programming (ILP) and solved it using a heuristic algorithm. DSP process the tasks emitted by applications one at a time, dynamically and in real-time. The framework selects all the fog nodes that satisfy the task's requirements and then chooses the node that minimizes the response time. The proposed algorithm performance is evaluated according to the QoS criteria and compared to other placement policies such as First Fit, Random, and Cloud Only. The results show that DSP satisfied all the application deadlines compared with the other algorithms; the applications are generally placed in first and second layers, while the energy consumption and network usage are lower than other algorithms.

REFERENCES

Apat, H. K., Nayak, R., Sahoo, B., and Mohanty, S. (2022). Application placement in fog-enabled internet of things (iot) healthcare system using body sensor networks. In *Handbook of Research on Mathematical Modeling for Smart Healthcare Systems*, pages 383–408. IGI Global.

Azizi, S., Khosroabadi, F., and Shojafar, M. (2019). A priority-based service placement policy for fog-cloud computing systems. *Computational Methods for Differential Equations*, 7(4 (Special Issue)):521–534.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

Farzin, P., Azizi, S., Shojafar, M., Rana, O., and Singhal, M. (2022). Flex: a platform for scalable service placement in multi-fog and multi-cloud environments. In *Australasian Computer Science Week 2022*, pages 106–114.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296.

Hassan, H. O., Azizi, S., and Shojafar, M. (2020). Priority, network and energy-aware placement of iot-based application services in fog-cloud environments. *IET communications*, 14(13):2117–2129.

Herrera, J. L., Galán-Jiménez, J., Bellavista, P., Foschini, L., Garcia-Alonso, J., Murillo, J. M., and Berrocal, J. (2021). Optimal deployment of fog nodes, microservices and sdn controllers in time-sensitive iot scenarios. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Khosroabadi, F., Fotouhi-Ghazvini, F., and Fotouhi, H. (2021). Scatter: Service placement in real-time fog-assisted iot networks. *Journal of Sensor and Actuator Networks*, 10(2):26.

Naas, M. I., Parvedy, P. R., Boukhobza, J., and Lemarchand, L. (2017). ifogstor: an iot data placement strategy for fog infrastructure. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 97–104. IEEE.

Natesha, B. and Guddeti, R. M. R. (2021). Adopting elitism-based genetic algorithm for minimizing multi-objective problems of iot service placement in fog computing environment. *Journal of Network and Computer Applications*, 178:102972.

Nezami, Z., Zamanifar, K., Djemame, K., and Pournaras, E. (2021). Decentralized edge-to-cloud load balancing: Service placement for the internet of things. *IEEE Access*, 9:64983–65000.

Salaht, F. A., Desprez, F., and Lebre, A. (2020). An overview of service placement problem in fog and edge computing. *ACM Computing Surveys (CSUR)*, 53(3):1–35.

Tavousi, F., Azizi, S., and Ghaderzadeh, A. (2022). A fuzzy approach for optimal placement of iot applications in fog-cloud computing. *Cluster Computing*, pages 1–18.

Tran, M.-Q., Nguyen, D. T., Le, V. A., Nguyen, D. H., and Pham, T. V. (2019). Task placement on fog computing made efficient for iot application provision. *Wireless Communications and Mobile Computing*, 2019:1–17.