sensors

MDPI

Article

Microservice Application Scheduling in Multi-Tiered Fog-Computing-Enabled IoT

Maria Ashraf [1,*], Muhammad Shiraz [2], Almas Abbasi [1], Omar Alqahtani [3], Gran Badshah [3] and Ayodele Lasisi [3]

[1] Department of Computer Science, Faculty of Computing and Information Technology, International Islamic University, Islamabad 44000, Pakistan; almas.abbasi@iiu.edu.pk
[2] Department of Computer Science, Federal Urdu University of Arts, Science and Technology, Islamabad 44000, Pakistan; muhammad.shiraz@fuuast.edu.pk
[3] Department of Computer Science, King Khalid University, Abha 61413, Saudi Arabia; osalqahtani@kku.edu.sa (O.A.); gdostan@kku.edu.sa (G.B.); alasisi@kku.edu.sa (A.L.)
* Correspondence: maria.ashraf@iiu.edu.pk

Abstract: Fog computing extends mobile cloud computing facilities at the network edge, yielding low-latency application execution. To supplement cloud services, computationally intensive applications can be distributed on resource-constrained mobile devices by leveraging underutilized nearby resources to meet the latency and bandwidth requirements of application execution. Building upon this premise, it is necessary to investigate idle or underutilized resources that are present at the edge of the network. The utilization of a microservice architecture in IoT application development, with its increased granularity in service breakdown, provides opportunities for improved scalability, maintainability, and extensibility. In this research, the proposed schedule tackles the latency requirements of applications by identifying suitable upward migration of microservices within a multi-tiered fog computing infrastructure. This approach enables optimal utilization of network edge resources. Experimental validation is performed using the iFogSim2 simulator and the results are compared with existing baselines. The results demonstrate that compared to the edgewards approach, our proposed technique significantly improves the latency requirements of application execution, network usage, and energy consumption by 66.92%, 69.83%, and 4.16%, respectively.

Keywords: fog computing; constrained devices; Internet of Things; microservice application scheduling; service delay; distributed application execution

Citation: Ashraf, M.; Shiraz, M.; Abbasi, A.; Alqahtani, O.; Badshah, G.; Lasisi, A. Microservice Application Scheduling in Multi-Tiered Fog-Computing-Enabled IoT. *Sensors* **2023**, *23*, 7142. https://doi.org/10.3390/s23167142

Academic Editor: Leopoldo Angrisani

Received: 23 June 2023
Revised: 18 July 2023
Accepted: 1 August 2023
Published: 12 August 2023

1. Introduction

With the proliferation of smart mobile devices (SMDs), ubiquitous computing, and advances in the IoT, diverse real-time applications can be executed on a plethora of these devices using their available idle resources. Emerging real-time IoT applications and cyber physical systems (CPSs) include real-time traffic monitoring, object tracking using portable video devices, health monitoring and autonomous vehicles, to name a few. Integration and collaboration of these applications from diverse domains lead to the vision of smart sustainable cities that can solve complex scenarios and situations for the development of sustainable smart IoT systems [1].

The cloud outsources significant computational and storage services to end users and makes it possible to execute computationally intensive applications on these resource-hungry devices [2]. However, high latencies that result from using central cloud services (CDCs) cannot be tolerated for real-time applications [3]. The emergence of edge/fog computing facilitates the provision of cloud services at the network edge, close to end users. This arrangement reduces latencies and bandwidth requirements on the communication infrastructure [4]. Fog computing (FC) is a hierarchical architecture that provides a low latency for the running of resource-hungry applications.

Microservice design allows for the development and deployment of applications as a collection of small, loosely coupled modular components, commonly referred to as 'microservices'. These microservices communicate through lightweight protocols to efficiently execute services requested by end users [5,6]. These services are features of an IoT application that a user may request. Nowadays, a growing number of influential IT firms or application providers, including Amazon, Netflix, and the Guardian, are developing large and complex applications with the help of microservice technologies (e.g., Kubernetes, Apache Mesos, etc.) [7]. The benefits of breaking up applications into a number of functionally distinct but logically related microservices include an increase in flexibility and ease of updating, but it also necessitates more effort from the developers to handle external complexities in application development, communication control, and failure recovery.

This has resulted in the deployment of applications on the fog or edge computing paradigm in a distributed manner [8]. The fog nodes (FNs) have heterogenous computing and storage resources and end users have different application preferences. To achieve efficient application execution on resource-constrained devices, the following key issues must be addressed:

- Given diverse FNs are available along the path from end users to the central cloud, how can we effectively select the most suitable FN or edge node (EN) by leveraging its idle resources and efficiently schedule microservices on the appropriate nodes?
- Sending services to a server for remote execution and receiving the results can further increase the overall delay of application execution. How can the latency of such critical applications be minimized?

To tackle the aforementioned challenges, we have designed an enhanced technique for microservice application placement in a multi-tiered fog-to-cloud architecture. For brevity, we call this dynamic microservice application placement (DMAP). This technique focuses on optimizing the overall latency of application execution. It leverages idle resources present in nearby IoT devices such as laptops and mobile phones, in addition to FNs and cloud servers. In this research, the primary focus is on the deployment challenge of microservice-based applications, considering the average latency and resource utilization of the edge network. The main contributions of this work are summarized as follows:

- We propose utilizing resources at hand from nearby IoT devices, FNs, and CDCs to establish an efficient, multi-tiered fog-to-cloud architecture. The primary objective of this architecture is to allocate services in proximity to end users whenever possible, with the goal of reducing latency, energy consumption, and network usage.
- We propose a decentralized resource mapping algorithm to allocate resources in a multi-tiered fog-to-cloud architecture. The algorithm aims to identify the optimal mapping of resources to tasks, ensuring an appropriate allocation.
- Finally, we demonstrate the performance of the proposed technique through simulations. Moreover, we also investigate the tradeoffs of shifting microservices from the traditional computing model of the fog–cloud and the results are compared with existing methods based on latency, bandwidth usage, and energy consumption.

The rest of this paper is organized as follows. Section 2 presents the related work on application placement. Section 3 introduces a description of the multi-tier fog-to-cloud architecture while illustrating the offloading problem. Section 4 presents the proposed technique. In Section 5, the proposed technique is investigated and elaborated on. Finally, the conclusions and future directions are highlighted in Section 6.

2. Related Work

In this section, we highlight the benefits of challenges introduced by distributed application execution in an FC environment, followed by a discussion of the need for application scheduling in the multi-tiered fog-to-cloud architecture that is being proposed in this paper.

2.1. Application Scheduling in Multi-Tiered Fog-to-Cloud Architecture

Numerous strategies are proposed in existing research to schedule applications in fog environments [9–12]. Some studies consider full application placement strategies [13], while others consider application service placement in which modules are placed on the fog servers so that multiple end users can use them [14,15].

Initially, Deng et al. [7] proposed a three-layer fog-to-cloud architecture to efficiently utilize resources. They proposed two algorithms for resource scheduling; first fit and random fit in a two-layer fog-to-cloud environment. Ren et al. [13] proposed improved resource scheduling to handle delay sensitive applications in three-layer fog-to-cloud architecture according to the delay sensitivity of the applications.

Skarlat et al. [16] introduced the concept of fog colonies. They considered one fog colony, and services were offloaded to the cloud if there was no host available to deploy the services in the colony. They extended their work and proposed if the host is not available for service deployment, resources in neighboring fog colonies should be considered [17]. Table 1 presents a comparison between the existing application placement strategies and our proposed approach DMAP.

Table 1. A qualitative comparison of the current application placement policies.

Technique	Simulator	IoT-Fog	Fog-Fog	Fog-Cloud	Failure Rate	Dynamic	Microservices	Scalability	Network Usage	Latency
[18]	iFogSim	✔	✘	✔	✘	✘	✘	✔	✔	✔
[19]	Java	✔	✘	✔	✘	✔	✘	✔	✘	✔
[20]	iFogsim	✔	✘	✔	✘	✘	✘	✔	✔	✔
[21]	Python	✘	✔	✔	✘	✔	✘	✘	✘	✔
[13]	CloudSim	✔	✔	✔	✔	✘	✘	✔	✘	✘
[22]	iFogSim	✔	✔	✔	✘	✔	✘	✔	✔	✔
[23]	iFogSim	✔	-	✔	✔	✔	✔	✔	✔	-
[24]	simulations	✔	✘	✘	✔	✔	✔	✔	✘	✔
DMAP	iFogSim2	✔	✔	✔	✔	✔	✔	✔	✔	✔

✔: Addressed, -: Partially Addressed, ✘: Not Addressed.

2.2. Discussion/Comparison

Efficiently utilizing fog and cloud resources and enhancing the quality of service during application execution requires that the crucial matter of resource scheduling be addressed. Most of the schemes consider two-tier fog-to-cloud architecture; however, they are not effective in a multi-tiered FC environment. The challenges caused by significant delays and network utilization in cloud services, as well as the opportunity to better utilize idle resources in low-resource (L-R) ENs/FNs belonging to nearby end users, prompted us to examine resource mapping. This motivated us to look into how we may utilize the resources available in the L-R edge tier. This paved our way to proposing a decentralized scheduling in FC for delay-sensitive applications, which we will discuss in detail in the following section.

To summarize, we can conclude that existing literature has shown the potential of IoT and FC concepts in facilitating real-time application execution and addressing the difficulties associated with remote CDCs. Nevertheless, a significant portion of these studies did not provide a demonstration of the selection and effective management of available and suitable resources through L-R FNs to further minimize response time and reduce the energy consumption associated with data transfer to remote FNs. The main motivation of this work is to develop a distributed microservices scheduling solution for multi-tiered fog computing-enabled IoT, as presented in Section 3.

3. System Model

We consider an environment consisting of multiple IoT devices, an n-tier edge to FC with heterogenous servers, and multiple cloud servers, as shown in Figure 1. We consider

microservices-based application workflows that can be represented as a Directed Acyclic Graph (DAG) [11].

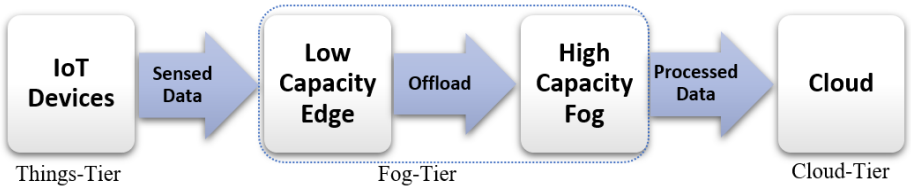


Figure 1. Three-tier fog-to-cloud architecture

3.1. Microservice-Based Applications

Microservices architecture is used in the design and development of emerging IoT applications to facilitate robust changes and the agile development. The microservices architecture provides increased granularity, allowing for the composition of multiple microservices within a single service. These microservices collaborate with each other to fulfill end-user requests. Various services may use a unique microservice. Therefore, it is more flexible and agile to define QoS settings at the composite service level as opposed to the application level. A microservices-based application is represented using a DAG, in which the vertices represent individual microservices and the edges represent the data dependencies among these microservices.

3.2. Microservices Application Scheduling

Our proposed approach DMAP aims to satisfy the critical latency requirements of application execution by using the underutilized resources of nearby devices instead of placing services on the cloud data centers in a multi-tiered FC environmment. Idle resources of smart IoT devices can also be utilized to provide the computational services. In DMAP, traditional fog devices comprise resource-rich (R-R) fog-tier and SMDs or IoT devices with processing capability and storage availability form the L-R edge-tier. These tiers can be extended to an n-tier fog cloud computing environment to reduce latency, energy consumption and bandwidth usage in an IoT environment, hence improving the QoS of application execution [7]. The higher the FN is from the IoT device, the more computational resources it contains. A three-tier fog-to-cloud architecture, along with its connection, is shown in Figure 2. It can easily be extended to an n-tier FC architecture.

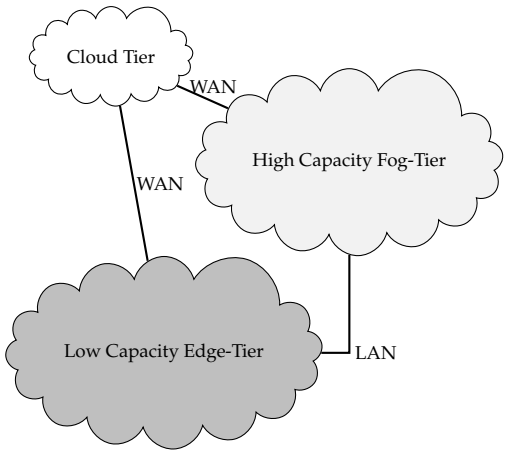


Figure 2. Three-tier fog-to-cloud architecture.

The main components in three-tier fog-to-cloud computing architecture are:

1. IoT Device/Things: IoT devices or other SMDs, e.g., heart rate monitors, can communicate to transmit the sensed data to fog tier using the following protocols: IEEE

802.15.4, WiFi, Bluetooth, MQTT, etc., for data processing and information retrieval. They include mobile phones, laptops, computers, cameras, smartwatches, etc., as well as small energy and power-constrained sensors and actuators.

2. L-R Edge: Represents L-R edge devices that are present next to IoT devices. They are responsible for processing data sensed by the things tier. They are equipped with computational, storage and communicational capabilities and are present between R-R fog servers and IoT devices. Every fog device consists of a tier controller that receives tasks from low-tier devices. These tasks, along with their CPU and delay requirements, are sent to an appropriate device for execution.
3. R-R Fog: consists of fog servers that are more powerful resources as compared to the local edge, and they act as a bridge to connect to CDC.
4. Cloud: provides ubiquitous access to huge shared resources (e.g., computational and storage) over the network.
5. Tier Controller: It makes the decision whether to admit the task or offload it to a higher tier. Tasks are inserted into the queue on arrival. Highly sensitive tasks are sent to the L-R edge tier for execution. If the resources are available, they are executed there; otherwise, they are sent to the controller at R-R fog-tier. Delay-tolerant tasks, however, are sent to a conventional cloud tier for remote execution.

In order to execute applications on resource constraint devices, the delay-sensitive tasks are offloaded to nearby L-R or R-R fog tier depending on the resource availability of ubiquitous mobile devices in the FC environment. We propose a priority rule: if sufficient computing resources are available at the L-R edge-tier, it is offloaded. Otherwise, task placement is shifted upwards towards the R-R fog-tier on the way towards the cloud. If the available device has sufficient resources and can share its computing services, the task is offloaded there; otherwise, the task is executed on a central cloud.

When a mobile device joins a network, it connects with the access point (AP) in the L-R edge-tier. Whenever an application requests computational services for execution of a service, the request is sent to the tier controller, along with the required amount of CPU resource, memory, and its delay requirements. The tier controller resides within the broker and serves the purpose of granting access to services on devices located in L-R, R-R, or the conventional cloud. This access is determined based on the resource requirements of the task and the resources that are currently available on the fog device within L-R, R-R, or the conventional cloud, ensuring their suitability for execution. If the processing requirement of the task is greater than the processing capabilities of the fog device, the task is shifted upwards towards the cloud, where the resources are already sufficient to fulfill the processing requirements of the task.

The tier controller, as shown in Figure 3, is responsible for resource allocation. The data produced can be processed at EN, FN or the cloud depending on the service requirement and the resource capabilities of the computing node. The task scheduler is responsible for actual scheduling of tasks on the virtual machine at the fog server. If a high number of requests is received by tier controller, the requests are shifted upwards towards the cloud to release the pressure on L-R ENs and to ensure efficient resource utilization in fog-to-cloud architecture. The scenario in which an intelligent decision is also made to choose an appropriate server in a fog-to-cloud architecture for task execution will be considered in our future work.

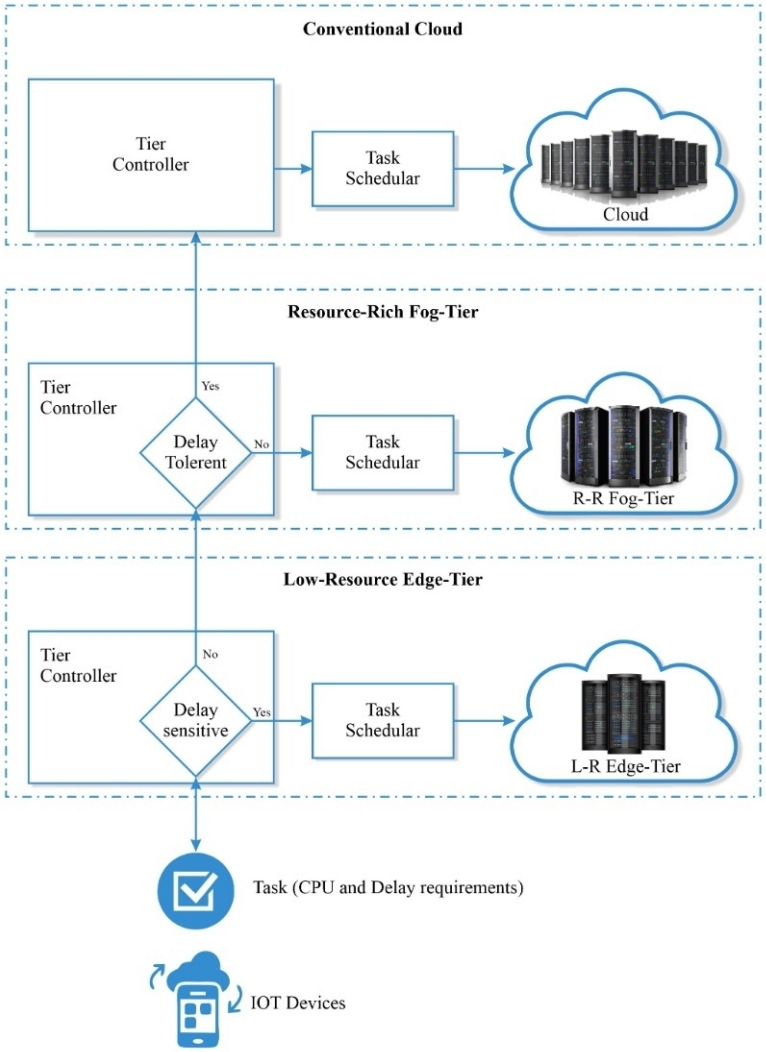


Figure 3. Multi-tiered fog computing architecture.

3.3. Performance Metric

We formulate the task offloading problem as an optimization problem that aims to minimize latency, network usage and service failure rate. The proposed algorithm is presented in Section 4. To evaluate the performance of the proposed approach, the following performance metrics are considered.

3.3.1. Minimize Delay

The total delay of microservice, denoted by, m [25] can be measured as follows:

$$\mathcal{D}_m(t) = \mathcal{C}_m(t) + \mathcal{T}_m(t) + \mathcal{W}_m(t). \quad (1)$$

where $\mathcal{C}_m$ is the time spent actually executing the microservice. $\mathcal{T}_m$ is the time spent transmitting the service data to the server and the time spent sending the results back. $\mathcal{W}_m$ is the time spent waiting in the queue. The computational time can be measured as follows:

$$\mathcal{C}_m(t) = \mu_m(t)/\mathcal{PR}_s(t) \quad (2)$$

where μ represents the processing requirements of microservice, and $\mathcal{PR}_s$ represents the processing capabilities of the server. At any time t, network usage can be calculated using Equation (3):

$$\mathcal{NU}(t) = \delta * \mathcal{T}_m(t) \quad (3)$$

where δ represents the tuple length.

Microservice can be offloaded to a surrogate device in L-R edge, R-R fog or conventional cloud. The transmission time taken to offload ith microservice to the server can be measured as:

$$\mathcal{T}_i = \mathcal{X}_i / \mathcal{B}_{up} + \mathcal{R}_i / \mathcal{B}_{down} \tag{4}$$

where $\mathcal{X}_i$ represents the size of the microservice, $\mathcal{R}_i$ represents the size of the results, and $\mathcal{B}_{up}$ and $\mathcal{B}_{down}$ represents the bandwidths available for uploads and downloads, respectively.

Due to the complex dynamic scenario, the task arrival rate and execution rate should follow general distribution. The total time spent by each microservice i waiting for its execution on server j is computed as follows:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \mathcal{W}_{ij} \tag{5}$$

We model our system using $M/M/1$. In this model, the first M represents the task arrival rate, which is governed by a Poisson distribution. The second M indicates that the task execution rate follows an exponential distribution, with a mean service time. The parameter n corresponds to the number of servers, each having unique capabilities and being available to process the incoming tasks. This waiting time is computed using queuing theory [26], considering the arrival rate and processing rate of the server to identify the congested server, as shown in Equation (6):

$$\mathcal{W}_m(t) = (\sigma^a + \sigma^s)/(2(X^a - \mu^s)) \tag{6}$$

where μ^s and σ^s are the standard mean and variance of the microservice execution time, respectively, and X^a and σ^a represent the mean and variance of its arrival intervals, respectively. The primary goal is to reduce the application execution response time for all incoming tasks. As a result, an optimal server is chosen that has enough computing resources to minimize computing time, considers the available bandwidth to minimize the transmission time, and measures the load on the server to minimize queueing delay. As a result, DMAP optimizes the task placement strategy for all microservices on available computing devices by assigning the greatest number of delay-sensitive tasks to the L-R ENs, reducing the latency and overall offloading time.

Let $\mathcal{FT}_i$ denote whether the task has been finished within the deadline constraint Equation (7b). If it is equal to one, the respective task has been completed before the deadline; otherwise, it has not been completed before the deadline. The above objectives and constraints are formulated as follows:

$$Minimize : \mathcal{D}_m(t) \tag{7a}$$

$$Subject\ to : \mathcal{FT}_t = \begin{cases} 1, \mathcal{D}_m(t) < \mathcal{D}max_m(t) \\ 0, otherwise \end{cases} \tag{7b}$$

$$Maximize : \sum_{i=1}^{m} \mathcal{FT}_i(t) \tag{7c}$$

In general, objectives for the scheduling problem are formulated in Equation (7). These include maximizing the number of finished tasks within deadline constraint Equation (7c) and minimizing the application execution delay in Equation (7a).

3.3.2. Analysis of Failure Rate

An application is considered as failed if its execution is not completed within the deadline. There are several possible reasons why this failure could have happened. In

cases in which the device does not possess sufficient resources to finish the task within the deadline, experiences extended waiting period in the execution queue or faces network congestion, the finished time, as indicated in Equation (7b), becomes zero, signifying that the task could not be executed successfully before the deadline. The failure rate of an application at a specific time is given as follows:

$$\mathcal{FR}(t) = \eta/\mathcal{N} \tag{8}$$

where $\mathcal{FR}$ represents the failure rate at instance t, η represents the number of failed tasks and N represents the total number of tasks submitted for execution.

4. Proposed Microservices Application Scheduling

Based on the proposed n-tier fog-to-cloud architecture, we propose a microservice scheduling technique. The performance of the proposed algorithm is compared with baseline approaches, as discussed in Section 5. In the proposed DMAP, arriving microservices are inserted in a waiting queue, where they await their turn for execution. The computing capacities of each fog server in the L-R edge-tier are measured (line 7 of Algorithm 1) and are compared with the resource requirements of the microservice module represented by CPU_{mn}. They are then assigned to L-R EN that have enough computational resources for execution. Furthermore, different modules can be assigned to different ENs for execution. They can be executed in parallel. If the actual delay of the microservice is less than the threshold value, $\mathcal{FT}$ is updated to 1 and the service is marked to be executed successfully (line 9–11 of Algorithm 1). If the computing requirements of a module exceed the computing capabilities of EN in the L-R edge-tier, it is shifted to R-R fog-tier (line 14 of Algorithm 1). Algorithm 1 exhibits the workings of the microservice application placement in L-R edge in a three-tier fog-to-cloud environment.

Algorithm 1 Workings of microservices application placement in L-R edge in three-tier fog-to-cloud

```
procedure MICROSERVICE-PLACEMENT-LR(
)
    for each arriving application A_mn do
        insert all its microservices m_mn in Queue Q for execution
    end for
    for each m_mn in Q do
        for each virtual machine VM_i in L-R do
            calculate CPU_vm
            if CPUv_mi ≥ CPU_mn then
                execute m on VM_i
                if FT_m = 1 then
                    mark the task as success
                end if
            else
                MicroservicePlacmentRR(m_mn)
            end if
        end for
    end for
end procedure
```

When the L-R edge-tier does not have enough resources to execute the microservice, the resources of the R-R fog tier in the three tier fog-to-cloud environment are considered. Algorithm 2 presents the steps involved in executing the microservice in the R-R fog tier in a three-tier fog-to-cloud environment. All the microservices are inserted in a queue waiting for their turn of execution. The computing capacity of each VM in the R-R fog tier is then measured. If the resource capacity (CPU_{Vmi}) of VM_i is higher than the resource requirements of the microservice mn CPU_{mn}, the microservice is outsourced to VM_i; otherwise, it is outsourced to the cloud for its execution. If the service completes its execution within the specified delay requirement, FT is updated to one.

Algorithm 2 Working of microservices application placement in R-R fog tier in three-tier fog-to-cloud technique

```
procedure MICROSERVICE-PLACEMENT-RR(
)
    for each arriving microservice in HC m_mn do
        insert all incoming microservices m_mn in Queue Q to wait for their turn
    end for
    for each m_mn in Q do
        for each virtual machine VM_i in HC do
            calculate CPU_vm
            if CPUv_mi ≥ CPU_mn then
                execute m on VM_i
                if FT_m = 1 then
                    mark task as success
                end if
            else
                MicroservicePlacmentCloud(m_mn)
            end if
        end for
    end for
end procedure
```

5. Performance Evaluation

This section presents the empirical evaluation of DMAP in comparison to the baseline approaches proposed in the existing literature. We begin with explaining the configuration of the simulation, our execution of the methods, and the introduction of the DMAP technique we propose. Following that, we outline the examination of the outcomes. Finally, we summarize our findings.

5.1. Simulation Setup and Parameters

A simulation-based approach is considered to experiment with different parameter configurations. The proposed approach is evaluated in iFogSim2 [27], which is built on CloudSim simulator [28]. To evaluate the performance of the proposed technique, a mobile device can run a delay-sensitive application, EEG tractor beam game (EEGTBG) [18], or delay-tolerant application, video surveillance/object tracking (VSOT) [29], with a gradually increasing number of users. These applications are developed as a set of microservices that are executed separately on the containers by the simulators. The details of the application model, with their modules and interactions among them, can be found in [29, 30]. We ensure the heterogeneity of microservices in terms of their respective computation costs as (300–900 MIPS) and bandwidth usage as (200-1500 bytes/packet). Additionally, a resource request of 100 and 200 per second is taken into consideration. The IoT simulation benchmarks reported in earlier simulation studies [18,29] are the foundation for all the aforementioned parametric values. Table 2 shows the overall configurations used in our simulations. B and I in Table 2 represent the busy and idle status of a device, respectively. The processing requirements of these two application modules are shown in Table 3.

Table 2. Configuration parameters based on [18].

Device	CPU (GHz)	RAM (GB)	Processing (Mips)	Power (W)
Cloud	3.0	49,152	80,000,000	16 × 103 (B), 16 × 83.25 (I)
Fog	3.0	2.8	2048	107.339 (B), 83.433 (I)
Edge	3.0	40	44,800	107.339 (B), 83.433 (I)
IoT	1	40	44,800	87.53 (B), 82.44 (I)

Table 3. Processing requirements of EEGTBG and VSOT application modules based on [31].

EEGTBG					VSOT	
Object Detector	**Motion Detector**	**Object Tracker**	**User Interface**	**Client**	**Concentration Calculator**	**Coordinator**
550	300	300	200	200	350	100

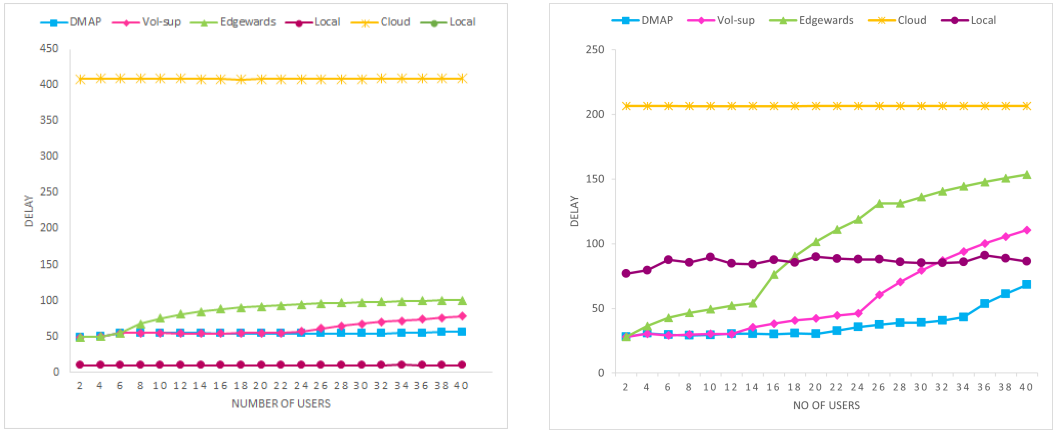
5.2. Implementation of Algorithms

The efficiency of the proposed technique is investigated according to the following approaches:

- Local: In this approach, all services are executed locally on the mobile device.
- Cloud Only: In this approach, all services except the user interface module are executed on cloud data centers.
- Edgewards [32]: All services are executed on the EN if resources are available there; otherwise, they are offloaded upwards towards the central cloud.
- Volunteer-Supported (Vol-sup) [18]: This is an extension of the Edgewards approach, in which idle/underutilized resources of volunteer devices near FNs are used to execute services in parallel to address the issues of higher latency, network usage and energy consumption instead of offloading them to remote cloud servers. The nearby resources are considered horizontally before offloading services to remote cloud servers.

5.3. Delay Analysis

The loop delay for the proposed technique DMAP, Vol-sup, Edgewards, local-only, and cloud-only approaches is shown in Figure 4. All of them first execute their services at the nearby ENs, causing a comparable latency, until more users are added, and ENs can no longer support the additional users. The microservices are then moved to the cloud using the Edgewards approach, while they are moved to R-R by L-R according to the proposed technique, causing a slow increase in their respective latency.



(**a**) EEGTBG (**b**) VSOT

Figure 4. Delay Comparison.

Due to the core network's capacity restriction, the latency of Edgewards approach increases with user 8 onwards for EEGTBG, as shown in Figure 4a. Likewise, the amount of data sent to the cloud grows as the number of users increases, eventually using all of the available bandwidth. Even though the cloud has an overwhelming amount of computing capacity, bandwidth restrictions create transmission delays that effect latency incurred using the Edgewards approach. The delay of the vol-sup approach starts increasing from user 24 onwards due to the bandwidth limitation of the core network as compared to DMAP, which can handle 40 users diligently. The DMAP tries to offload the services to the

L-R edge-tier, and then H-R fog-tier, before offloading services to the central cloud. This results in offloading few services towards the cloud, as the maximum amount of resources available in fog tier and edge tier is utilized. There is no delay involved in executing application locally on the IoT device, but the energy consumption of local execution is comparatively high.

Figure 4b depicts a delay loop for delay-tolerant application (VSOT). Its value increases drastically as 14 users join the network for the Edgewards approach, whereas it can accommodate 24 users in the vol-sup approach, which is increased to 34 in the proposed technique. Hence, as depicted in Figure 4, it can be concluded that under normal to heavy load situations, which are reasonably predicted given the flood of data that are anticipated to be created under the IoT storm, our proposed technique DMAP performs better as compared to baseline approaches.

5.4. Network Usage Analysis

Figure 5 shows the network usage for EEGTBG and VSOT applications. The presence of the cloud with a propagation delay of 100 ms increases network usage. It is clearly depicted in Figure 5 that the bandwidth for the cloud-only approach is the worst, as microservices are moved to a distant cloud. With an increase in the user count, there is a corresponding rise in network load, leading to substantial network utilization. However, if we look closely at Figure 5a, we can see that the slope of network utilization for the Edgewards curve is steeper than that for the proposed approach due to the increased load. This spike also illustrates bandwidth constraint, which, as already discussed in the previous section, causes the delay in the Edgewards approach to increase drastically. On the other hand, because the delay of the IoT to edge tier link is just about 2 ms, the network utilization of the proposed technique is quite low. It increases drastically once 34 users join the network, which causes the resources of proximate servers to be exhausted and services to be offloaded to distant servers, compared to 16 for the Edgewards approach and 24 for the vol-sup approach, as shown in Figure 5b. As a result, when compared to the vol-sup and Edgewards approach, the network is busy for a shorter period of time. Therefore, as a result, the network is only momentarily active.

(**a**) EEGTBG (**b**) VSOT

Figure 5. Network usage comparison.

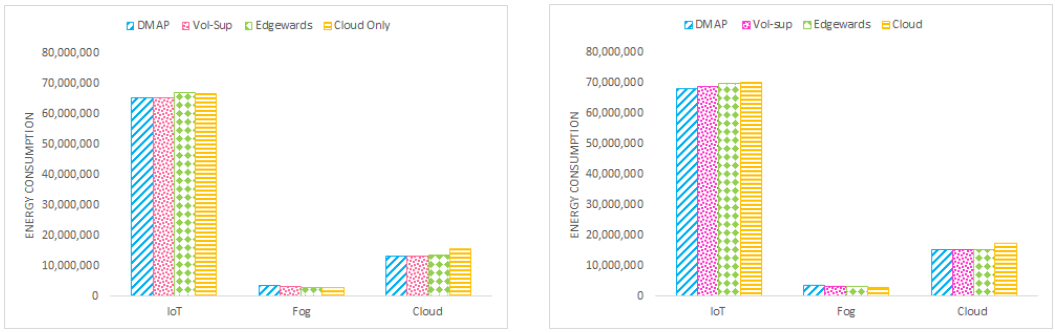
5.5. Energy Consumption of IoT Devices Analysis

Figure 6 compares the energy consumption rate of the proposed approach with local execution on IoT devices. The energy consumption of IoT devices is crucial; hence, it is taken into account when executing microservices. Only local execution is considered for comparison. This is because, when an offloading decision is made, IoT's idle energy consumption rate is used. Therefore, other approaches are not taken into consideration for comparison. In contrast, when all services are executed locally, there is a higher rate of power usage.

(**a**) EEGTBG (**b**) VSOT

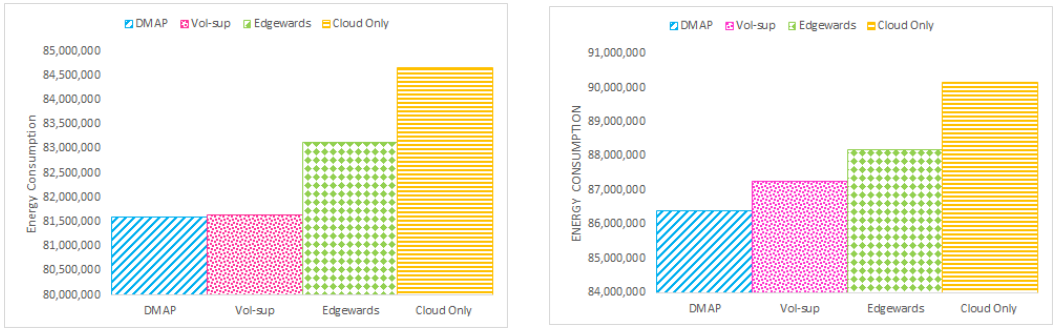
Figure 6. Energy consumption comparison.

The energy consumption of IoT devices, fog nodes and the cloud data center is shown in Figure 7. As is evident from Figure 7, the energy consumption of fog devices of the proposed technique is the highest compared to the rest of the approaches. This is because there is an attempt to deploy the maximum number of services at neighboring devices in the L-R edge tier or R-R fog tier. Figure 8 represents the total energy consumption of all devices in multi-tiered FC. By comparing the total energy consumption in this graph, it can be deduced that DMAP performs better than alternative strategies by decreasing the overall energy consumption of all devices within the multi-tiered FC architecture.



(**a**) EEGTBG (**b**) VSOT

Figure 7. Energy consumption of tier landscape comparison.

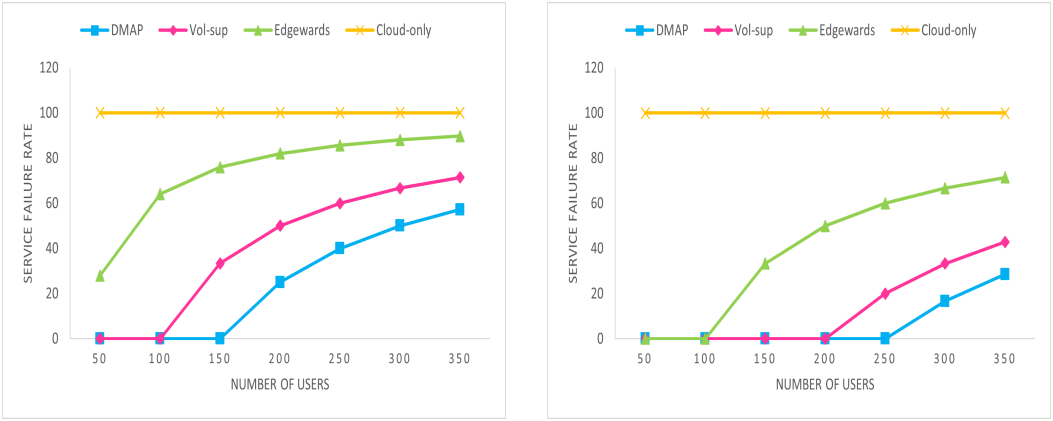


(**a**) EEGTBG (**b**) VSOT

Figure 8. Total energy consumption comparison.

5.6. Service Failure Rate Analysis

Figure 9 illustrates the number of delay-sensitive tasks that are executed before their deadlines. It is evident that for a cloud-only approach, the delay is quite high and deadlines of tasks are not met. Task deadlines are mostly met; but, in some scenarios, they are not met due to limited resource constraints and communication overhead of local FNs, when the tasks need to be offloaded to a distant cloud. Hence, in most scenarios, DMAP fulfills the demands of tasks generated by IoT devices.



(**a**) EEGTBG (**b**) VSOT

Figure 9. Service failure rate comparison.

5.7. Summary

In this study, we focused on investigating the impact of outsourcing services to the R-R fog-tier. The offloading decision is made by the broker for its respective IoT devices. As the number of services increases, the delay and network utilization of the proposed technique DMAP proves to be superior as compared to the baseline approaches. This is verified by experimental results, as the proposed technique uses underutilized resources from the L-R edge tier and R-R fog tier, before offloading tasks to the cloud, lowering the energy consumption of all IoT devices. Since services are performed on nearby servers, network usage is also reduced. The proposed technique stands out from other methods due to these two significant reasons, as is evident from the analysis. It minimizes delay and network utilization by utilizing the resources of the R-R fog tier as well as the L-R edge tier before offloading services to the central cloud.

6. Conclusions and Future Work

This paper focuses on investigating the issue of microservices application scheduling in a multi-tier fog environment. Two fog tiers are implemented for this purpose: nearby L-R edge tier and R-R fog tier. Extensive simulations are conducted to compare the proposed technique with baseline approaches. Our findings reveal that the Edgewards method produces satisfactory outcomes for relatively small problem instances. However, when the infrastructure size and/or the number of services to be placed increases, the performance of the edgewards approach significantly declines. In contrast, all evaluated decentralized techniques exhibit superior scalability under these circumstances.

We intend to expand the proposed technique to include deep learning approaches in future work. The tasks can be forwarded to nearby FNs, and fog clustering techniques can be considered for further horizontal resource load balancing. Real-time implementation with fine-grained optimization can also be used to examine the effects of proposed decentralised microservices placement to validate these findings.

Author Contributions: Conceptualization, M.A., M.S., A.A., O.A., G.B. and A.L.; methodology, M.A., M.S. and A.A.; software, M.A. and O.A.; validation, M.A., M.S. and A.A.; formal analysis, M.A. and M.S.; writing—original draft preparation, M.A.; writing—review and editing, M.S., A.A., O.A., G.B. and A.L.; visualization, O.A.; supervision, M.S. and A.A.; project administration, G.B. and A.L.; funding acquisition, A.L. All authors have read and agreed to the published version of the manuscript.

Funding: Deanship of Scientific Research at King Khalid University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All supporting data will be provided on request.

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University, Saudi Arabia for funding this work through a Large Group Research Project under grant number RGP2/394/44.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MCC	Mobile Cloud Computing.
FC	Fog Computing.
SMDs	Smart Mobile Devices.
CPS	Cyber–Physical Systems.
FNs	Fog Nodes.
ENs	Edge Nodes.
R-R	Resource Rich.
L-R	Low Resource.

References

1. Khan, L.U.; Yaqoob, I.; Tran, N.H.; Kazmi, S.A.; Dang, T.N.; Hong, C.S. Edge-computing-enabled smart cities: A comprehensive survey. *IEEE Internet Things J.* **2020**, *7*, 10200–10232. [CrossRef]
2. Shiraz, M.; Sookhak, M.; Gani, A.; Shah, S.A.A. A study on the critical analysis of computational offloading frameworks for mobile cloud computing. *J. Netw. Comput. Appl.* **2015**, *47*, 47–60. [CrossRef]
3. Shiraz, M.; Gani, A.; Shamim, A.; Khan, S.; Ahmad, R.W. Energy efficient computational offloading framework for mobile cloud computing. *J. Grid Comput.* **2015**, *13*, 1–18. [CrossRef]
4. Tahirkheli, A.I.; Shiraz, M.; Hayat, B.; Idrees, M.; Sajid, A.; Ullah, R.; Ayub, N.; Kim, K.I. A survey on modern cloud computing security over smart city networks: Threats, vulnerabilities, consequences, countermeasures, and challenges. *Electronics* **2021**, *10*, 1811. [CrossRef]
5. Lewis, J.; Fowler, M. Microservices: A Definition of This New Architectural Term. Available online: https://martinfowler.com/articles/microservices.html (accessed on 8 August 2023).
6. Newman, S. *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*; O'Reilly Media: Sebastopol, CA, USA, 2019.
7. Deng, S.; Xiang, Z.; Taheri, J.; Khoshkholghi, M.A.; Yin, J.; Zomaya, A.Y.; Dustdar, S. Optimal application deployment in resource constrained distributed edges. *IEEE Trans. Mob. Comput.* **2020**, *20*, 1907–1923. [CrossRef]
8. Ashraf, M.; Shiraz, M.; Abbasi, A.; Albahli, S. Distributed application execution in fog computing: A taxonomy, challenges and future directions. *J. King Saud-Univ.-Comput. Inf. Sci.* **2022**, *34*, 3887–3909. [CrossRef]
9. Yuan, H.; Tang, G.; Li, X.; Guo, D.; Luo, L.; Luo, X. Online Dispatching and Fair Scheduling of Edge Computing Tasks: A Learning-Based Approach. *IEEE Internet Things J.* **2021**, *8*, 14985–14998. [CrossRef]
10. Tran, M.Q.; Nguyen, D.T.; Le, V.A.; Nguyen, D.H.; Pham, T.V. Task placement on fog computing made efficient for iot application provision. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 1–17. [CrossRef]
11. Goudarzi, M.; Wu, H.; Palaniswami, M.; Buyya, R. An application placement technique for concurrent IoT applications in edge and fog computing environments. *IEEE Trans. Mob. Comput.* **2020**, *20*, 1298–1311. [CrossRef]
12. Shaheen, Q.; Shiraz, M.; Hashmi, M.U.; Mahmood, D.; Zhiyu, Z.; Akhtar, R. A lightweight location-aware fog framework (LAFF) for QoS in Internet of Things paradigm. *Mob. Inf. Syst.* **2020**, *2020*, 1–15. [CrossRef]
13. Ren, Z.; Lu, T.; Wang, X.; Guo, W.; Liu, G.; Chang, S. Resource scheduling for delay-sensitive application in three-layer fog-to-cloud architecture. *Peer-to-Peer Netw. Appl.* **2020**, *13*, 1474–1485. [CrossRef]
14. Wu, H.; Knottenbelt, W.J.; Wolter, K. An efficient application partitioning algorithm in mobile environments. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 1464–1480. [CrossRef]
15. Guerrero, C.; Lera, I.; Juiz, C. Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures. *Future Gener. Comput. Syst.* **2019**, *97*, 131–144. [CrossRef]
16. Skarlat, O.; Schulte, S.; Borkowski, M.; Leitner, P. Resource provisioning for IoT services in the fog. In Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 4–6 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 32–39.
17. Skarlat, O.; Nardelli, M.; Schulte, S.; Dustdar, S. Towards qos-aware fog service placement. In Proceedings of the 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, 14–15 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 89–96.
18. Ali, B.; Pasha, M.A.; ul Islam, S.; Song, H.; Buyya, R. A volunteer-supported fog computing environment for delay-sensitive iot applications. *IEEE Internet Things J.* **2020**, *8*, 3822–3830. [CrossRef]

19. Mann, Z.A. Decentralized application placement in fog computing. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 3262–3273. [CrossRef]
20. Hassan, S.R.; Ahmad, I.; Ahmad, S.; Alfaify, A.; Shafiq, M. Remote pain monitoring using fog computing for e-healthcare: An efficient architecture. *Sensors* **2020**, *20*, 6574. [CrossRef] [PubMed]
21. Gao, X.; Huang, X.; Bian, S.; Shao, Z.; Yang, Y. PORA: Predictive offloading and resource allocation in dynamic fog computing systems. *IEEE Internet Things J.* **2019**, *7*, 72–87. [CrossRef]
22. Hossain, M.R.; Whaiduzzaman, M.; Barros, A.; Tuly, S.R.; Mahi, M.J.N.; Roy, S.; Fidge, C.; Buyya, R. A scheduling-based dynamic fog computing framework for augmenting resource utilization. *Simul. Model. Pract. Theory* **2021**, *111*, 102336. [CrossRef]
23. Pallewatta, S.; Kostakos, V.; Buyya, R. QoS-aware placement of microservices-based IoT applications in Fog computing environments. *Future Gener. Comput. Syst.* **2022**, *131*, 121–136. [CrossRef]
24. Samanta, A.; Tang, J. Dyme: Dynamic microservice scheduling in edge computing enabled IoT. *IEEE Internet Things J.* **2020**, *7*, 6164–6174. [CrossRef]
25. Lu, J.; Hao, Y.; Wu, K.; Chen, Y.; Wang, Q. Dynamic offloading for energy-aware scheduling in a mobile cloud. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 3167–3177. [CrossRef]
26. Yang, S.; Adeel, U.; Tahir, Y.; McCann, J.A. Practical opportunistic data collection in wireless sensor networks with mobile sinks. *IEEE Trans. Mob. Comput.* **2016**, *16*, 1420–1433. [CrossRef]
27. Mahmud, R.; Pallewatta, S.; Goudarzi, M.; Buyya, R. Ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *J. Syst. Softw.* **2022**, *190*, 111351. [CrossRef]
28. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
29. Bittencourt, L.F.; Diaz-Montes, J.; Buyya, R.; Rana, O.F.; Parashar, M. Mobility-aware application scheduling in fog computing. *IEEE Cloud Comput.* **2017**, *4*, 26–35. [CrossRef]
30. Pallewatta, S.; Kostakos, V.; Buyya, R. Microservices-based IoT application placement within heterogeneous and resource constrained fog computing environments. In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing, Auckland, New Zealand, 2–5 December 2019; pp. 71–81.
31. Djemai, T.; Stolf, P.; Monteil, T.; Pierson, J.M. A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures. In Proceedings of the 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), Amsterdam, The Netherlands, 5–7 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 32–40.
32. Gupta, H.; Vahid Dastjerdi, A.; Ghosh, S.K.; Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Softw. Pract. Exp.* **2017**, *47*, 1275–1296. [CrossRef]