Almutairi and Aldossary *Journal of Cloud Computing: Advances, Systems and Applications* (2021) 10:28
https://doi.org/10.1186/s13677-021-00243-9

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

A novel approach for IoT tasks offloading in edge-cloud environments

Jaber Almutairi[1] and Mohammad Aldossary[2*]

Abstract

Recently, the number of Internet of Things (IoT) devices connected to the Internet has increased dramatically as well as the data produced by these devices. This would require offloading IoT tasks to release heavy computation and storage to the resource-rich nodes such as Edge Computing and Cloud Computing. Although Edge Computing is a promising enabler for latency-sensitive related issues, its deployment produces new challenges. Besides, different service architectures and offloading strategies have a different impact on the service time performance of IoT applications. Therefore, this paper presents a novel approach for task offloading in an Edge-Cloud system in order to minimize the overall service time for latency-sensitive applications. This approach adopts fuzzy logic algorithms, considering application characteristics (e.g., CPU demand, network demand and delay sensitivity) as well as resource utilization and resource heterogeneity. A number of simulation experiments are conducted to evaluate the proposed approach with other related approaches, where it was found to improve the overall service time for latency-sensitive applications and utilize the edge-cloud resources effectively. Also, the results show that different offloading decisions within the Edge-Cloud system can lead to various service time due to the computational resources and communications types.

Keywords: Edge-cloud computing, Edge orchestrator, Resource management, Latency sensitivity, Task offloading, Scheduling, Internet of things

Introduction

In recent years, the Information Technology (IT) sector has developed at a massive rate, in which more than 50 billion Internet of Things (IoT) devices will be connected to the internet in the coming years [1–4]. In addition, the availability of stable and high-speed internet as well as communication technologies lead to the proliferation of complex, and computation-intensive IoT applications that often generate and process large volumes of data. Such applications include Augmented Reality (AR), Online Gaming and processing of Video Streaming [5]. This immense growth, therefore, requires platforms to support the increased amount of IoT devices and to organize and process the produced data. However, the limited power and computational capabilities (i.e., CPU and memory) further restrict the execution of such resource-demanding applications on the devices [6]. To alleviate these limitations and meet the communication/processing delay requirement, complex computations can be offloaded to more resourceful devices.

Cloud Computing is considered viable and promising technology to support this growth by enabling on-demand access to a massive pool of computation resources for services process and data analytics [7, 8]. Notwithstanding this, cloud computing resources are centralized and far away from IoT devices where the enormous amount of data generated by IoT devices is required to be transferred and processed in a real-time manner. Therefore, the cloud computing paradigm is not suitable for addressing low-latency, real-time interaction and high Quality of Service (QoS) applications due to network delay [9].

To address the cloud computing limitations, edge computing paradigm has been emerged where it provides a pool of virtually operated computational and storage

*Correspondence: mm.aldossary@psau.edu.sa
[2]Department of Computer Science, College of Arts and Science, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia
Full list of author information is available at the end of the article

capabilities at the edge of network that are proximity to IoT devices and thereby fulfill the latency gaps [10, 11]. In addition, it provides the opportunity to serve better streaming services, which are both latency-sensitive and bandwidth-intensive such as Google Stadia and Netflix. Moreover, edge computing architecture avoids uploading/downloading of massive files and prevents pre-processing of offloading tasks, which contributes to minimize the overall service time [12, 13]. However, managing edge-cloud computing resources efficiently and handling computation tasks for latency-sensitive applications is a critical issue [14] which could require proposing an efficient task scheduling technique to enhance the overall service performance and minimize the delay of offloaded tasks.

Furthermore, there are several parameters that influence proposing a realistic model of scheduling the offloaded tasks on the edge-cloud system. These parameters can be classified into two main categories namely *infrastructure characteristics* and *application characteristics*. The first category (i.e., Infrastructure characteristics) deals with the features related to the infrastructure such as allocating the appropriate resource for a specific task, managing the utilization level of the edge server and the network conditions. For example, CPU utilization could vary depending on the assigned task and whether the number of IoT devices increases in a shared network, which may lead to fluctuations in network bandwidth. Whereas, the second category (i.e., application characteristics) manages the characteristics of IoT application tasks such as computation demand, required transfer data for uploading and downloading, and the required deadline to complete tasks.

Note that, regarding the previous explanation, the realistic model for scheduling the offloaded tasks on the edge-cloud system considers different parameters in terms of application characteristics (computational, communications and latency), resource heterogeneity and resource utilization which can be formulated as a dynamic multi-objective optimization problem, and can vary over time [15]. However, it is a challenge to implement an optimal scheduling approach and gets accurate mathematical models [16] in edge-cloud environment due to its complexity, uncertainty and vagueness [17] where it changes dynamically and unpredictably. For instance, the number of IoT devices could be increased or decreased in a specific area due to IoT mobility, which has an impact on the load of the edge node and the shared network. Also, the incoming tasks are not known in advance, which requires a system to handle them in real-time. Furthermore, the edge-cloud environments consist of a set of heterogeneous resources (e.g., different computation resource capabilities).

In this regard, several and significant research efforts have been intended to address the latency challenges and resource heterogeneity in the edge-cloud environment. For example, researchers in [18–20] have considered the computational and communication parameters in order to enhance the overall latency. Elsewhere, [5, 21] have investigated the impact of resource heterogeneity in the edge-cloud environment and its role in enhancing the end-to-end service time. Additionally, [22–24] have focused on load balancing and server utilization in Edge-Cloud systems in an effort to avoid overloaded edge nodes, which affect application service time. However, most of the studies are designed to meet a specific scenario or for a particular application, which makes them less adaptive and scalable [25]. In addition, the complexity and computational time needed for solving this type of problems are not addressed, where the resources at the edge involve computational constraints. Moreover, most of the proposed approaches use traditional methods for solving this problem which assumes that all the parameters of an optimization model are precisely known and could add extra overhead at the edge nodes, thereby affects the ability to meet stringent service requirements for latency-sensitive applications [15].

Fuzzy Logic (FL) is a method of reasoning that seems closer to the way our brains work. The concept of fuzzy logic is to abstract the problem complexity to a level that can be understood. It helps to model imprecision and uncertainty of the system, where it can define the imprecise information in a more logical and meaningful fashion [26]. It can also handle system uncertainty by dealing with many input and output variables and can represent the problem with simple if-then rules. Many researchers in the field of the distributed systems use fuzzy logic to deal with the challenges caused by vagueness, uncertainty and the dynamicity of the environment [27]. Therefore, in this study, *Fuzzy Logic* is considered to be among the most feasible solutions for a multi-objective optimization problem when the activity of multiple parameters is significant. It can be easily adapted to the dynamicity of computational resources and application parameters as well as providing scalability within the context of the system. It also averts the computational complexity and can provide decisions very quickly [28]. As a consequence, fuzzy logic has been adopted in this research to determine where to offload the tasks based on application and system parameters. To the best of our knowledge, this is one of the early attempts to design and implement such a system with regards to application's demands, edge-cloud resource utilization and resource heterogeneity by adopting fuzzy logic.

The aim of this research is to develop a novel approach for offloading tasks to handle the requirements of latency-sensitive IoT applications and efficiently utilizing the

resources in Edge-Cloud environments. Also, the proposed approach is used for scheduling offloading tasks in order to reduce the overall service time and improve resource utilization. The contributions reported in this study can be summarized as follows:

- Introduce an Edge-Cloud system architecture that includes the required components to support scheduling offloading tasks of IoT applications;
- Propose a novel approach that adopts the fuzzy logic technique, which considers application characteristics (e.g., CPU demand, network demand and delay sensitivity) as well as resource utilization and resource heterogeneity in order to minimize the overall time of latency-sensitive applications;
- Provide a set of algorithms for offloading tasks scheduling in order to enhance service time and resource utilization in Edge-Cloud environments;
- An evaluation of the proposed approach for scheduling offloading tasks decisions within the edge-cloud system, considering application characteristics, resource utilization and resource heterogeneity.

The remainder of this paper is organized as follows: A thorough discussion of the related work is presented in "Related work" section. "Proposed system architecture" section presents the system architecture that supports scheduling offloading tasks of IoT applications, followed by the descriptions of the required components and their interactions within the proposed architecture. "Tasks scheduling approach for minimum latency" section presents a scheduling approach for minimum latency. "Task selection phase based on resource type" section presents the task selection phase based on resource type, followed by the experimental implementation and results discussion in "Implementation" section. Finally, "Conclusion and future work" section concludes this paper and discusses the future work.

Related work

Computation offloading is not a new paradigm; it is widely used in the area of Cloud Computing. Offloading transfers computations from the resource-limited mobile device to resource-rich Cloud nodes in order to improve the execution performance of mobile applications and the holistic power efficiency. Users devices are evenly located at the edge of the network. They could offload computation to Edge and Cloud nodes via Wireless Local Area Network (WLAN) or 4G/5G networks. Generally, if a single edge node is insufficient to deal with the surging workloads, other edge nodes or cloud nodes are ready for assisting such an application. This is a practical solution to support IoT applications by transferring heavy computation tasks to powerful servers in the Edge-Cloud system. Also, it is used to overcome the limitations of IoT devices in terms of computation power (e.g., CPU and memory) and insufficient battery. It is one of the most important enabling techniques of IoT, because it allows performing a sophisticated computational task more than their capacity [29]. Thus, the decisions of computational offloading in the context of IoT can be summarized as follows:

- First, whether the IoT device decides to offload a computational task or not. In this case, several factors could be considered, such as the required computational power and transferred data.
- Second, if there is a need for offloading, do partial offloading or full offloading. *Partial offloading* refers to the part of the tasks that will be processed locally at the IoT device and other parts in the Edge-Cloud servers. Also, factors such as task dependency and task priority can be considered in this case. *Full offloading* means, the whole application will be processed remotely in the Edge-Cloud servers [30].

In terms of the objectives of computation offloading in the context of Edge Computing, it can be classified into two categories; objectives that focus on application characteristics and objectives that focus on Edge-Cloud resources. Several studies [31–34] had aimed to minimize service latency, energy consumption and mandatory cost, as well as maximize total revenue and resource utilization. In fact, scheduling offloading tasks is a challenging issue in the Edge-Cloud Computing paradigm, since it considers several trade-offs from application requirements (e.g., reduce latency) and system requirements (e.g., maximize resource utilization). Thus, developing an efficient resource management technique, that meets the requirements of both application and system, is an active area of research.

In the following subsections, some of the studies conducted on task offloading in Edge-Cloud environments to reduce the latency and maximize resource utilization, are reviewed and discussed.

Task offloading based on application characteristics

As stated in [5, 20], scheduling offloaded tasks that focused on application characteristics is considered significantly important, especially, with the increase of IoT applications. Therefore, this subsection presents the conducted studies on task offloading, which mainly focuses on application characteristics including (computation and communication demands, and latency-sensitivity).

Computation and communication demands

There are many ongoing research projects focusing on the task computation and communication demands of IoT applications. For example, Wang et al. [35] proposed an online approximation algorithm that mainly objective to balance the load and minimizing resource utilization

in order to enhance application performance. This work considers the attributes of computational and communications for homogenous resources, without considering the service latency. Rodrigues et al. [36], presented a hyper method for minimizing service latency and reduce power consumption. This method aims to reduce the communication and computational delays by migrating the VM to an unloaded server. The authors investigate the impact of tasks computational and communication demands. They evaluate their approach under realistic conditions by mathematical modelling. However, their method does not consider the application delay constraints as well as the offloading to the cloud. Deng et al. [37], proposed an approximate approach for minimizing network latency and power consumption by allocating workload between Fog and Cloud. However, their approach does not optimize the trade-off between all mentioned objectives (e.g., computational delay and resource utilization).

Zeng et al. [38] designed a strategy for task offloading that aims to minimize the completion time. In their work, both computation time and transmission time are considered. Also, the authors investigate the impact of other factors such as I/O interrupt requests and storage activities. However, delay-constraints applications and resource heterogeneity are not considered in their work. Fan et al. [39] designed an allocation scheme that aims to minimize service latency for IoT applications, by taking into account both computation and communication delays. Furthermore, the authors investigate the impact of the overloaded VM on processing time, and they evaluated their work with different types of applications. However, the proposed method does not show the effectiveness of the heterogeneity of the VMs in terms of service time and also does not consider the latency-sensitive application.

Latency sensitivity

In terms of application latency-sensitivity, a number of studies are conducted in order to enhance the overall service time in the Edge-Cloud environment. For instance, Mahmud et al. [20] proposed a latency-aware policy that aims to meet the required deadlines for offloading tasks. This approach considering task dependency as well as the computational and communication requirements. Also, the resource utilization at the edge level is considered. However, the issue of resource heterogeneity is not addressed in their work. Azizi et al. [40] designed a priority-based service placement policy that prioritizes tasks with deadlines; thus, the nearest deadlines are scheduled first. Further, their work considers both computational and communication demands. However, their evaluation does not address the issue when the system has multi IoT devices with different resource utilization. Sonmez et al. [41] presented an approach for task offloading that targets latency-sensitive applications. This approach is based on fuzzy logic, which focused on delay as a key factor along with computational and communication demands. Nevertheless, in this approach resource heterogeneity is not considered.

Task offloading based on edge-cloud resources

This subsection presents the literature on offloading tasks and mainly focused on resource utilization and resource heterogeneity as main objectives.

Resource utilization

Scheduling offloading tasks based on resource utilization or resource heterogeneity has received considerable critical attention from many researchers. For example, Nan et al. [42] developed an online optimization algorithm for offloading tasks that aim to minimize the cost of renting Cloud services by utilizing resources at the edge using the Lyapunov technique. Further, their algorithm guarantees the availability of edge resources and ensures processing the task within the required time. Yet, this algorithm does not consider the impact of computational and communication demands for latency-sensitive applications. Xu et al. [43] proposed a model for resource allocation that aims to maximize resource utilization and reduce task execution latency, as well as, reducing the dependability on the cloud in order to minimize Cloud cost. However, this work only considers resource utilization and does not refer to resource heterogeneity. Besides, application uploading and downloading data are not addressed in their work, which plays a significant role in overall service time. Li and Wang [44] introduced a placement approach that aims to reduce edge nodes' energy consumption and maximize resource utilization. They evaluated the proposed algorithm through applied numerical analysis based on the Shanghai Telecom dataset. However, their work does not provide any information regarding the application characteristics (e.g., computation, communication and delay-sensitivity).

Resource heterogeneity

Resource heterogeneity for the offloading decision plays a critical role to enhance the performance of service time in the Edge-Cloud environment. Thus, a number of studies have investigated the impact of resource heterogeneity on service time. For instance, Scoca et al. [45] proposed a scour-based algorithm for scheduling offloading tasks that considers both computation and communication parameters. Furthermore, their algorithm considers a heterogeneous VMs and sorts heavy tasks to be allocated to the most powerful VM. However, their algorithm does not consider server utilization as key parameters, which could affect the performance of service time. Roy et al. [46] proposed a strategy for task allocation that allocates different application tasks to an appropriate edge server by

considering resource heterogeneity. This approach aims to reduce the execution latency as well as balancing the load between edge nodes. Yet, task communication time is not considered in this approach. Taneja et al. [47] proposed a resource-aware placement for IoT offloading tasks. Their approach ranks the resources at the edge with their capabilities and then assigns tasks to a suitable server based on the task's requirements (e.g., CPU, RAM and Bandwidth). However, this method focused on improving the performance of application service time, but without explicitly considering application latency-sensitivity.

With the dynamicity of IoT workload demands, Edge-Cloud service providers aimed to find a balance between utilizing Edge-Cloud resources efficiently and satisfying the QoS objectives of IoT applications. Consequently, designing a new task offloading mechanism can contribute to enhancing resource utilization and supporting the latency-sensitive application requirements in the Edge-Cloud environment.

Proposed system architecture

As illustrated in Fig. 1, the edge-cloud system from bottom to the top consists of three layers/tiers: IoT devices (end-user devices), multiple Edge Computing nodes and the Cloud (service provider). The IoT level is composed of a group of connected devices (e.g., smartphones, self-driving cars, smart CCTV); these devices have different applications where each application has several tasks (e.g., smart CCTV [48] application consists of movement dedication and face recognition). These services can be deployed and executed in different computing resources (connected Edge node, other Edge nodes or Cloud), where the infrastructure manager and service providers have to decide where to run these services.

In this proposed system, at the Edge level, each Edge Computing node is a micro datacenter with a virtualized environment. It has been placed close to the connected IoT devices at the base station or Wi-Fi access point. These edge nodes have been distributed geographically and could be owned by the same Cloud provider or other brokers [49]. Note that, it has limited computational resources compared to the resources in the cloud. Each edge node has a node manager that can manage computational resources and application services that run on. All the edge nodes are connected to the *Edge Controller*.

The offloading tasks can be achieved when the IoT devices decide to process the task remotely in Edge-Cloud environments. Applications running on IoT devices can send their offloadable tasks to be processed by the Edge-Cloud system through their associated Edge node. We assume that each IoT application is deployed in a Virtual Machine (VM) in the edge node and the cloud. IoT devices offload tasks which belong to a predefined set of applications, these tasks are varied in term of the computational requirement (task length) and communication demand (amount of transferred data). It is assumed that tasks are already offloaded from the IoT devices, and each task is independent; thus, the dependency between the tasks is not addressed in this paper. The locations of IoT devices are important for the service time performance because it is assumed that each location is covered by a dedicated wireless access point with an Edge node and the IoT devices connect to the related WLAN when they move to the covered location.

The associated Edge can process IoT tasks and also can be processed collaboratively with other edge nodes or the cloud, based on Edge orchestrator decisions. For example, if an IoT application is located in an edge node faraway from its connected edge, its data traffic has to be routed to it via a longer path in the Edge-Cloud system. At the cloud level, a massive amount of resources that enable IoT applications' tasks to be processed and stored.

The proposed architecture is just a possible implementation of other architectures in the literature such as [3, 43, 50]. The main difference in the proposed architecture is the introduced layer between the edge nodes and the cloud. This layer is responsible for managing and assigning offloading tasks to the edge nodes. In practice, the edge computing nodes are connected through an intermediate layer (for example, backbone router) that serves as a central control manager to monitor them. In addition, software-defined network (SDN) technology can be utilized at this layer to monitor and manage the application services between the edge computing nodes depending on data gathered, in which where SDN has a global view of the network and is capable of making more efficient and precise decisions [51]. More details about the required components and their interactions within the proposed architecture are as follow.

Edge controller

Edge Controller (EC) is designed similar to [33, 52, 53], some studies called *Edge Orchestrator*, which is a centralized component that responsible for planning, deploying and managing application services in the Edge-Cloud system. EC communicates with other components in the architecture to know the status of resources in the system (e.g., available and used), the number of IoT devices, their applications' tasks and where IoT tasks have been allocated (e.g., Edge or Cloud). EC consists of the following components: *Application Manager, Infrastructure Manager, Monitoring and Planner*. The location of the *Edge Controller* can be deployed in any layer between Edge and Cloud. For example, in [54], EC act as an independent entity in the edge layer that manage all the edge nodes in its control. It is also responsible for scheduling the offloading tasks in order to satisfy applications' users and Edge-Cloud System requirements. The EC is

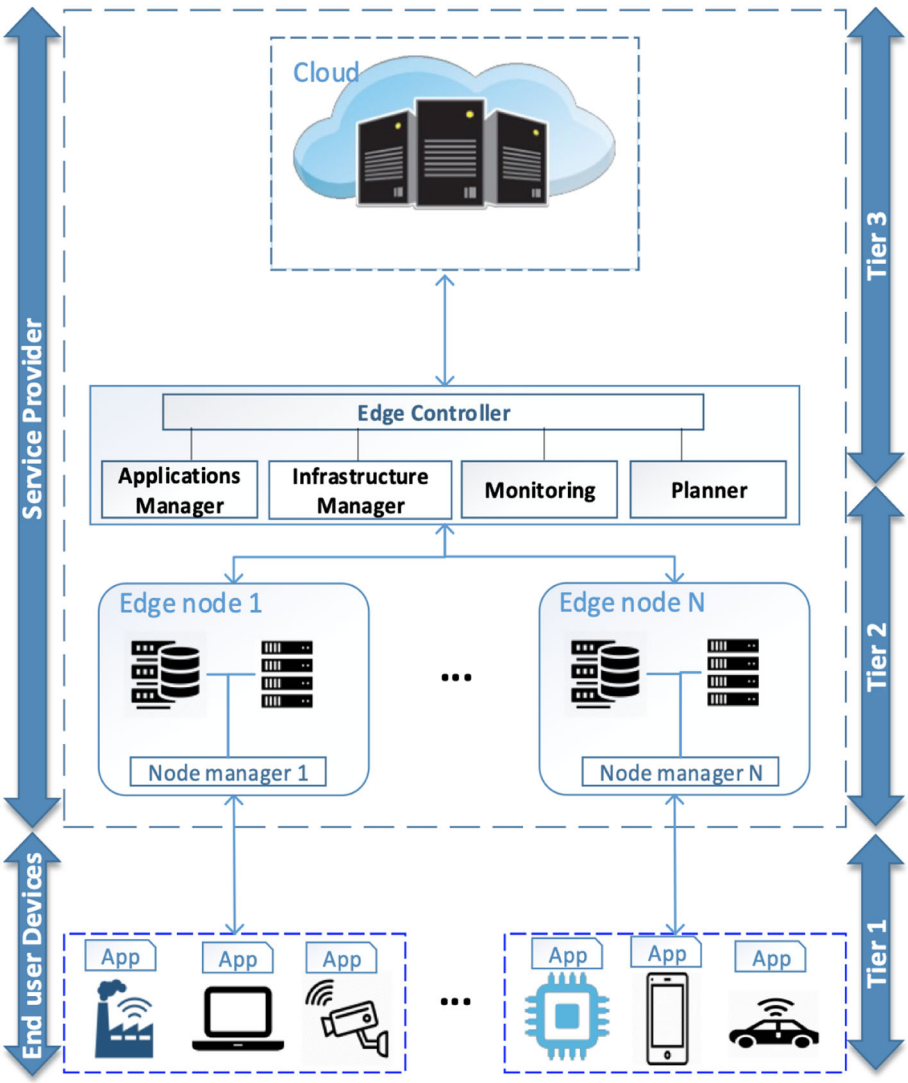

Fig. 1 An overview of Edge-Cloud system

synchronizing its data with the centralized Cloud because if there is any failure, other edge nodes can take EC responsibility from the cloud [55, 56].

Application manager

The application manager is responsible for managing applications running in the Edge-Cloud system. This includes requirements of application tasks, such as the amount of data to be transferred, the amount of computational requirement (e.g., required CPU) and the latency constraints. Besides, the number of application users for each edge node.

Infrastructure manager

The role of the infrastructure manager is to be in charge of the physical resources in the entire Edge-Cloud system. For instance, processors, networking and the connected IoT devices for all edge nodes. As mentioned earlier, Edge-Cloud is a virtualized environment; thus, this component responsible for the VMs as well. In the context of this research, this component provides the EC with the utilization level of the VMs.

Monitoring

The main responsibility of this component is to monitoring application tasks (e.g., computational delay and communication delay) and computational resources (e.g., CPU utilization) during the execution of applications' tasks in the Edge-Cloud system. Furthermore, detecting the tasks' failures due to network issues or the shortage of computational resources.

Planner

The main role of this component is to propose the scheduling policy of the offloading tasks in the Edge-Cloud system and the location where they will be placed (e.g., local edge, other edges or the cloud). In the context of this research, the proposed approach for offloading tasks works on this component and passes its results to EC for execution.

Tasks scheduling approach for minimum latency

In the Edge-Cloud environment, IoT devices produce a stream of incoming offloading tasks that differ in terms of their computation and network demand. This would require an efficient task scheduling technique that considers these differences in order to enhance the overall service performance and minimize the delay in the processing of offloaded tasks.

Therefore, the proposed approach supports the resource manager in the Edge-Cloud system regarding scheduling the offloading tasks in order to minimize the overall service time and improve the efficiency of Edge-Cloud resources. As shown in Fig. 2, the approach can be described using the MAPE method (Monitoring, Analyzing, Planning and Executing) to assign the tasks to appropriate resources and monitoring the system performance periodically. The proposed approach works in the Edge Controller (EC) as follows.

First, the edge controller receives the IoT devices information summary from edge computing nodes including the number of connected IoT devices, task length, the required number of cycles for task, and deadline requirement to complete task. In addition, the status of computation resources at each edge node is monitored periodically. Subsequently, the edge controller computes and decides the optimal strategy for scheduling and assigning the computation tasks to the best server (i.e. one of the edge computing server nodes or the cloud server) for execution based on the gathered information through Fuzzy logic and task scheduling algorithms. Later, we present the proposed algorithms in detail.

Fuzzy logic system

In this stage, the proposed approach will get the information of the offloading tasks and server utilization in order to determine the appropriate location of the offloading tasks, as depicted in Fig. 3. The following is a brief description of the process of fuzzy logic system.

1 **Fuzzy Input Variables:** In this step, the necessary inputs are specified for the fuzzy system. The required inputs are VM utilization at the edge, task length, the amount of data to be transferred for each task and delay sensitivity. All these variables are represented as a linguistic variable: Low, Medium and High, as depicted in Fig. 3. These categories represent the dynamic changing over Edge-Cloud infrastructure and the characteristics of applications' offloaded tasks.

   (a) **VM Utilization:** this parameter indicates the current utilization level of the VM hosted by the local edge server. Thus, we can know how much resource capacity is available on that VM. If it is highly utilized, then offloading to other edge servers or the cloud could be the solution, depending on the task characteristics in terms of computational, communication and latency sensitivity.

   (b) **Task Length:** this parameter represents the computational demand of the task; it measures by Million Instruction Per Second (MIPS). As the edge has a limited computational resource, heavy tasks might be

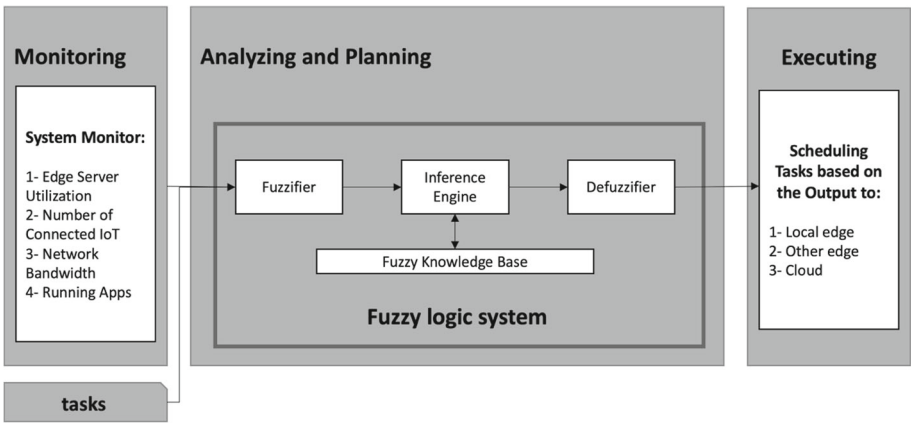

Fig. 2 The proposed approach of scheduling offloading tasks

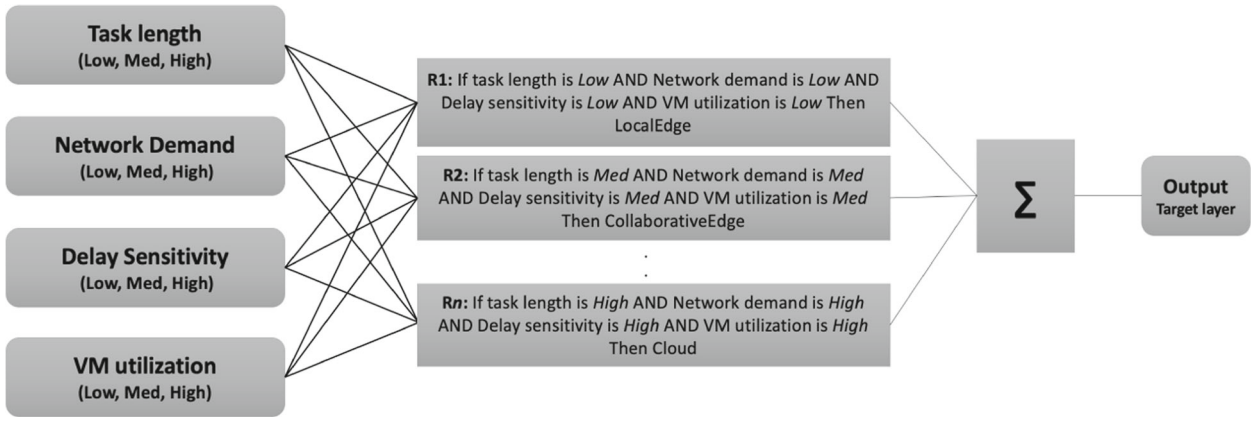

Fig. 3 Process of the proposed fuzzy logic system

appropriate to offloaded to the cloud and vice versa. However, we cannot take this parameter without considering other parameters such as VM utilization, communication demand and delay sensitivity.

(c) **Network Demand:** this parameter represents the required communication of the tasks for both uploading and downloading. It is an important measure for the offloading decision to consider where to offload the task to (local edge, other edge or cloud). For example, tasks of Augmented Reality (AR) applications that require video streaming must upload the request, then do some processing (e.g., 3D rendering, image processing, etc.), and then receive the results as a video stream. This requires transferring a high amount of data for uploading and downloading, which takes a significant amount of the total service time.

(d) **Delay Sensitivity of the Task:** this parameter refers to the sensitivity of the tasks to accept the level of latency (computation or communication delay). For example, some application has urgent tasks that require ultra-low latency; whereas, some tasks may accept some higher level of latency. This parameter could help the task scheduler to assign the tasks to an appropriate server within the Edge-Cloud system.

2 **Fuzzification:** In the fuzzification stage, the fuzzifier will take all the required values as numerical input from system infrastructure monitoring and incoming tasks. Then, assign each value to its predefined linguistic variables in the membership functions (e.g., Low, Medium and High). After that, fuzzy variables are combined and evaluates in the fuzzy rules-base to take the decision and produce the output in the defuzzification stage.

(a) **Fuzzy Membership Functions:** the fuzzy membership function is used to quantify the linguistic term for each fuzzy variable. In this research, four functions have been used (average VM utilization, task length, network bandwidth and delay sensitivity) and each function has three variables (Low, Medium, High). The values of each fuzzy variable are determined empirically based on a number of experiments similar to approaches used in [41, 57]. Figure 4 shows the four membership functions.

(b) **Fuzzy Rules-Base:** a fuzzy rules-base is composed of a set of fuzzy rules that similar to the reasoning process of human. It is a simple if-then rule that covers all the possible situations of the application characteristics and system conditions. These rules play critical directions to define the overall system performance. An example of the rules, if task length is *high* AND Network demand is *low* AND VM utilization is *high* AND the delay sensitivity is *high* THEN offloaded the task to the cloud. The output will be used in the defuzzification stage. Table 1 gives results examples of the system's fuzzy rules. The main aim is to provide low latency for the IoT applications by reducing the data movement from IoT device to the cloud and avoiding the overloaded node, which will affect the end to end service time.

3 **Defuzzification:** Defuzzification is the process to convert the fuzzy rules output to a specific value based on the output membership function. There are

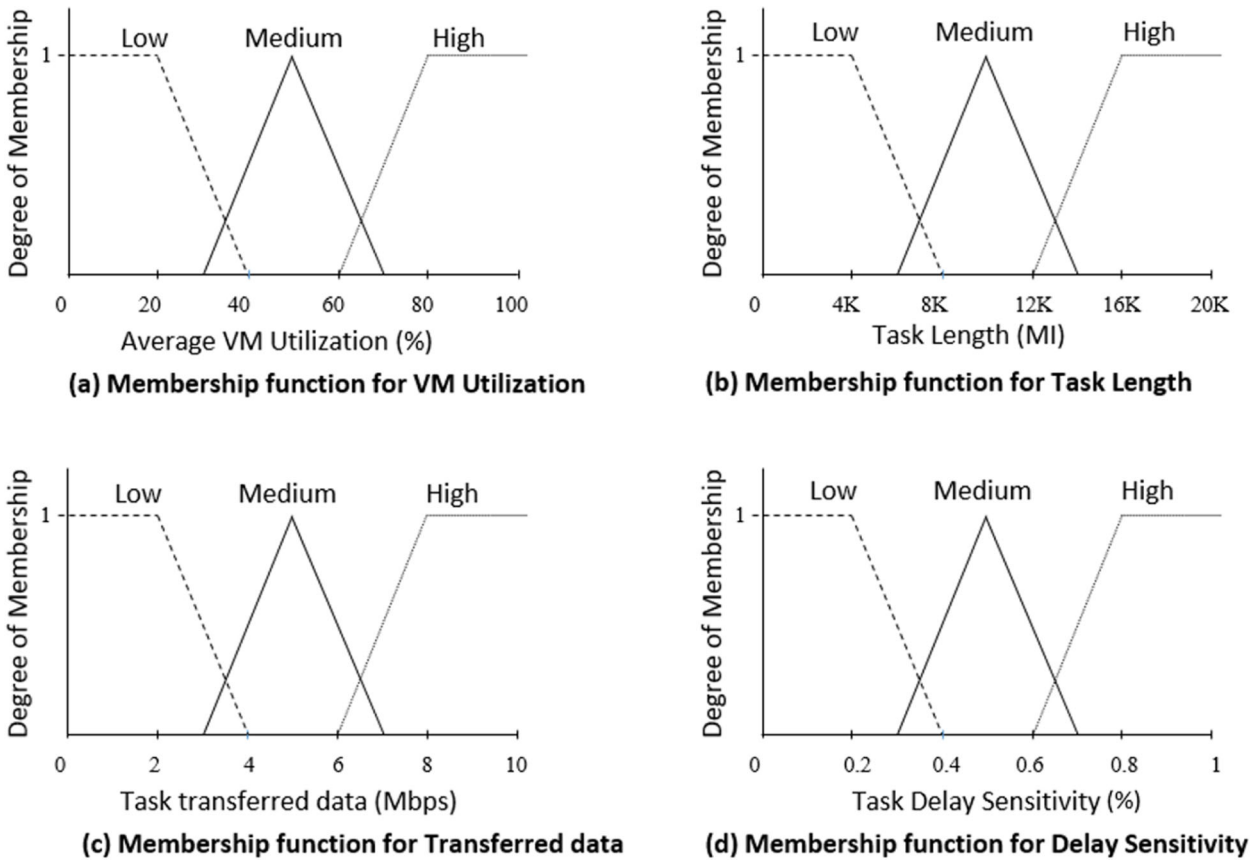

Fig. 4 The four membership functions

a range of ways to produce the output membership function in the fuzzy logic system and these examples of the often-used method (e.g., maximum, mean of maximum and centroid). This work adopts the maximum approach because our membership function has one maximum at a time. Figure 5 represents the output membership function of the fuzzy logic system. For example, if the output fuzzification process is 38, then $\mu^{LocalEdge}$ is 0.1 and $\mu^{CollaborativeEdge}$ is 0.4, the defuzzification process will take the maximum, and the task will be offloaded to the other collaborative edge node.

Algorithm 1 provides the detailed processes to derive the optimal target layer for offloading the computation tasks. First, the information of IoT devices and their applications' tasks are gathered which includes the number of connected IoT devices, task length, the required number of cycles for task, and deadline requirement to complete task. Besides, the edge computing nodes and the cloud server send their VM utilization. Then, as shown in line 3, the EC entity iterates over the computation tasks and uses the fuzzy logic function to quantify the linguistic term for each fuzzy variable as output. Afterwards, Fuzzy rules-based is utilized to determine the optimal target layer for each computation task wherever edge node or cloud server.

The computational complexity for this algorithm is $\mathcal{O}(t)$, where t denotes the number of parameters in applications' tasks (T). The step of sending the required information to the fuzzy logic system for each task requires $\mathcal{O}(t)$ time. According to the fuzzy inference logic, one of the three different output of fuzzy sets can be allocated to each task; thus, the time complexity of proposed fuzzy logic is $\mathcal{O}(n)$.

Task selection phase based on resource type

As shown in Fig. 6, the incoming tasks inter to the fuzzy logic system. Thus, the proposed fuzzy logic system is applied in order to decide the target layer to offload the task. The task scheduling algorithm will assign the tasks to the appropriate computational resources within Local Edge or Collaborative Edge based on the information from Infrastructure monitoring. This process runs on the EC, which is described in "Proposed system architecture" section. Further, we assume that each Edge node has a heterogeneity of computational resources.

Table 1 Fuzzy rules-base

	Input Variables			
Task Length (MIPS)	**Network Demand (Mbps)**	**VM Utilization**	**Delay Sensitivity**	**Output Decision**
Low	Low	Low	Low	Local Edge
Low	Low	Low	Medium	Local Edge
Low	Low	Medium	High	Local Edge
Low	Low	Medium	Low	Local Edge
Low	Medium	High	Medium	Local Edge
Low	Medium	High	High	Local Edge
Low	Medium	Low	Low	Local Edge
Low	Medium	Low	Medium	Local Edge
Medium	High	Medium	High	Other Edge
Medium	High	Medium	Low	Other Edge
Medium	High	High	Medium	Other Edge
Medium	High	High	High	Other Edge
Medium	Low	Low	Low	Local Edge
Medium	Low	Low	Medium	Other Edge
Medium	Low	Medium	High	Other Edge
Medium	Low	Medium	Low	Other Edge
High	Medium	High	Medium	Cloud
High	Medium	High	High	Cloud
High	Medium	Low	Low	Other Edge
High	Medium	Low	Medium	Other Edge
High	High	Medium	High	Cloud
High	High	Medium	Low	Cloud
High	High	High	Medium	Cloud
High	High	High	High	Cloud

Algorithm 2 shows the detailed processes to assign each computation task to the appropriate computational resources. The process for assigning computation tasks is as follows. Firstly, the set of application, computation tasks, and the available computational resources at edge node are gathered. Then, as shown in line 1, the computation tasks are sorted in descending order regrading their CPU requirements, in which the heavy tasks come first and the lightweight tasks come last. In addition, as shown in line 2, the available computational resources (VMs) of edge node are sorted in descending order regrading their CPU capabilities (i.e., number of cores), where the most powerful VMs comes first. After that, the algorithm iterates over the ordered tasks and assigns each application task to the appropriate computational resources, where the heavy tasks could be assigned to the powerful VMs, shown in lines 3-11. This ensures that heavy tasks have the priority to be assigned to a powerful VM, thereby will produce less processing time.

The computational complexity for this algorithm is $\mathcal{O}(n^2)$, where n denotes the number of computation tasks. This can be analyzed as follow. Firstly, the computation tasks and computation resources are sorted, in which $\mathcal{O}(n^2)$ is the worst-case time complexity of the sorting. Then, the computation tasks of IoT applications are iterated to be assigned to computational resources, thereby $\mathcal{O}(n^2)$ is the time complexity. Consequently, the overall time complexity is $(\mathcal{O}(n^2)+ \mathcal{O}(n^2))$ which is $\mathcal{O}(n^2)$, where the constant is removed.

Implementation

The task offloading approach based on a fuzzy logic system that aims to enhance the end-to-end service time is introduced. This approach considered both tasks requirements (e.g., computational, network and delay) and the dynamicity of the edge-cloud system in terms of resource utilization. In order to evaluate this approach, a number of experiments have been conducted on EdgeCloudSim [58] and compared with other competitors' solutions. The EdgeCloudSim has been used since it provides the vital functionality of Edge-Cloud environment such as support offloading and users mobility [41, 45, 59, 60]. The process started with generating tasks of different IoT applications, then scheduling tasks in the Edge-Cloud system based on the proposed scheduling algorithms.

Approaches that dealt with offloading tasks using fuzzy logic are limited in the area of Edge Computing. Thus, the evaluation of our approach will be against the existing approaches. First is a utilization-based approach, which makes decisions on offloading tasks based on the server utilization level, by selecting the least-load machine for offloaded tasks. The aim of this approach is to utilize edge resources and make load balancing. This approach has been adopted in a number of studies due to the simplicity of its logic and the feasibility of its implementation [61, 62]. It is well suitable for the common situation, in which the number of applications and the execution time of tasks is both moderate. However, it doesn't consider task communication demand and application delay

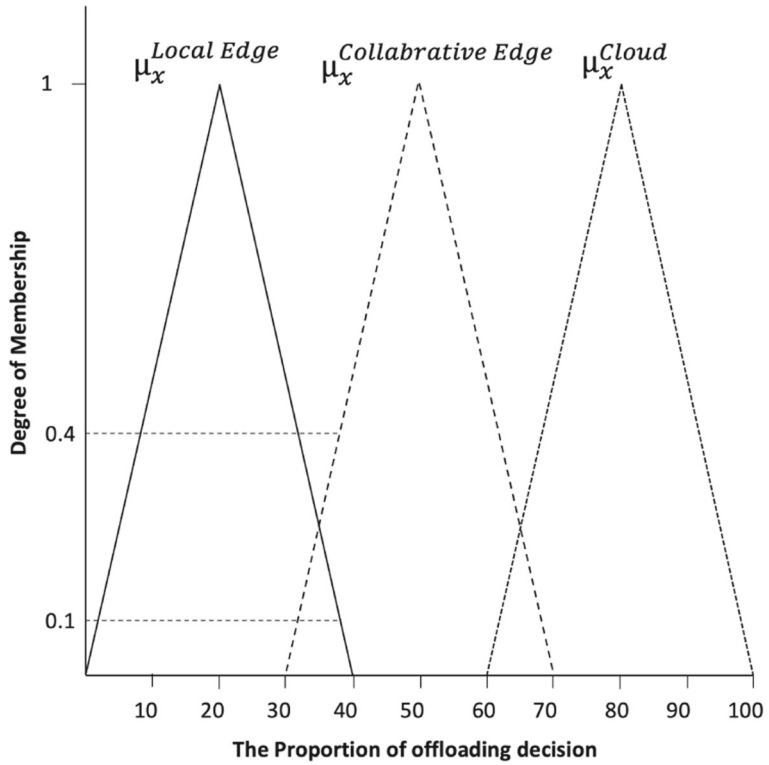

Fig. 5 The output membership function of the fuzzy logic system

sensitivity. Second, Flroes [63] proposed a task offloading approach based on fuzzy logic for IoT applications. This approach aims to decide whether to offload to the cloud or perform the tasks in end devices at the edge layer. However, this approach neglected the utilization of the Edge-Cloud resources, which could cause an overloaded VM and lead to significant latency. Finally, Snomes [41] proposed tasks offloading approach that consider both applications tasks requirements and resource utilization using a fuzzy logic system. However, this approach focused on homogeneous resources, whereas the Edge-Cloud system is composed of heterogeneous resources. Moreover, their solution decides whether to offload to the local edge or the cloud, whereas our proposed approach considers the heterogeneity of resources as well as the available resources in other nearby edge nodes. All of these approaches have been implemented in the simulation tool in order to evaluate and compare them with the proposed approach.

Algorithm 1: Fuzzy Logic for Offloading to the Target Layer

INPUT: Applications' tasks T_i with their parameters T_{Length}, $T_{Network}$ and T_{Delay}
OUTPUT: Select computational resources at the target layer: $R_{Local\ Edge}$, $R_{Collaborative\ Edge}$ and R_{Cloud}

for all Tasks in T_i **do**
  Read VM average utilization VM_u
  F = FuzzyLogicSystem(T_i^{Length}, $T_i^{Network}$, T_i^{Delay} ,VM_u);
  if $F \leq F_{Low}$ **then**
    Allocate T_i on $R_{Local\ Edge}$
  else
    if $F \leq F_{Med}$ **then**
      Allocate T_i on $R_{Collaborative\ Edge}$
    else
      Allocate T_i on R_{Cloud}
    end if
  end if
end for

Simulation set-up

In the Edge-Cloud environment, there are a number of IoT/mobile devices that have a number of applications. These applications consist of different tasks which require to be processed in the Edge-Cloud resources. Edge nodes are distributed closer to end devices, and we assume each edge node covers a specific area. IoT devices connect to the nearest edge node through WLAN and then can send the offloaded tasks. Also, we assume that each node has a node manager and all edge nodes are managed by the

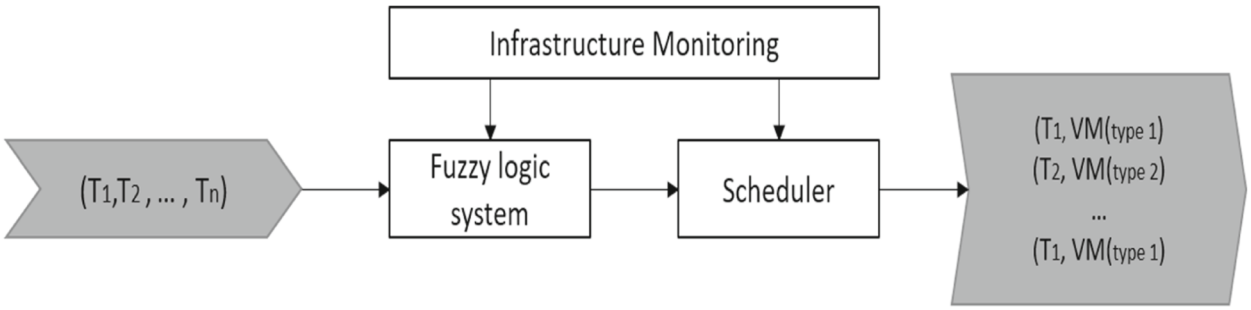

Fig. 6 Task selection phase

Algorithm 2: Task Scheduling Algorithm

INPUT: Set of Applications A_i, Set of Tasks T_j and Set of Computational Resources at Edge node R_c
OUTPUT: Assign Task T_{ij} to Target Computational resources R_c
Sort task list T in descending order of task length.
Sort computational resources list R in descending order of resource capacity.
for all Application in A_i **do**
  while T_{ij} has not been scheduled **do**
    if $T_{ij}^{CPU} \leq R_c^{CPU}$ **then**
      Assign T_{ij} on R_c
      T_{ij} ++
    else
      R_c ++
    end if
  end while
end for

EC, described in "Proposed system architecture" section. In our experiments, we have three edge nodes and a variable number of IoT devices (from 200 to 2000) dispersed and mobile between the three nodes. Table 2 represents the key parameters of our simulations.

Table 2 Simulation key parameters

Parameters	Values
Simulation Time	30 minutes
Warm-up Period	3 minutes
Number of Iterations	5
Number of IoT devices	200-2000
Number of Edge Nodes	3
Number of VM per edge server	8
Number of VM in Cloud	Not limited
Probability of selecting location	Equal

Additionally, we assume that the VMs on each edge node are heterogeneous. Table 3 shows the configurations of the VMs that were considered in the experiments. These configurations are based on Rackspace, which provides a wide range of VM types [45, 64]. Two types of VMs are used with different capabilities to supports the end devices with computational resources. The first type of VMs has two cores Intel Xen CPU, and the second type of VMs has four cores Intel Xen CPU.

IoT applications generate different offloading tasks in terms of CPU and network load. To evaluate our approach, we need different applications with different computational and communication demands. Several research studies generate random tasks in their experiments [36, 65]. Table 4 summarized the main characteristics of the four applications that are used in this experiment similar to the works presented in [41, 45]. Task Length refers to require CPU resources for the task in Million Instructions (MI) unit. Uploading and downloading data determines the amount of data to send/receive for each task from the IoT device to the Edge-Cloud system. Delay sensitivity refers to the acceptance level of delay sensitivity.

Figure 7 shows a snapshot of the simulation results for one scenario. Each scenario takes one approach (e.g., Utilization-Based) with a specific number of devices. All the experiments have been repeated five times (iterations) for each scenario, and the statistical analysis has been performed to consider the mean values of the results in order to avoid any anomalies from the simulation results.

Results and discussion

This section presents the quantitative evaluation of the proposed approach compared to other related works'

Table 3 Configurations of VMs

VM Type	CPU Cores	MIPS	Storage
Medium VM	2 vCPUs	10000	50000
Large VM	4 vCPUs	20000	100000

Table 4 Application characteristics

Apps	Task Length (MI)	Uploading Data (KB)	Downloading Data (KB)	Delay Sensitivity
Augmented Reality	9000	1500	25	0.9
Health Care	3000	20	1250	0.7
Compute Intensive	45000	2500	200	0.1
Infotainment	15000	25	1000	0.3

algorithms (e.g., Utilization-Based, Sonmez and Flores). The simulation results consist of the average service time, average processing delay, average network delay, average task failure and average VM utilization. The service time of each task will depend on the location of processing, which can be one of the following: 1) *Local Edge*, the overall service time consists of WLAN time and processing time. 2) *Collaborative Edge*, the overall service time will be WLAN/Metropolitan-Area Network (MAN) time and processing time. 3) *Cloud*, the overall service time consist of WLAN/MAN/Wide Area Network (WAN) time and processing time in the cloud. After that, we take the average for all tasks in each scenario based on the following equation:

$$Service\ Time = \frac{\sum T_{processing\ time} + \sum T_{network\ time}}{Number\ of\ Tasks}$$

The main performance metric is the service time, since the end-to-end service time of an offloading task is most significant for IoT latency-sensitive application. Figure 8 shows the average service time for the 4 different approaches versus number of IoT devices, in which the service time is composed of processing and network time. The purpose of the experiments was to enhance the resources management in Edge-Cloud system in order to reduce latency for IoT applications. It is seen from the figure that all the approaches have nearly the same performance when the system is unloaded. However, with increasing the number of IoT devices, the service time of the proposed approach remains steady compared to the other approaches. Moreover, the service time for Flores algorithm increases rapidly after the number of IoT devices exceeds 1400. Whereas, Utilization-Based and Sonmez algorithms nearly have the same performance. This is attributed to the usability of VM utilization in task scheduling policy which will avoid processing delays and then produce less service time.

The average network time of all approaches related to a different numbers of IoT devices is shown in Fig. 9. It is observed from the figure that all the approaches have the same network time when the system is stable, whereas, the results are differentiated after the system is overloaded. Also, the utilization-based approach can provide the lowest network time compared to the other approaches. This is due to the demands associated with increased time regarding the processes of communication when the task might be sent to the cloud for execution, particularly with the larger number of IoT devices.

Similarly, Fig. 10 assesses the processing time of the four approaches versus different numbers of IoT devices. It is seen from the figure that when the number of IoT

```
----------------------------------------------------------------------
Scenario started at 18/02/2020 17:36:19
Scenario: TWO_TIER_WITH_EO - Policy: UTILIZATION_BASED - #iteration: 1
Duration: 33.0 min (warm up period: 3.0 min) - #devices: 200
Creating tasks...Done,
Creating device locations...Done.
SimManager is starting...Done.
.........10%.........20%.........30%30%.........40%.........50%.........60%.........70%.........80%
# of tasks (Edge/Cloud/Mobile): 48869(48869/0/0)
# of failed tasks (Edge/Cloud/Mobile): 222(222/0/0)
# of completed tasks (Edge/Cloud/Mobile): 48647(48647/0/0)
# of uncompleted tasks (Edge/Cloud/Mobile): 30(26/0/4)
# of failed tasks due to vm capacity (Edge/Cloud/Mobile): 0(0/0/0)
# of failed tasks due to Mobility/Network(WLAN/MAN/WAN): 222/0(0/0/0)
percentage of failed tasks: 0.454276%
average service time: 1.134301 seconds. (on Edge: 1.134301, on Cloud: NaN, on Mobile: NaN)
average processing time: 1.072191 seconds. (on Edge: 1.072191, on Cloud: NaN, on Mobile: NaN)
average network delay: 0.062110 seconds. (LAN delay: 0.051209, MAN delay: 0.012486, WAN delay: NaN)
average server utilization Edge/Cloud/Mobile: 2.531046/0.000000/0.000000
average cost: 0.0$
Scenario finished at 18/02/2020 17:36:23. It took 4 Seconds
----------------------------------------------------------------------
```

Fig. 7 A snapshot of the simulation results for one scenario

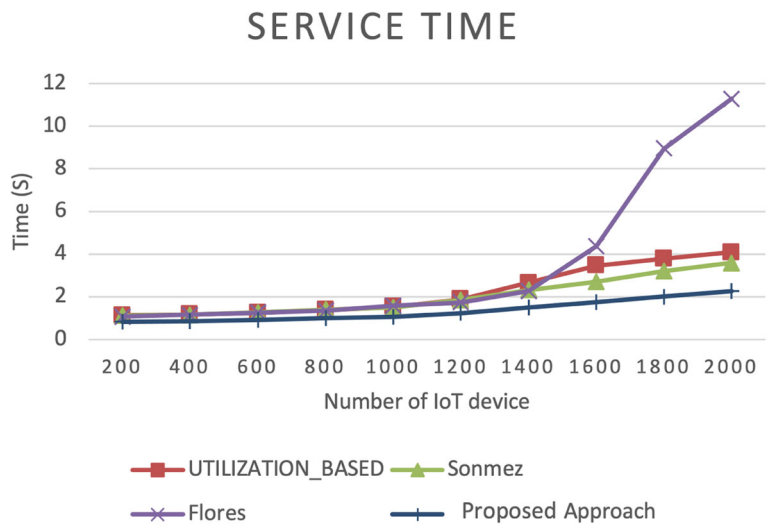

Fig. 8 The service time of the proposed approach with other related approaches

devices is less than 1400, all the approaches approximately have the same processing time. However, as the number of IoT devices further increases, the performance of the Flores approach decorates in comparison with Sonmez and Utilization-based approaches while our proposed model remains steady and outperforms the others. This is traced to the shorter processing time as the computation tasks are assigned to the appropriate resources. Whereas, Flores approach offloads the task to the edge whenever possible without considering whether the resource is overloaded, thereby leads to an increase in the processing time.

In the EdgeCloudSim, the task failure can be happened due to different reasons such as the lack of computational resources at the VM (e.g., overloaded VM) and congested network. Therefore, task failure is considered as important performance metric in order to evaluate the offloading approach. In the following, the evaluation of task failures for our approach will be divided into two parts, *system stable* and *system overloaded*.

First, in the case of system stability, the proposed approach has the lowest percentage while the other approaches have nearly the same performance, and around 0.5% of tasks will fail, shown in Fig. 11. This is due to that our approach considers the required amount of data to be uploaded and downloaded. On the other hand, when the system load is high (the second part), it can be seen that the lowest task failure average is for the proposed approach, as shown in Fig. 12. Interestingly, there

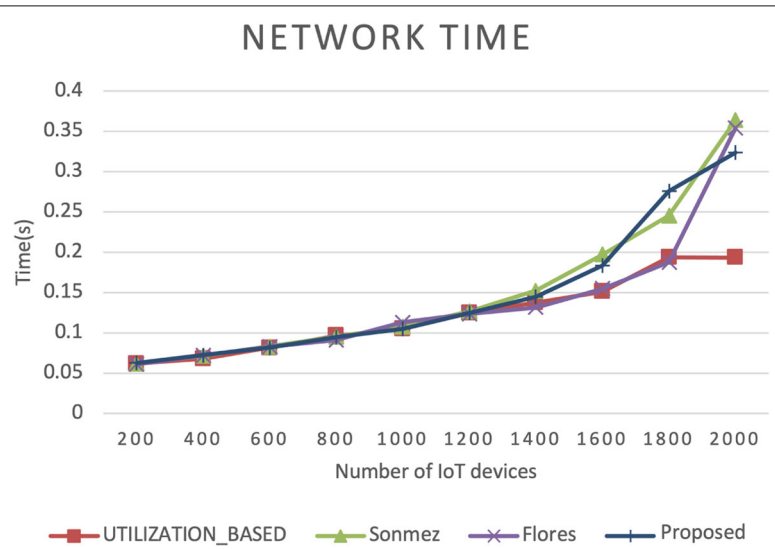

Fig. 9 The network time of the proposed approach with other related approaches

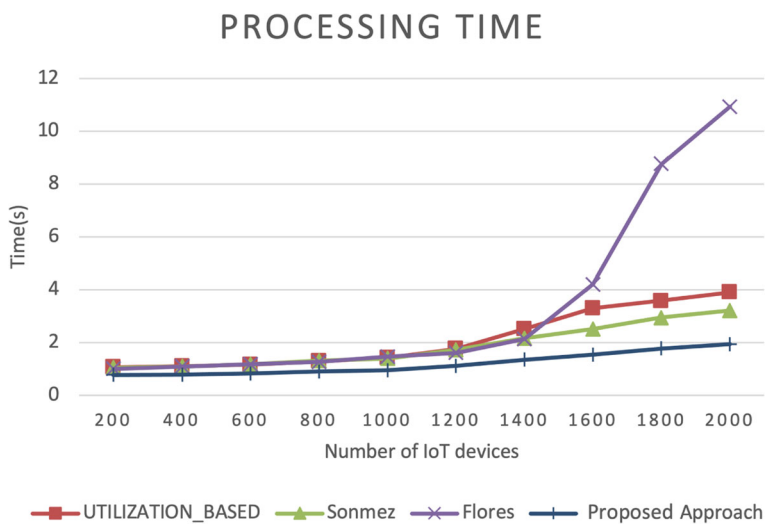

Fig. 10 The processing time of the proposed approach with other related approaches

were slight differences between Utilization-Based and Flores for all number of IoT devices. When the system load is low, most of the tasks failures are due to network issues such as losses of the packet [9], but when the system is overloaded, task failures are due to the lack of computation (e.g., unsuccessful completion task) as well as network issues. In comparison, the proposed approach has the lowest task failures because it assigns the heavy tasks to the powerful VM as well as considers other factors (e.g., VM utilization, network demand and delay sensitivity).

Finally, the preliminary analysis of the average VM utilization at edge servers versus different number of IoT devices is shown in Fig. 13, where the system server IoT devices less than 1000. It can be seen that, the utilization level of all the approaches at 200 devices is similar, while its value is changed when the number of devices increased. In addition, the proposed approach is keeping the utilization level relatively low comparing to other approaches when the number of devices increased.

On the other hand, as shown in Fig. 14, when the system load is high, the proposed approach was the lowest compared to other algorithms. This is because it trades utilization for reduced service time. Also, it can be seen that Sonmez and the proposed approach were relatively similar and lower than the others. Flores was the highest and we can link that with results of failed tasks because it assigns the tasks to a highly utilized VM (overloaded VM). Moreover, the proposed approach succeeded in avoiding reaching the exponential deterioration when the computational resources reach their limit comparing to other existing approaches. When the resources reach their limit,

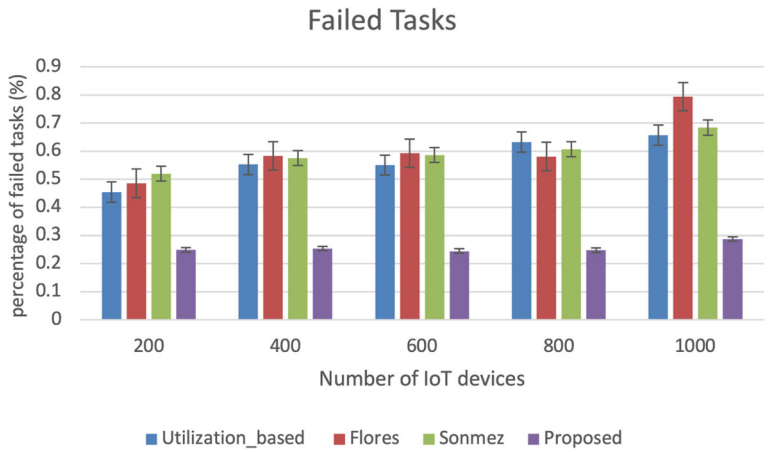

Fig. 11 Percentage of failed tasks of the proposed approach with other related approaches part 1

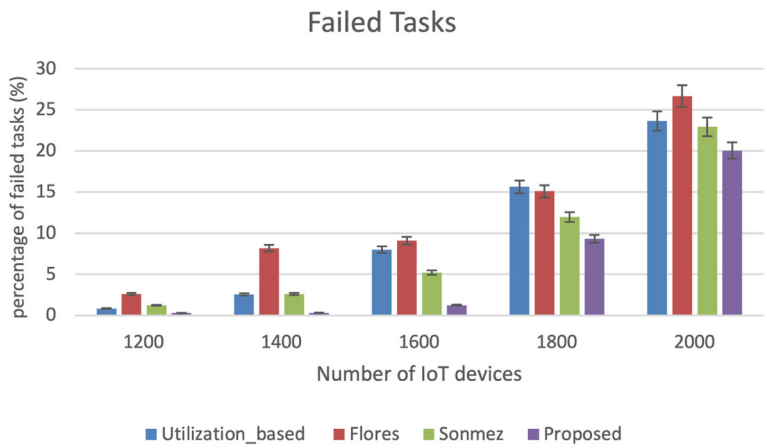

Fig. 12 Percentage of failed tasks of the proposed approach with other related approaches part 2

this will increase the overall service time and task failure due to insufficient computational resources.

Lessons learned

The proposed approach was compared against existing related works using a simulation tool called EdgeCloudSim, and it was evaluated in the domain of the edge-cloud environment; where it was found to improve the overall service time and task failure for latency-sensitive applications as well as effectively utilizing the edge-cloud resources.

However, EdgeCloudSim simulator has some limitations, in which only a single server queuing model is used for calculating the communication delay [66]. Therefore, all the available network technologies could not be represented and then could be limited to obtain the results. In addition, the VM migration process between EdgeCloud nodes is not handled where it can help to reduce the latency, improve the utilization and decrease the task failures.

On the other hand, there are also a few limitations on the proposed approach. In this research, we assume that we know the required parameters (e.g., task length, the amount of transferred data for uploading and downloading data) in advance, which might not always be accurate due to the impact of some other factors. For example, task length is not the only parameter that detriment the CPU time, other parameters such as retrieving data from memory and I/O could affect the CPU time. Additionally, the network time of transferred data affected by other factors such as network congestion. Therefore, methods such as Reinforcement Learning could be useful to measure the effectiveness of the offloading decision by observing each action and train the system to have accurate decisions. Moreover, this work trades utilization for reduced overall service time; thus, it could lead to wastage in

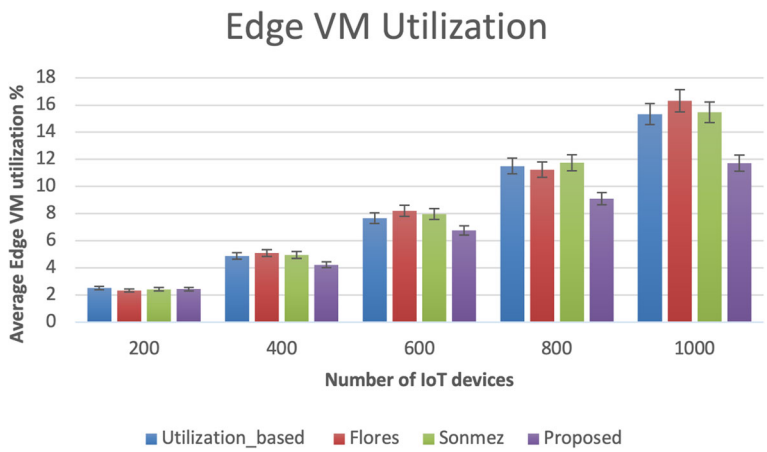

Fig. 13 Edge server utilization of the proposed approach with other related approaches part 1

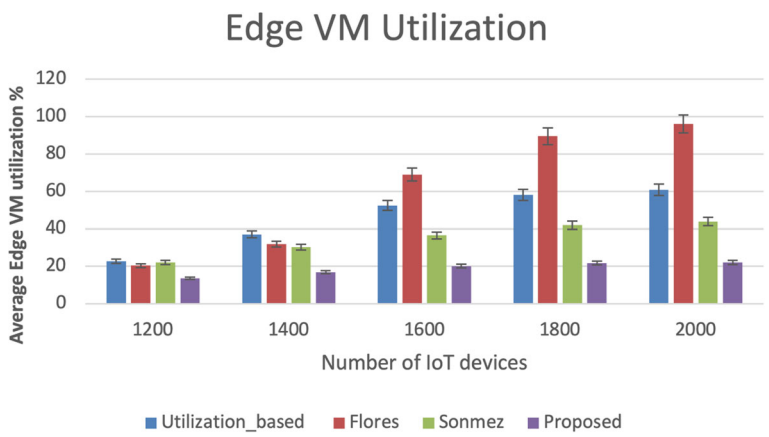

Fig. 14 Edge server utilization of the proposed approach with other related approaches part 2

both computation power and resources at the edge level. Other energy efficiency techniques (e.g., VM migration and auto-scaling horizontal/vertical) might be used in the future to overcome this issue and strike a balance between satisfying the demands of applications and utilizing the Edge/Cloud computational resources efficiently.

Conclusion and future work

This paper has presented and evaluated a novel task offloading approach for latency-sensitivity IoT applications in the Edge-Cloud systems. This approach considered application characteristics (e.g., CPU demand, network demand and delay sensitivity) as well as the dynamicity and heterogeneity of resources in order to minimize the overall service time and enhance resource utilization. Moreover, it considered different types of computational resources which represent the real-world scenario. Different approaches to schedule offloading tasks are simulated in order to evaluate the proposed approach. The obtained results show that the scheduling algorithms of offloading tasks not considering application characteristics and system behavior could lead to service time degradation for latency-sensitive applications. Moreover, the proposed approach works effectively with task offloading more than other related approaches in terms of overall service time and resource utilization. It can also reduce the overall task failures due to issues in both network and computational resources.

As a part of future work, we intend to extend our approach by considering more computational resources such as different Graphic Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs), since there are many applications for AR/VR and video gaming requiring intensive computational in order to process their tasks. Also, task dependency plays an essential factor in affecting the decision of scheduling tasks. Thus, this work can be extended to consider tasks dependency in the process of scheduling offloading tasks. Tasks dependency and the intercommunication between tasks can be represented as Direct Acyclic Graph (DAG), which can be modelled within the proposed approach to enhance the overall service time of latency-sensitive applications.

Acknowledgements
The authors would like to acknowledge the Deanship of Scientific Research, Taibah University, Al-Madinah, Saudi Arabia, for providing research resources and equipment.
In addition, the authors would like to thank the Deanship of Scientific Research, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia, for supporting this work.

Authors' contributions
Conceptualization, Data curation, Formal analysis, Methodology, Resources, Software, Visualization and Writing - original draft: J.Almutairi and M.Aldossary. Supervision, Validation, Writing - review and editing M.Aldossary. All authors have read and agreed to the published version of the manuscript.

Authors' information
Jaber Almutairi received his BSc degree in Computer Science from Taibah University, Medina, Saudi Arabia in 2012 and his MSc degree in Advanced Computer Science (Cloud Computing) and PhD from University of Leeds, Leeds, UK in 2016 and 2020. His research interests include resource management and simulation in the edge computing environment and applications of the internet of things. Dr Almutairi is currently Assistant professor with the Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah, Saudi Arabia.
Mohammad Aldossary received his B.Sc. degree in Computer Science from King Saud University, Riyadh, Saudi Arabia in 2009 and the M.S. degree in Computer Science from Southern Polytechnic State University, Georgia, USA in 2013. He was awarded a Ph.D. degree in Computer Science from the University of Leeds, UK, in 2019. His main research areas focus on Distributed Systems, including Cloud Computing, Fog Computing, Edge Computing, System Architectures, Resource Management, and Energy Efficiency. Dr Aldossary is currently an assistant professor in the Computer Science department at Prince Sattam bin Abdulaziz University, Saudi Arabia.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Authors details
[1]Department of Computer Science, College of Computer Science and Engineering, Taibah University, Al-Madinah, Saudi Arabia. [2]Department of Computer Science, College of Arts and Science, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia.

Received: 23 November 2020 Accepted: 30 March 2021
Published online: 20 April 2021

References

1. Rababah B, Alam T, Eskicioglu R (2020) The next generation internet of things architecture towards distributed intelligence: Reviews, applications, and research challenges. J Telecommun Electron Comput Eng 12(2)
2. Perera C, Zaslavsky A, Christen P, Georgakopoulos D (2014) Sensing as a service model for smart cities supported by internet of things. Trans Emerg Telecommun Technol 25(1):81–93
3. Vaquero LM, Rodero-Merino L (2014) Finding your way in the fog: Towards a comprehensive definition of fog computing. ACM SIGCOMM Comput Commun Rev 44(5):27–32
4. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of Things (IoT): A vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660
5. Shekhar S, Gokhale A (2017) Dynamic resource management across cloud-edge resources for performance-sensitive applications. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, Madrid. pp 707–710
6. Yousefpour A, Fung C, Nguyen T, Kadiyala K, Jalali F, Niakanlahiji A, Kong J, Jue JP (2019) All one needs to know about fog computing and related edge computing paradigms: A complete survey. J Syst Archit 98:289–330
7. Elgendy I, Zhang W, Liu C, Hsu C-H (2018) An efficient and secured framework for mobile cloud computing. IEEE Trans Cloud Comput 9(1):79–87. https://doi.org/10.1109/TCC.2018.2847347
8. Tyagi H, Kumar R (2020) Cloud computing for iot. In: Internet of Things (IoT). Springer, Berlin. pp 25–41
9. Sahni Y, Cao J, Zhang S, Yang L (2017) Edge mesh: A new paradigm to enable distributed intelligence in internet of things. IEEE access 5:16441–16458
10. Cong P, Zhou J, Li L, Cao K, Wei T, Li K (2020) A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud. ACM Comput Surv(CSUR) 53(2):1–44
11. Elgendy IA, Zhang W, Tian Y-C, Li K (2019) Resource allocation and computation offloading with data security for mobile edge computing. Futur Gener Comput Syst 100:531–541
12. Zhang W-Z, Elgendy IA, Hammad M, Iliyasu AM, Du X, Guizani M, Abd El-Latif AA (2020) Secure and optimized load balancing for multi-tier iot and edge-cloud computing systems. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2020.3042433
13. Elgendy IA, Zhang W-Z, Zeng Y, He H, Tian Y-C, Yang Y (2020) Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks. IEEE Trans Netw Serv Manag 17(4):2410–2422
14. Mahmud R, Ramamohanarao K, Buyya R (2020) Application management in fog computing environments: A taxonomy, review and future directions. ACM Comput Surv 53(4):1–43
15. Helbig M, Deb K, Engelbrecht A (2016) Key challenges and future directions of dynamic multi-objective optimisation. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE, Vancouver. pp 1256–1261
16. Zhou D, Chao F, Lin C-M, Yang L, Shi M, Zhou C (2017) Integration of fuzzy CMAC and BELC networks for uncertain nonlinear system control. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, Naples. pp 1–6
17. Abdullah L (2013) Fuzzy multi criteria decision making and its applications: a brief review of category. Procedia-Soc Behav Sci 97:131–136
18. Wei X, Tang C, Fan J, Subramaniam S (2019) Joint optimization of energy consumption and delay in cloud-to-thing continuum. IEEE Internet Things J 6(2):2325–2337
19. Yang L, Cao J, Cheng H, Ji Y (2014) Multi-user computation partitioning for latency sensitive mobile cloud applications. IEEE Trans Comput 64(8):2253–2266
20. Mahmud R, Ramamohanarao K, Buyya R (2018) Latency-aware application module management for fog computing environments. ACM Trans Internet Technol (TOIT) 19(1):1–21
21. Mukherjee A, De D, Roy DG (2016) A power and latency aware cloudlet selection strategy for multi-cloudlet environment. IEEE Trans Cloud Comput 7(1):141–154
22. Sharma S, Saini H (2019) A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. Sustain Comput Informa Syst 24:100355
23. Xu X, Fu S, Cai Q, Tian W, Liu W, Dou W, Sun X, Liu AX (2018) Dynamic resource allocation for load balancing in fog environment. Wirel Commun Mob Comput 2018:1–15
24. Yang B, Chai WK, Pavlou G, Katsaros KV (2016) Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud. In: 2016 5th IEEE International Conference on Cloud Networking (Cloudnet). IEEE, Pisa. pp 136–141
25. Mahmud R, Srirama SN, Ramamohanarao K, Buyya R (2019) Quality of Experience (QoE)-aware placement of applications in Fog computing environments. J Parallel Distrib Comput 132:190–203
26. Hájek P (2013) Metamathematics of Fuzzy Logic, Vol. 4. Springer, Springer Netherlands
27. Kong X, Lin C, Jiang Y, Yan W, Chu X (2011) Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. J Netw Comput Appl 34(4):1068–1077
28. Ansari A, Bakar AA (2014) A comparative study of three artificial intelligence techniques: Genetic algorithm, neural network, and fuzzy logic, on scheduling problem. In: 2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology. IEEE, Kota Kinabalu. pp 31–36
29. Jiang C, Cheng X, Gao H, Zhou X, Wan J (2019) Toward computation offloading in edge computing: A survey. IEEE Access 7:131543–131558
30. Mach P, Becvar Z (2017) Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun Surv Tutor 19(3):1628–1656
31. Lyu X, Tian H, Jiang L, Vinel A, Maharjan S, Gjessing S, Zhang Y (2018) Selective offloading in mobile edge computing for the green internet of things. IEEE Network 32(1):54–60
32. Dinh TQ, Tang J, La QD, Quek TQ (2017) Offloading in mobile edge computing: Task allocation and computational frequency scaling. IEEE Trans Commun 65(8):3571–3584
33. Flores H, Su X, Kostakos V, Ding AY, Nurmi P, Tarkoma S, Hui P, Li Y (2017) Large-scale offloading in the internet of things. In: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). IEEE, Kona. pp 479–484
34. Samie F, Tsoutsouras V, Bauer L, Xydis S, Soudris D, Henkel J (2016) Computation offloading and resource allocation for low-power iot edge devices. In: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT). IEEE, Reston. pp 7–12
35. Wang S, Zafer M, Leung KK (2017) Online placement of multi-component applications in edge computing environments. IEEE Access 5:2514–2533
36. Rodrigues TG, Suto K, Nishiyama H, Kato N (2016) Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control. IEEE Trans Comput 66(5):810–819
37. Deng R, Lu R, Lai C, Luan TH, Liang H (2016) Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J 3(6):1171–1181
38. Zeng D, Gu L, Guo S, Cheng Z, Yu S (2016) Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Trans Comput 65(12):3702–3712
39. Fan Q, Ansari N (2018) Application aware workload allocation for edge computing-based IoT. IEEE Internet Things J 5(3):2146–2153
40. Hassan HO, Azizi S, Shojafar M (2020) Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments. IET Communications 14(13):2117–2129
41. Sonmez C, Ozgovde A, Ersoy C (2019) Fuzzy workload orchestration for edge computing. IEEE Trans Netw Serv Manag 16(2):769–782

42. Nan Y, Li W, Bao W, Delicato FC, Pires PF, Zomaya AY (2016) Cost-effective processing for delay-sensitive applications in cloud of things systems. In: 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA). IEEE, Cambridge. pp 162–169
43. Xu J, Palanisamy B, Ludwig H, Wang Q (2017) Zenith: Utility-aware resource allocation for edge computing. In: 2017 IEEE International Conference on Edge Computing (EDGE). IEEE, Honolulu. pp 47–54
44. Li Y, Wang S (2018) An energy-aware edge server placement algorithm in mobile edge computing. In: 2018 IEEE International Conference on Edge Computing (EDGE). IEEE, San Francisco. pp 66–73
45. Scoca V, Aral A, Brandic I, De Nicola R, Uriarte RB (2018) Scheduling latency-sensitive applications in edge computing. In: Closer. pp 158–168
46. Roy DG, De D, Mukherjee A, Buyya R (2017) Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. J Supercomput 73(4):1672–1690
47. Taneja M, Davy A (2017) Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, Lisbon. pp 1222–1228
48. Taleb T, Samdanis K, Mada B, Flinck H, Dutta S, Sabella D (2017) On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Commun Surv Tutor 19(3):1657–1681
49. Choi N, Kim D, Lee S-J, Yi Y (2017) A fog operating system for user-oriented iot services: Challenges and research directions. IEEE Commun Mag 55(8):44–51
50. Bonomi F, Milito R, Zhu J, 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. pp 13–16
51. Li X, Li D, Wan J, Liu C, Imran M (2018) Adaptive transmission optimization in SDN-based industrial Internet of Things with edge computing. IEEE Internet Things J 5(3):1351–1360
52. Santoro D, Zozin D, Pizzolli D, De Pellegrini F, Cretti S (2017) Foggy: a platform for workload orchestration in a fog computing environment. In: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, Hong Kong. pp 231–234
53. Hegyi A, Flinck H, Ketyko I, Kuure P, Nemes C, Pinter L (2016) Application orchestration in mobile edge cloud: placing of iot applications to the edge. In: 2016 IEEE 1st International Workshops on Foundations and Applications of Self Systems. IEEE, Augsburg. pp 230–235
54. Imagane K, Kanai K, Katto J, Tsuda T, Nakazato H (2018) Performance evaluations of multimedia service function chaining in edge clouds. In: 2018 15th IEEE Annual Consumer Communications and Networking Conference (CCNC). IEEE, Las Vegas. pp 1–4
55. Carrega A, Repetto M, Gouvas P, Zafeiropoulos A (2017) A middleware for mobile edge computing. IEEE Cloud Comput 4(4):26–37
56. Taleb T, Dutta S, Ksentini A, Iqbal M, Flinck H (2017) Mobile edge computing potential in making cities smarter. IEEE Commun Mag 55(3):38–43
57. Basic F, Aral A, Brandic I (2019) Fuzzy handoff control in edge offloading. In: 2019 IEEE International Conference on Fog Computing (ICFC). IEEE, Prague. pp 87–96
58. Sonmez C, Ozgovde A, Ersoy C (2018) Edgecloudsim: An environment for performance evaluation of edge computing systems. Trans Emerg Telecommun Technolog 29(11):3493
59. Zhang Q, Lin M, Yang LT, Chen Z, Khan SU, Li P (2018) A double deep Q-learning model for energy-efficient edge scheduling. IEEE Trans Serv Comput 12(5):739–749
60. Kovalenko A, Hussain RF, Semiari O, Salehi MA (2019) Robust resource allocation using edge computing for vehicle to infrastructure (v2i) networks. In: 2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC). IEEE, Larnaca. pp 1–6
61. Ramaswamy L, Iyengar A, Chen J (2006) Cooperative data placement and replication in edge cache networks. In: 2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing. IEEE, Atlanta. pp 1–9
62. Mao L, Li Y, Peng G, Xu X, Lin W (2018) A multi-resource task scheduling algorithm for energy-performance trade-offs in green clouds. Sustain Comput Informa Syst 19:233–241
63. Flores H, Srirama S (2013) Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning. In: Proceeding of the Fourth ACM Workshop on Mobile Cloud Computing and Services. pp 9–16
64. Aldossary M, Djemame K (2018) Performance and energy-based cost prediction of virtual machines auto-scaling in clouds. In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, Prague. pp 502–509
65. Sonmez C, Ozgovde A, Ersoy C (2017) Performance evaluation of single-tier and two-tier cloudlet assisted applications. In: 2017 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, Paris. pp 302–307
66. Abreu DP, Velasquez K, Curado M, Monteiro E (2020) A comparative analysis of simulators for the cloud to fog continuum. Simul Model Pract Theory 101:102029

Publisher's Note