Computer Systems Science & Engineering
DOI:10.32604/csse.2022.018112
Article

Tech Science Press

Exploring and Modelling IoT Offloading Policies in Edge Cloud Environments

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

[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
*Corresponding Author: Mohammad Aldossary. Email: mm.aldossary@psau.edu.sa
Received: 25 February 2021; Accepted: 19 April 2021

Abstract: The Internet of Things (IoT) has recently become a popular technology that can play increasingly important roles in every aspect of our daily life. For collaboration between IoT devices and edge cloud servers, edge server nodes provide the computation and storage capabilities for IoT devices through the task offloading process for accelerating tasks with large resource requests. However, the quantitative impact of different offloading architectures and policies on IoT applications' performance remains far from clear, especially with a dynamic and unpredictable range of connected physical and virtual devices. To this end, this work models the performance impact by exploiting a potential latency that exhibits within the environment of edge cloud. Also, it investigates and compares the effects of loosely-coupled (LC) and orchestrator-enabled (OE) architecture. The LC scheme can smoothly address task redistribution with less time consumption for the offloading sceneries with small scale and small task requests. Moreover, the OE scheme not only outperforms the LC scheme in the large-scale tasks requests and offloading occurs but also reduces the overall time by 28.19%. Finally, to achieve optimized solutions for optimal offloading placement with different constraints, orchestration is important.

Keywords: Internet of things; application deployment; latency-sensitive; edge orchestrator

1 Introduction

In this digitized era, the number of sensor-enabled objects and devices connected to the network has significantly increased, where this number was doubled five years ago (i.e., between 2014 and 2019) [1]. This revolution has led to a new era of technology called the Internet of Things (IoT), that has gained well consideration from both industry and academia. Generally, IoT technology is well-defined as "the infrastructure of a global network with self-configuring and dynamic capabilities on the basis offset of interoperable communication and standards protocols. Additionally, the identities and attributes of physical and virtual IoT *things* are capable to use intelligent interfaces and can be integrated as a network of information" [2]. Further, through the IoT technology, physical objects, such as vehicles, buildings, and sensors, are interconnected and created a virtual environment, which leads to increase the integration of cyber-physical objects.

Furthermore, in nature, these “things” are considered as mobile, which demand data from other sources, however their computational resources are limited. Therefore, cloud and edge computing support these devices with computational and storage capabilities to address the challenges related to energy and performance and to guarantees IoT service provisions, primarily through task offloading process. Specifically, in the task offloading process, the computations are transferred from resource-limited devices (i.e., IoT devices) to resource-rich nodes (cloud servers) for improving the performance of mobile applications and total usage of power. Consequently, the task offloading concept is utilized in various domains and industries including transportation, e-health care, smart homes, and factories [3]. Moreover, it is particularly useful for streaming-processing applications, online gaming, virtual reality, and video conferencing [4], that are latency-sensitive services with high-quality demand. In such scenarios, the offloading process and data transmission between IoT devices and cloud nodes are going together that may be different among applications. For example, some applications demand low communication and high computation resources, or vice versa. Furthermore, due to the mobility feature of these devices, their number exponentially increases in some areas, and thereby exacerbates network issues [5].

Usually, the workloads of IoT include streaming of data and controlling the flows across different regions are required to be processed and analyzed in real-time. Therefore, the most effective way to meet these requirements is using small-scale access points at the network edge to complement cloud services. Such access points, such as computers or clusters of mobile users, can provide resource-rich communication intermediates with a small-scale. For instance, the architecture of multi-layered, including cloudlets [6], CloudAP [7], and other systems [8,9], not only aims to underpin delay-sensitive applications but also minimizes the overall service time. However, because of the lack of effective orchestration and integration among cloud and cloudlets, the quality of service cannot be entirely guaranteed. To this end, the intelligent task offloading across edges and cloud nodes can be improved through adopting various algorithms of orchestration [10,11]. In this regard, the quantitative impact of different offloading architecture and policies on IoT applications and services’ end-to-end performance, primarily given a dynamic pattern of resource and task characteristics manifests.

Motivated by such consideration, in this study, we investigate the effectiveness of various architectures of edge cloud offloading on the total IoT service time throughout the process of task offloading and study how the demands of various application parameters, such as communication and computation, can influence on the holistic efficiency. Specifically, we investigate two kinds of basic three-tier offloading architectures namely loosely-coupled (LC) and orchestrator-enabled (OE). In addition, the impact of LC and OE schemes on the execution of IoT service are compared through performance-driven modeling, in which computation resources’ allocation and the latency of communication that derivatized from various connections of network among tiers are jointly considered. Experimental results showed that the computation requirement has more impact on IoT applications’ performance than the requirement of communication. However, with scaling IoT devices’ numbers up, the bandwidth of communication will be the leading resource and become the main factor that can directly impact the total performance.

Furthermore, for small-scale and small tasks offloading scenarios, the LC scheme can smoothly address task redistribution with shortened time consumption for the offloading scenarios with small scale and small tasks requests, but when the resource requests of IoT tasks become bigger or when offloading is frequent, the OE scheme outperforms LC and can reduce overall time by 28.19%.

The main contributions reported in this paper are summarized as follows:

- A performance-driven scheme is proposed to evaluate the effectiveness of IoT services considering computational and communication resources.
- A performance analysis is then conducted to study the behavior of the system under LC and OE offloading schemes.

- Several meaningful findings are concluded from our simulation-based evaluation, which can be used in a joint cloud environment to improve the task offloading efficiency and to achieve well-balanced management of resources.

The remainder of this paper is organized as follows: related work is summarized in Section 2. Section 3 presents the research problem and challenges. Section 4 analyzes mainstream architectural schemes for IoT task offloading. Section 5 describes the modeling and measurement of performance. Simulation experiments are conducted in Section 6, followed by results and discussion. Finally, the conclusion along with a future work discussion are presented in Section 7.

2 Related Work

In recent years, research has received considerable attention to deal with the problem of service time in the environment of edge cloud computing, in which addressing the computation and\or the communication delay is the main concern. In this section, a brief overview of the main studies addressing the service time with other objectives including cost and energy will be introduced.

In [12], a new framework is suggested for offloading the computation tasks from mobile devices to multiple edge nodes with the objective of minimizing the computational latency and power consumption. Meanwhile, Du et al. [13] designed a new algorithm for ensuring a reasonable delay of computation in a fog-cloud system. Furthermore, Liu et al. [14] developed a new algorithm for task scheduling with the aim of reducing the overall latency, in which queuing state and execution are jointly considered. Whereas, in order to minimize the latency of end-to-end devices, Rodrigues et al. [15] introduced a new hybrid method focusing on the migration and transmission delay of virtual machines through enhancing the computational process and communication resources. Wei et al. [16] developed an algorithm with the objective of minimizing the latency, in which the transmission power and processing frequencies are jointly considered. Yang et al. [17] presented a delay-sensitive applications' approach that can maintain the maximum allowable time and reduce the operational cost. Zeng et al. [18] designed a novel algorithm for minimizing the tasks' completion time where the load balancing and task images' allocation are focused.

From the application side, a limited number of papers have addressed the service time minimization with different application's types which are maintained by the edge cloud system. Since the variation of the computational and communication requirements of IoT applications [19] and their dynamic demands [20], a new approach is proposed in [21] to address the communication and computational delays, in which the offloaded tasks are allocated to the suitable resources at the edge cloud system concerning the application type. Roy et al. [22] suggested a strategy based on application specifications to select the target edge node with the object of minimizing the delay and power consumption. Although IoT applications with latency-sensitive were studied, the effects of various deployments of edge cloud in terms of service time minimization have not adequately been addressed.

It is observed from the literature studies that while the service time delay with different applications' requirements has been considered in some research, there is a lack of scientific understanding of the effectiveness of different architectures of edge on the system performance, especially total service time. Consequently, numerous edge computing systems-based architectures were described in [5,9,23] in which pushing computational resources nearest to end-users is the main goal. However, there are key differences between edge nodes and an edge network, including deployment, network technology, and the size and location of an edge and the communication way with the central cloud, which lead to overall latency minimization when they are efficiently considered through the application owner and the service provider.

3 Research Problem and Challenges

3.1 Problem Statement

The advances in mobile devices lead to be more adapted to function IoT services. However, mobile devices' limited resources (i.e., computation and storage) and battery capabilities further restrict the execution of resource-demanding applications (e.g., Artificial Intelligence (AI), assisted multimedia applications, and Augmented Reality (AR)), in which low latency and the throughput of broad bandwidth are urgently required. To alleviate these limitations and meet the applications' requirements, various architectures based on cloud and edge resources have been proposed, where these architectures can perform the coordination roles between these devices and edge and cloud resources.

The key requirement for coordination is generally agreed that these resources should be added at the network edge (i.e., move the computation, storage and bandwidth resources more closed to these devices) to reduce the traffic of the network and minimize the latency response. In contrast, resource-demanding tasks are offloaded and parallelly processed at a centralized cloud for acceleration.

Noting that, the task offloading's effectiveness and efficiency can be affected by many factors, either directly or implicitly. Consequently, quantified modeling and analysis of their performance impacts as well as comparisons between different policies of offloading are necessary required. Moreover, the computation and communication requirements, as well as the existing resource supply of IoT applications, are variant by nature.

Motivated by such consideration, in this paper, we evaluate the behavior of different workloads of IoT devices through adopting different policies of task offloading. Moreover, the result of this evaluation can be utilized to enhance the service quality, where service time will be reduced.

3.2 Emerging Challenges

Several obstacles appear due to the proliferation of IoT applications and their variation such as:

- **Scale and complexity:** As the development of heterogeneous sensors and smart devices is increasing, it becomes difficult to select optimal resources for IoT tasks in a collaborative cloud environment with a wide variety of personalized needs and customized hardware configurations. For example, some tasks require specific hardware architectures (e.g., ARM, Intel) or operating systems for execution, where others (e.g., security-based tasks) need specific hardware and working protocols. Thereby, orchestration is considered the best solution which not only meets these functional requirements, but also can respond to the growth in workflows that are dynamically changing. The orchestrator must determine whether the complex services can be provided correctly and efficiently through the assembled systems which consist of end-user devices, edge nodes, and cloud resources tied with geographic distributions and constraints. Particularly, the scalability bottlenecks caused by the increasing application scale must be predicated, detected and resolved automatically using an orchestrator.
- **Dynamicity:** One of IoT applications' key features is that the topology or diversity of resources may change dynamically. With software upgrades or frequent join-leave network object behavior, this can be a particular problem. The cloud's internal properties and output may be modified, thereby lead to a change in the overall workload pattern. Specifically, fluctuations in connectivity, bandwidth, and device mobility fluctuation may affect the communication links. There can also be unpredictable requirements of resource allocation and task offloading across the combination of cloud and edge nodes.
- **Scalability:** Scalability is considered as another point of a challenge, in which the management of the explosive IoT applications' number and the dynamic changes in the resources are difficult in the

present context. In addition, the attributes of IoT tasks can be changed dynamically, thereby the execution time may be different for each task's procedure. Besides, the mobility of IoT devices leads to over-crowd them in some areas, which then can increase the workload for the connected edge node and thereby the service performance will be degraded.

To summarize, the central stratagem of a resource manager is responsible for allocating the physical resources to IoT tasks, in the system of edge cloud. Offloading the IoT tasks to the best destination regarding the state of the runtime system can improve the performance of applications. This study examines the efficiency and effectiveness of task offloading through various systems of edge cloud systems with different environmental parameters, where the above-mentioned challenges are considered.

4 Analyzing Mainstream Architectural Schemes for IoT Task Offloading

In this section, we present the offloading schemes and their relevant factors of performance, including communication and computing delays, that we chose to tackle.

Generally, for supporting IoT applications, edge cloud systems are composed of three different tiers, in which the edge tier is placed in the middle and consists of a set of edge server nodes that are geographically distributed and connected to the upper tier (i.e., cloud data center) through the core network. Besides, IoT devices are interconnected to the nodes of the edge directly *via* a wireless channel. It should be noted that the bandwidth of the network and efficiency of the communication among edge and cloud nodes are considered the main aspects that determine IoT devices' performance.

This study aims to evaluate the task offloading influence within an edge cloud system on IoT service performance by measuring the end-to-end service time for each task in the LC and OE architectures.

We summarize mainstream edge-cloud-based IoT supporting systems as belonging to two categories, as illustrated in Fig. 1.

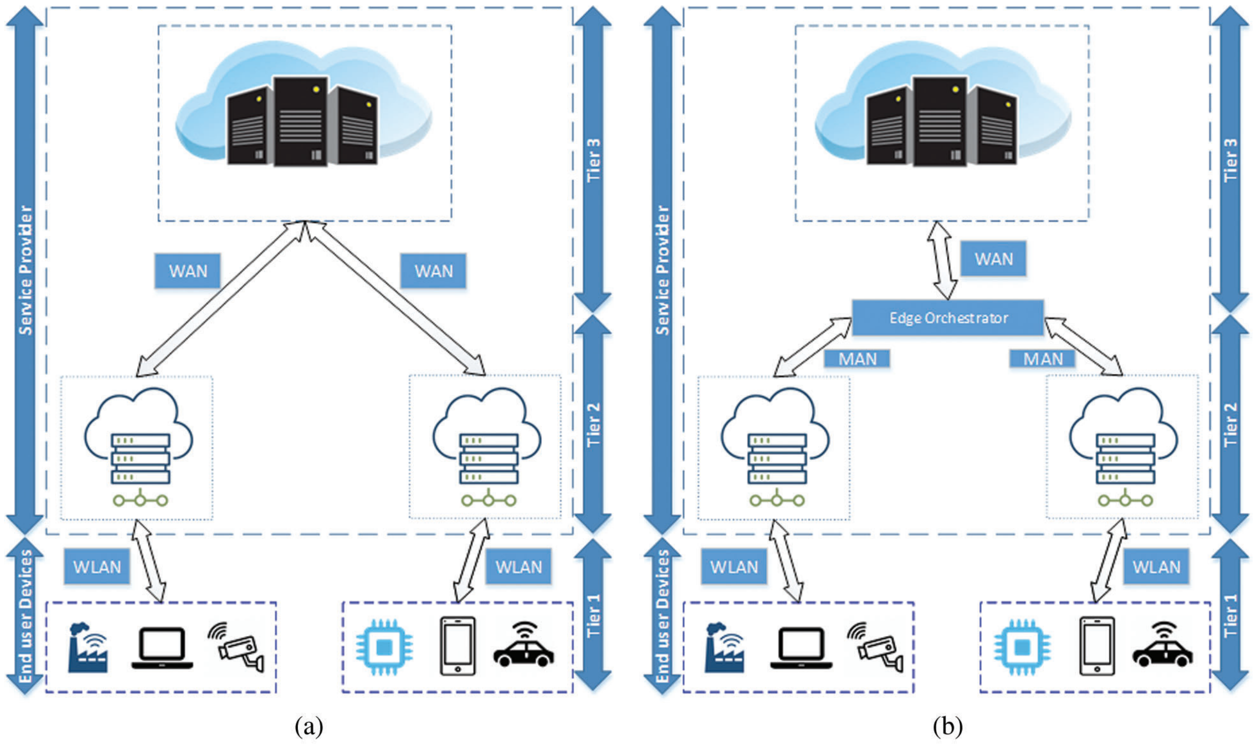


Figure 1: Two paradigmatic schemes for cross-cloud IoT services: (a) Loosely-coupled three-tier scheme (LC); (b) Orchestrator-enabled three-tier scheme (OE)

4.1 Loosely-Coupled Three-Tier Scheme (LC)

In the LC scheme, the applications of IoT are deployed over the nodes of edge and cloud datacenter which are connected. This means that task offloading can only be executed in the connected edge or the central cloud. Various studies have adopted this scheme in theirs work (e.g., [24–26]). In reality, mobile devices' workload is only allowed to be offloaded to one destination for sequential execution or to multiple servers for parallel execution. The main shortcoming of this scheme is that it does not allow collaboration between different edge infrastructures. This mechanism leads to static partitioning and delivery of tasks. Once offloaded onto a node, the tasks cannot be adjusted at runtime.

4.2 Orchestrator-Enabled Three-Tier Scheme (OE)

In the OE scheme, the applications of IoT are deployed across set of different edge nodes that are managed and controlled by an edge orchestrator and the central cloud, where this orchestrator can bind each IoT task to the appropriate resource of edge nodes. A task placement algorithm plays a significant role in selecting the offloading destination. Regarding the mathematical formulation and optimization models and guided by the intuition in [27], the offloading algorithms are classified into five different categories namely 0–1 integer linear programming, K-dimensional bin parking convex optimization, Markov decision process, and Lyapunov optimization problems, where most of them can be applied to solve an NP-hard problem type. Therefore, the solution accuracy and time complexity trade-off are considered as a main point that needs to be struck with the consideration of different scale types of workload. This scheme has been utilized in several works for supporting their applications of IoT, such as [28–30].

4.3 Comparison (LC vs. OE)

In effect, LC accomplishes the offloading process just by linking IoT devices with a close edge node, where the task is only allowed to be offloaded to the connected node. In addition, in the case of no available node for holding the tasks, the LC system will wait till the edge node release more resources in order to cover the pending tasks. Otherwise, the offloaded tasks will be directed to the cloud. The task offload is unidirectional and cannot be balanced collaboratively between various infrastructures of edge.

On the other hand, in the task-offloading procedure in OE, tasks will be received by an edge orchestrator through the connected host edge, which then the offloading algorithm is responsible for allocating them to the suitable set of different constraints, including the availability of resources and expected delays. Therefore, an edge orchestrator can easily address the dynamic offloading at runtime and partition the tasks and parallelly offloading them to a set of destinations including several edge cloud nodes. Also, orchestrators can coordinate the infrastructures of individual edge.

5 Modelling and Measurement of Performance

5.1 End-to-End Service Time

In this section, the total service time of the end-to-end device is split into separate measurable segments. In general, as shown in Fig. 1, the edge-cloud system is composed of a set of IoT devices, a set of edge nodes, and a single cloud server node. In addition, there are two types of assignable resources which are computation (e.g., GPU and CPU, etc.) and the bandwidth of the communication network.

More specifically, there are a set of IoT devices that are connected with edge nodes *via* the Wide Area Network (WAN). A Wi-Fi access point typically covers each spot, and these devices are interconnected to associated WLAN (Wireless Local Area Network), which can continue to send a set of requests to edge server nodes when they have entered the relevant zone. Afterward, the WAN or Metropolitan Area Network (MAN), provided through the Wi-Fi access point, will eventually be utilized, once the request is picked up and moved to nodes of edge or cloud. Noting that, the transferred (i.e., upload and download) and being processed (i.e., input and output) data may have different sizes with unpredictable lengths. For example, workloads will probably vary, depending greatly on the functional requirements of the IoT tasks' demands, such as CPU core numbers, power supplier, communication bandwidth amount, and non-functional demands including access control and security.

5.2 Measurement

In this study, the service time for each task is assumed to be roughly computed based on the summation of communication and computation time, where the computation time is expressed by the queuing time Q and the actual processing time P for each task. Then, the queuing time is composed of queuing in edge node, edge node in-between, and cloud node: $Q \leftarrow (Q_{edge}, Q_{xedge}, Q_{cloud})$, and the actual processing time is expressed by $P \leftarrow (P_{edge}, P_{xedge}, P_{cloud})$.

Furthermore, the communication time is calculated based on the summation of transmission and delay of propagation for data (i.e., upload and download). More specifically, the delay of transmission can be represented by the needed time for pushing the data through the link, whereas the propagation delay is represented by the required time for transferring the data between the sender and receiver. Moreover, from the point of view of uploading and downloading, the consumption of communication time is defined as a composition of uploading time $U \leftarrow (U_{WLAN}, U_{MAN}, U_{WAN})$ and downloading time $D \leftarrow (D_{WLAN}, D_{MAN}, D_{WAN})$ to describe the total delay produced between edge cloud architecture's three tiers.

Although the edge nodes have a limitation of computation resources, they can provide minimal networking delays *via* network connections. Whereas, the pool of cloud resources can be utilized to address and process big data, thereby leading to minimize the processing time. Moreover, the communication time required for offloading and then processing the task depends upon the location of the server node, wherever collaborative edge nodes, edge node, or cloud node, in which the cloud node takes a longer time than the closet edge nodes due to the existing of WLAN network protocol.

Therefore, in the edge cloud system, the location for processing each task (i.e., cloud, another nearby edge, or connected edge) needs to be determined to model the time of computation and communication. Noting that, each edge node provides small computational resources with the same functionality, which is proximity to end devices and differs from the cloud resources in their capacity. Thus, the task offloading's service time latency in the edge cloud system can be computed as follows:

- **Latency to local edge:** the overhead of service time for executing the computation tasks at the connected edge node within a WLAN can be computed based on the summation of uploading, queuing, processing and downloading time which is expressed as:

$$L_{edge} = \sum U_{WLAN} + Q_{edge} + P_{edge} + D_{WLAN} \tag{1}$$

- **Latency to collaborative edge:** the overhead of service time for executing the computation tasks at the collaborative edge nodes can be computed based on the summation of uploading time between end device and local edge through WLAN and between end device and the collaborative edge nodes through MAN. Besides, the queuing, processing and downloading time through MAN and WLAN, which is expressed as:

$$L_{xedge} = \sum U_{WLAN} + U_{MAN} + Q_{xedge} + P_{xedge} + D_{MAN} + D_{WLAN} \tag{2}$$

- **Latency to central cloud:** to calculate the total overhead of service time for executing the computation task at the cloud server node, the delay of the network *via* WLAN, MAN, WAN and the queuing and processing time should be considered as follows:

$$L_{cloud} = \sum U_{WLAN} + U_{MAN} + U_{WAN} + Q_{cloud} + P_{cloud} + D_{WAN} + D_{MAN} + D_{WLAN} \tag{3}$$

6 Evaluation

6.1 Experimental Setup

Empirically testing different edge computing architectures is not a simple procedure because of the variety of frameworks and applications and the different devices, computing services, and communication protocols therein. EdgeCloudSim [31] is an environment that simulates the desirable architectures by adjusting the CloudSim [32]. EdgeCloudSim can provide extra models for representing some sub-processes and can be used to account for characteristics of the edge computing node and the IoT devices' services. An extra model of queuing was presented to denote the delay in the WLAN, MAN, and WAN and a mobility model and a CPU utilization model for Virtual Machines (VMs). This motivates us to use this simulation in order to investigate and evaluate the service time for a two-tier architecture with or without edge orchestrators for several IoT workloads.

Using the simulator tool, we did several experiments to examine the two architecture models. The key experiment parameters are shown in Tab. 1. Moreover, for each architecture model, there are three nodes of edge which are distributed and connected to a centralized cloud. In addition, every edge node serves a set of IoT devices in its zone. Further, to examine and investigate the service time performance for each architecture, the total devices' number is increased from 100 to 1000 in increments of 100. Finally, each IoT device can generate a set of tasks that can be used to represent the workload of IoT. Therefore, increasing the number of IoT devices will result in an increase in the number of tasks.

Table 1: Simulation key parameters

Parameter	Value
Simulation time (h)	2
Warm up period (s)	3
Repetitions number	5
Edge nodes number	3
Hosts' number per edge nodes	2
VMs' number per edge server/cloud	4/not limited
VM speed (MIPS) per edge server/cloud	10000
End devices' minimum number	100
End devices' maximum number	1000
Active/idle for end devices period (s)	45/15

6.2 Methodology

There is a variation in IoT workload, where the applications' demand is ranging from low communication and computation (e.g., healthcare) to high communication and computation (e.g., online gaming). The numerical setup is considered a significant step. In the following steps, the uncertainty associated with the unpredictable workload presented in [5] inspires and motivates us.

To deal with various applications that might be used, the bandwidth of communication requirement is increased from 0.25 to 1 MB in increments of 0.25 MB, while the computation requirement is doubled between 500 MIPS and 4000 MIPS. This configuration results in 16 different combinations, labeled App1 to App16, as shown in Tab. 2. We tried to determine how those resource requirements proposed by IoT tasks affect the total service time (See Section 6.3.1).

Table 2: Examples of workloads and their related CPU speed and bandwidth configurations

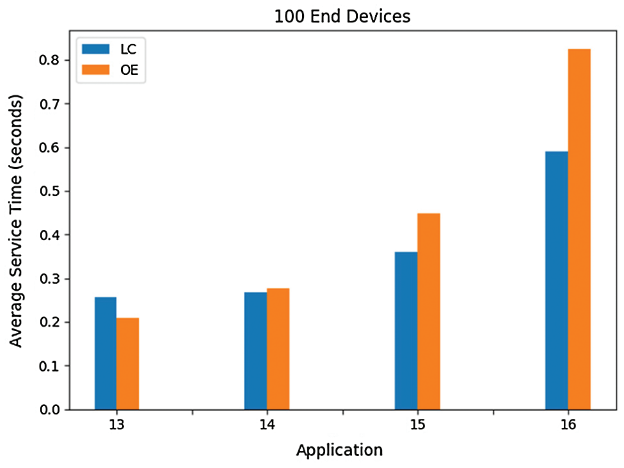
	0.25 MB	0.5 MB	0.75 MB	1 MB
500 MIPS	App1	App5	App9	App13
1000 MIPS	App2	App6	App10	App14
2000 MIPS	App3	App7	App11	App15
4000 MIPS	App4	App8	App12	App16

Furthermore, to validate different schemes' scalability and the sensitivity of performance to the edge cloud environment, we tuned the number of end devices from 100 to 900 and examined the corresponding performance (See Section 6.3.2). We also put them together and then presented the results aggregated for the submitted applications to show the integral impacts of various performance parameters (see Section 6.3.3).

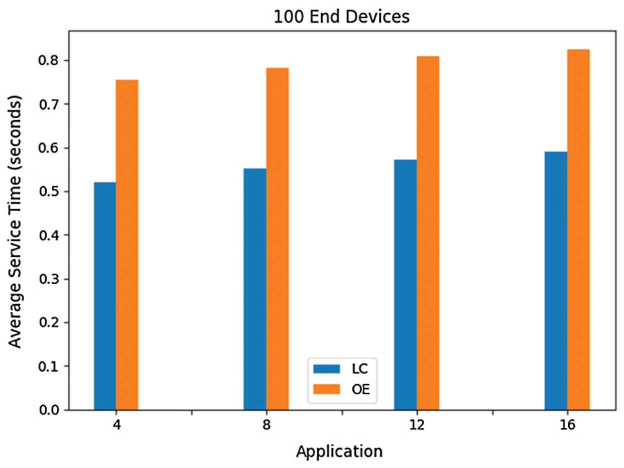
6.3 Results and Discussion

6.3.1 Impact of IoT Task Resource Requirements

As depicted in Fig. 2a, the service time will sharply increase when the CPU requirement per task grows. The reason for this is that computation resources are relatively scarce. Once a large number of tasks ask and compete for CPU clock cycles, the orchestrator usually takes a long time to coordinate among different resource entities, resulting in an increased holistic time consumption. LC and OE's difference will grow enormously when all CPU cores are pinned and shared by co-located tasks.



(a)



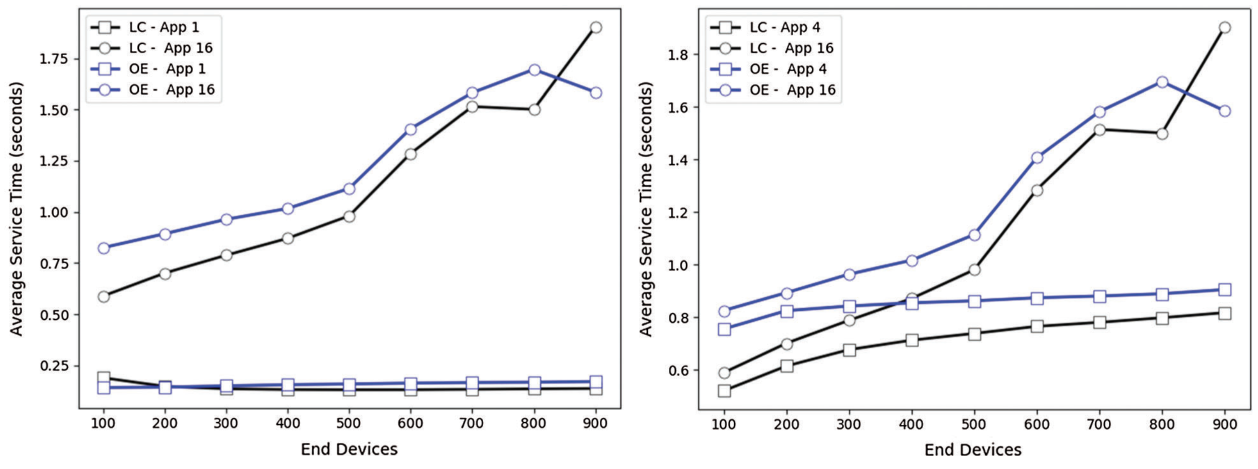
(b)

Figure 2: The impact of computational *vs*. communication demands (a) Impact of varying CPU speed (b) Impact of varying bandwidth

By contrast, as seen in Fig. 2b, if the bandwidth requirement of the task varies, the service time is only slightly increased because bandwidth resources are easily found. Orchestrators can very quickly find suitable bandwidth and allocate the isolated bandwidth by using heuristic or approximate algorithms.

6.3.2 Impact of End Device Number

In this section, we verify how the service time changes regarding the variation in IoT device number. As observed from Fig. 3a, with a small number of resource requests using IoT tasks (e.g., App1), the total service time is insensitive to the end devices' number increasing. In reality, even if the offloading procedures significantly increase, the available resources from edge nodes and cloud nodes are more than enough to accept such offloaded tasks. The orchestrator can rapidly select a destination for the incoming tasks. In comparison, LC and OE experience an enormous time when the end devices' number is increasing, especially for dealing with App16 (i.e., largest resource requests' tasks). Moreover, under the OE scheme, the orchestrator spends more time coordinating resources between edge and cloud nodes and subsequently make the decisions of placement. Further, the same phenomenon can be observed for App4 in comparison with App16, seen in Fig. 3b.



(a) (b)

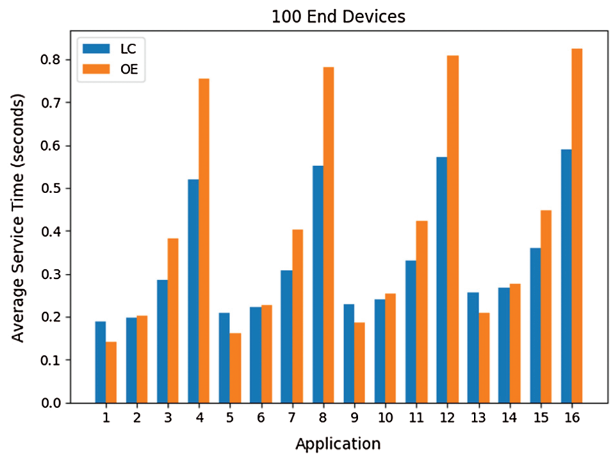
Figure 3: The impact of application number under different offloading schemes (a) App1 *vs*. App16 (b) App4 *vs*. App16

In this analysis, the sensitivity to the device scale changes across different applications. Service providers of IoT may require coordinating and then specify the appropriate resource that an IoT task can leverage regarding the edge cloud infrastructure's changing configuration.

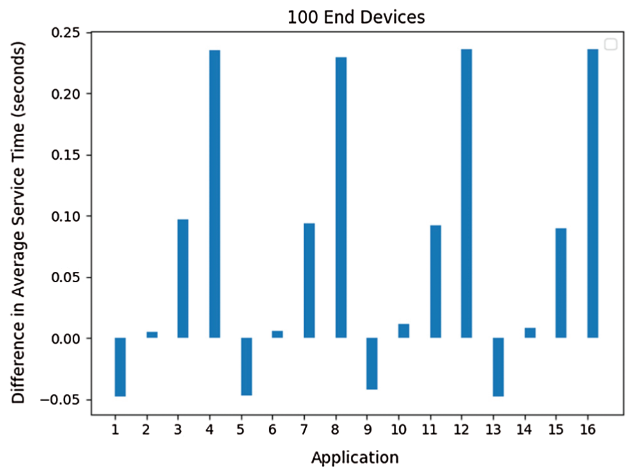
Another important finding is that an LC architecture may be much more suitable with a small number of IoT devices. However, with increasing resource request's number (e.g., App-16 or more), the LC scheme increases the consumption of time, probably because plain offloading to a single node cannot satisfy the offloading requirements. This problem can only be solved by orchestration with complex constraints.

6.3.3 Putting it Together

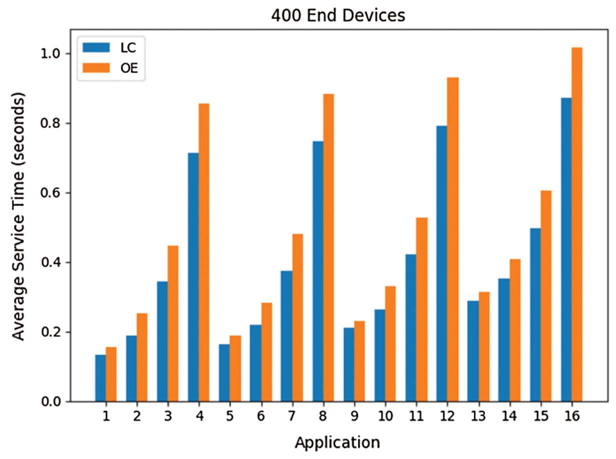
We further examine the scalability of the system to investigate how LC and OE schemes impact service time performance. Fig. 4 shows the detailed results with device numbers selected from 100, 400, and 900, and the corresponding performance gaps between LC and OE. All kinds of applications mentioned in Tab. 2 are evaluated.



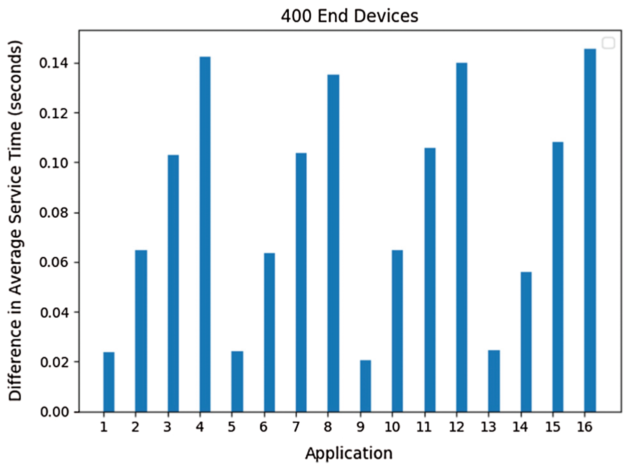
(a)



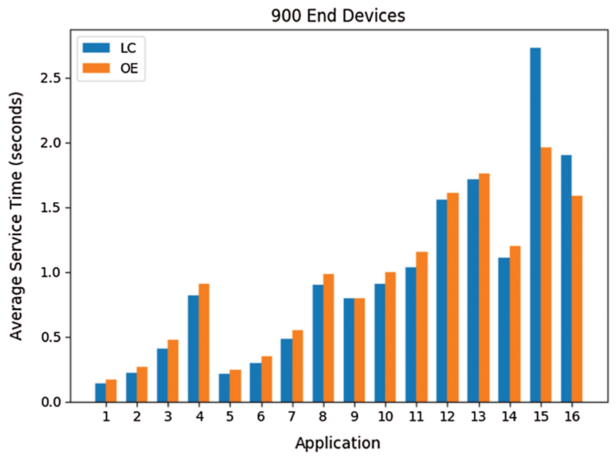
(b)



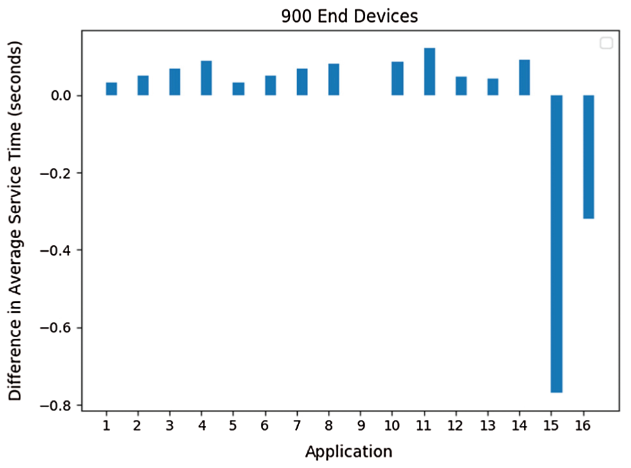
(c)



(d)



(e)



(f)

Figure 4: (Left) The overall service time comparison of all applications under LC and OE; (Right) The difference between LC and OE (a) 100 devices (b) 100 devices-diff between LC and OE (c) 400 devices (d) 400 devices-diff between LC and OE (e) 900 devices (f) 900 devices-diff between LC and OE

Overall, there is an approximate upward trend in both CPU speed and bandwidth with overall service time. Figs. 4a, 4c, and 4e all demonstrate that within each comparison group Application 1–4, 5–8, 9–12, and 13–16, the service time increases linearly in both LC and OE. However, the discrepancy between LC and OE incurred by the increased amount of CPU requests becomes higher due to the increased difficulties in finding CPU resources from the edge cloud environment. The overall time consumption to satisfy the changing bandwidth requirement remains stable.

Additionally, it is observable that OE outperforms LC only in certain cases. For instance, when the device number is fixed at 100 and CPU speed is limited to 500 MIPS, the service times under different bandwidths are stable and similar. In the OE scheme, the time can be reduced by an average of 24.1% compared to the LC scheme. Another extreme case occurs when the CPU speed is increased to 2k and 4k MIPS if the number of end devices is 900 and bandwidth allocation is 1 MB, in which the time in OE can be reduced by roughly 28.19%. Indeed, the LC system can simply find a single node to perform the task offloading, and then this task cannot be transferred to another node once it is offloaded to a specific one. Also, in the case of no available node for holding tasks, the LC scheme will wait till the edge node release more resources in order to cover the awaiting tasks. This operation spends more time with respect to the OE scheme, that can address the dynamic offloading of task easily at runtime and partition the tasks and parallelly offloading them to a set of destinations.

To sum up, the computation requirement spends an additional effect on the performance of IoT applications in comparison with the communication requirement. However, in the case of IoT devices scaled up, the bandwidth of communication will be the key factor that directly impacts overall performance. On the other hand, for the offloading sceneries with small scale and small task requests, the LC scheme can smoothly address task redistribution. Nevertheless, in the case of large-scale tasks requests and offloading occurs, orchestration is importantly required to include optimized solutions for optimal offloading placement under different constraints.

7 Conclusion and Future Work

Cloud and edge computing are increasingly progressing into the fundamental infrastructure, enabling the potential connection of billions of IoT devices. Efficient collaboration schemes and algorithms of task offloading can be utilized to allow edge cloud services and mobile devices to work cooperatively. To this end, in this study, the quantitative impact of different offloading architectures and policies on different IoT applications' performance is analyzed and evaluated and discussed their effectiveness in the case of increasing IoT devices' number and the requested resources.

Future research will investigate the possibility of independently determining the optimal deployment mechanism for the proposed system. In addition, we will implement different offloading algorithms in a customized offloading library that other real-world systems can quickly adapt.

Acknowledgement: This work was supported by the Deanship of Scientific Research, Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia. Besides, the authors thank the Deanship of Scientific Research, Taibah University, Al-Madinah, Saudi Arabia for their equipment supply and resources.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Greenough, "The internet of everything," [Online]. Available: https://www.businessinsider.com/internet-of-everything-2015-bi-2014-12. 2015.

[2] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker *et al.,* "Internet of things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, vol. 1, no. 2011, pp. 9–52, 2011.

[3] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] S. Yi, C. Li and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. of the Int. Symp. on Mobile Ad Hoc Networking and Computing*, Hangzhou, China, pp. 37–42, 2015.

[5] S. Shekhar and A. Gokhale, "Dynamic resource management across cloud-edge resources for performance-sensitive applications," in *Proc. of the 17th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*, Madrid, Spain, pp. 707–710, 2017.

[6] M. Satyanarayanan, P. Bahl, R. Cáceres and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[7] Y. Zhang, R. Yang, T. Wo, C. Hu, J. Kang *et al.,* "Cloudap: improving the QoS of mobile applications with efficient vm migration," in *Proc. of the 10th IEEE Int. Conf. on High Performance Computing and Communications & IEEE Int. Conf. on Embedded and Ubiquitous Computing*, Zhangjiajie, China, pp. 1374–1381, 2013.

[8] K. Toczé and S. Nadjm-Tehrani, "A taxonomy for management and optimization of multiple resources in edge computing," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 23, 2018.

[9] G. Premsankar, M. Di Francesco and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[10] W. Li, I. Santos, F. C. Delicato, P. F. Pires, L. Pirmez *et al.,* "System modelling and performance evaluation of a three-tier cloud of things," *Future Generation Computer Systems*, vol. 70, pp. 104–125, 2017.

[11] A. Hegyi, H. Flinck, I. Ketyko, P. Kuure, C. Nemes *et al.,* "Application orchestration in mobile edge cloud : Placing of IoT applications to the edge," in *2016 IEEE 1st Int. Workshops on Foundations and Applications of Self-Systems*, Augsburg, Germany, pp. 230–235, 2016.

[12] T. Q. Dinh, S. Member, J. Tang, Q. D. La, T. Q. S. Quek *et al.,* "Offloading in mobile edge computing : Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[13] J. Du, L. Zhao, J. Feng and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[14] J. Liu, Y. Mao, J. Zhang and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. of the IEEE Int. Symp. on Information Theory*, Barcelona, Spain, pp. 1451–1455, 2016.

[15] T. G. Rodrigues, K. Suto, H. Nishiyama and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[16] X. Wei, C. Tang, J. Fan and S. Subramaniam, "Joint optimization of energy consumption and delay in cloud-to-thing continuum," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2325–2337, 2019.

[17] B. Yang, W. K. Chai, G. Pavlou and K. V. Katsaros, "Seamless support of low latency mobile applications with NFV-enabled mobile edge-cloud," in *Proc. of the 5th IEEE Int. Conf. on Cloud Networking*, Pisa, Italy, pp. 136–141, 2016.

[18] D. Zeng, L. Gu, S. Guo and Z. Cheng, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[19] S. Wang, M. Zafer and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[20] W. Tärneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker *et al.,* "Dynamic application placement in the mobile cloud network," *Future Generation Computer Systems*, vol. 70, no. 1, pp. 163–177, 2017.

[21] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[22] D. G. Roy, D. De, A. Mukherjee and R. Buyya, "Application-aware cloudlet selection for computation offloading in multi-cloudlet environment," *Journal of Supercomputing*, vol. 73, no. 4, pp. 1672–1690, 2017.

[23] R. Mahmud, R. Kotagiri and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Proc. Internet of Everything*, Singapore, Singapore, pp. 103–130, 2018.

[24] S. K. Sharma and X. Wang, "Live data analytics with collaborative edge and cloud processing in wireless IoT networks," *IEEE Access*, vol. 5, pp. 4621–4635, 2017.

[25] A. Yousefpour, G. Ishigaki and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *Proc. of the IEEE Int. Conf. on Edge Computing*, Honolulu, HI, USA, pp. 17–24, 2017.

[26] N. Wang, B. Varghese, M. Matthaiou and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE Transactions on Services Computing*, vol. 13, no. 6, pp. 1086–1099, 2017.

[27] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang *et al.,* "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 2, 2019.

[28] I. Farris, T. Taleb, H. Flinck and A. Iera, "Providing ultra-short latency to user-centric 5G applications at the mobile network edge," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 4, pp. e3169, 2018.

[29] C. Li, J. Tang, H. Tang and Y. Luo, "Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment," *Future Generation Computer Systems*, vol. 95, no. 99, pp. 249–264, 2019.

[30] R. Mahmud, F. L. Koch and R. Buyya, "Cloud-fog interoperability in IoT-enabled healthcare solutions," in *Proc. of the 19th Int. Conf. on Distributed Computing and Networking*, Varanasi, India, pp. 32, 2018.

[31] C. Sonmez and A. Ozgovde, "EdgeCloudSim : An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, pp. e3493, 2018.

[32] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.