IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received November 26, 2020, accepted December 9, 2020, date of publication January 18, 2021, date of current version January 26, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3052458*

# Improving IoT Services Using a Hybrid Fog-Cloud Offloading

**SAIF ALJANABI AND ABDOLAH CHALECHALE**
Department of Computer Engineering, Faculty of Engineering, Razi University, Kermanshah 6714414971, Iran
Corresponding author: Abdolah Chalechale (chalechale@razi.ac.ir)

**ABSTRACT** With the rapid development of the internet of things (IoT) devices and applications, the necessity to provide these devices with high processing capabilities appears to run the applications more quickly and smoothly. Though the manufacturing companies try to provide IoT devices with the best technologies, some drawbacks related to run some sophisticated applications like virtual reality and smart healthcare-based are still there. To overcome these drawbacks, a hybrid fog-cloud offloading (HFCO) is introduced, where the tasks associated with the complex applications are offloaded to the cloud servers to be executed and sent back the results to the corresponding applications. In the HFCO, when an IoT node generates a high-requirement processing task that cannot handle itself, it must decide to offload the task to the cloud server or to the nearby fog nodes. The decision depends on the conditions of the task requirements and the nearby fog nodes. Considering many fog nodes and many IoT nodes that need to offload their tasks, the problem is to select the best fog node to offload each task. In this paper, we propose a novel solution to the problem, where the IoT node has the choice to offload tasks to the best fog node or to the cloud based on the requirements of the applications and the conditions of the nearby fog nodes. In addition, fog nodes can offload tasks to each other or to the cloud to balance the load and improve the current conditions allowing the tasks to be executed more efficiently. The problem is formulated as a Markov Decision Process (MDP). Besides, a Q-learning-based algorithm is presented to solve the model and select the optimal offload policy. Numerical simulation results show that the proposed approach has superiority over other methods regarding reducing delay, executing more tasks, and balance the load.

**INDEX TERMS** Internet of Things, cloud computing, fog computing, task offloading, Q-learning.

## I. INTRODUCTION

New digital devices that use emerging networking technologies generate sophisticated tasks, which require broad processing capabilities. These sophisticated tasks, which are presented as online applications increase the needs of the processing power, required data rates, and other different resources [1]–[3]. For these reasons, new digital devices have been designed to be able to meet these requirements. However, with all the powerful features embedded in the latest devices, these devices are not able to run new smart applications that require high processing capabilities such as virtual reality, smart healthcare, some internet of things (IoT) applications, and so on [4]–[7]. In comparison to desktop computers and even mobile phones, it can be said that the IoT nodes have limited capabilities due to the functions it can perform in factory environments, hospitals and many other places [8].

For this, the IoT nodes are often small in size, which leads to limited processing capacities as well as energy consumption considerations. To overcome these, the concept of task offloading was proposed, where the smart applications can be executed in a remote-way on a high-performance entity [1]–[8]. For this, these applications must be designed capable to contact with other components in the network. In other words, other network components have to be able to run these applications instead of end-nodes devices. The network components that execute the applications on behalf of the end-nodes are referred to as fog nodes or cloud servers. After performing the applications, the results must be sent back to the end nodes to be used by the starting application. Task offloading technique is suitable for some kind of networks, where the end-nodes have limited resources to run specific applications [10]–[12]. At the same time, a cloud sever or fog nodes are exist in the system and are

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenyu Zhou.

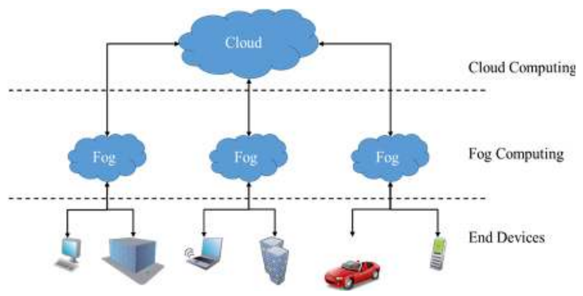

**FIGURE 1.** **The structure of the hybrid fog-cloud computing.**

compatible to coexist with the end-nodes and to execute their applications. End-nodes can be mobile phones in normal places, sensors in factories or sensing fields, or other control devices in special cases.

Generally, from network administration's point of view, the task-offloading technique is needed to achieve different goals. For example, providing computational resources to help edge-nodes to execute high complex tasks, balancing traffic load, save energy in un-rechargeable devices, or for some other QoS metrics related to latency, especially for real-time applications [13]–[15].

From another side, cloud computing has gotten more attention to perform high-complex jobs in different areas. Therefore, the cloud is provided with great capabilities related to large storage, high-speed communication, and powerful processing units. One of the essential features presented in cloud computing is that they can be accessed online at any time and from anywhere [16]–[20]. Many cloud-computing providers are now on the business side, offering cloud-computing services in the form of virtual machines. Based on [1], about 90 percent of internet users benefit from some cloud-computing services in different ways.

Sometimes, edge-node devices or IoT devices can be located far away from the cloud. This results in a considerable delay, which is not suitable for some applications, especially for real-time applications, which are sensitive to delay. To overcome this problem, fog-computing was proposed in which fog nodes are located between cloud servers and edge-node devices, and close to the end-nodes. Many types of fog computing, such as cloud-lets, mobile edge computing, were proposed and discussed in the literature [1], [2], [4], [6], and [8], [9]. Fog nodes are distributed to be close to the IoT devices, thereby covering more larger areas. Figure 1 shows how fog computing consists of distributed fog-nodes, which work as a middleware between cloud-computing and IoT devices. The fog-nodes that are distributed at different locations make it easier for the end devices to exploit resources, and reduce latency. Some examples of fog computing can be central servers, controllers, or base stations.

To meet the QoS requirements of IoT nodes, task-offloading technique is used to send some jobs or tasks to the cloud or to the fog nodes. This technique is expected to solve the problem of running heavy applications on resource-poor devices. The main problem here is to specify when and where task offloading must be performed. In other words, "when" an IoT node decides to offload a task, and "where" to offload it. The answer to the question "when" depends on the running application requirements and the currently available resources on the operator node. On the other hand, the answer to the question "where" depends on the application specifications that has to be offloaded and the network state. In other words, an IoT node needs to decide to which place a task should be offloaded; to a certain fog node or to the cloud.

Many fog nodes with different resources capabilities and delay constraints can exist nearby to IoT nodes. Therefore, the problem here is to select the best fog node that can run the task under the constraints to meet the QoS requirements of IoT nodes. Another choice is that IoT nodes can offload a task directly to the cloud. In these cases, IoT nodes are considered as decision-makers. In some different cases, fog nodes can be the decision-makers, which decide to offload a task (or some tasks) to other fog nodes or to the cloud according to the changes in the network state aiming at balancing load between fog nodes.

Based on the above background, the main contributions of this paper can be summarized as follows:

1. We consider the task offloading problem by taking both load balancing and delay minimizing into account. In the previous related works, only one parameter was considered during task offloading.
2. To choose the best fog node for each new task, sequential decisions need to be taken. Therefore, task-offloading problem is modelled as an MDP, where two decision-makers (IoT users and fog nodes) can make decisions to where to offload tasks. To the best of our knowledge, this is the first work for sequential decisions that consider two decision-makers.
3. We presented a Q-Learning task offloading scheme to solve the MDP model. The main advantage of Q-Learning is that it can solve MDP without the explicit specification of the transition probabilities, which are needed in policy iteration and value iteration.
4. Simulation results show that the proposed approach makes better performance than other related works concerning load balancing and minimizing delay.

The structure of this paper is organized as follows. In Section II, we review some related works, which discuss the task-offloading problem. Section III presents the imperatives on why we need to offload tasks. Some scenarios of task offloading are discussed in details in Section IV. System description and assumptions are provided in Section V. Section VI, presents modelling the problem as MDP and present a Q-learning-based algorithm to determine a better choice. Formulation and simulation results are described in Sections VII and VIII, respectively. Finally, Section IX concludes the paper.

## II. LITERATURE REVIEW

Since the objective of this research is to study task offloading, we review some works that studied criteria on how to offload tasks from IoT nodes to corresponding servers. Many works can be found which discussed the task offloading problem because this problem becomes more attractive recently, especially for emerging communication technologies. In [2], task offloading is used to balance the load between two fog nodes, where load balancing was done using a cooperative model between two fog-nodes. In the same way, the authors in [3] discussed the load-balancing problem between multiple fog nodes. The fog nodes were represented as data centers, which are connected to construct a ring network. In this scheme, if a task is offloaded to one data center with its task buffer full, then the task is forwarded to the other cooperating data center and served by that data center, consuming its CPU cycles.

A task offloading problem to reduce the latency of IoT devices was discussed in [4]. In this paper, first, IoT devices offload a task to one fog node which in turn may process the task, forwards it to the cloud, or offloads it again to other fog nodes. Also, no constraints on the number of fog nodes or the type of network structure were considered in details. In [5], the radio access network is integrated with cloud and fog computing to manage the network efficiently, aiming at minimizing end-to-end delay for real-time applications in 5G networks.

Another technique to manage tasks and to distribute them among fog nodes is task assignment. In this technique, all tasks that could be generated at each IoT node is assumed to be known. Therefore, each task is pre-allocated to one fog node before generating the task. In other words, the task assignment is static and cannot be changed during running time. Before task offloading becomes necessary to be done, task assignment may solve the problem of delay minimizing or load balancing. Task assignment assigns a task to one fog node or the cloud servers. If the task assignment considers the state of all fog nodes, further task offloading can be avoided. However, to optimally assign tasks, the states of all fog nodes and properties of applications are required. For example, a medical system was considered in [6], where the problem is to assign tasks to suitable base stations. In each base station, many virtual machines can exist. Therefore, the problem of virtual machine assignment was also discussed. The task assignment problem was also studied in [7], where energy consumption was considered while minimizing end-to-end delay. Another work that discussed task assignment in IoT structure was presented in [8], where energy consumption is considered as long as latency and also QoS requirements.

As mentioned before, task assignment is expensive and complex where the specifications of running applications and the states of all fog nodes must be known before start solving the task assignment problem. For this reason, task offloading is more practical in the real world to be applied; for example, the authors in [9] modelled task assignment problems as an optimization problem. Since solving the optimization problem is complex and must be centralized, the authors then presented a distributed algorithm to handle the task assignment problem. Nevertheless, many important parameters, like processing power and traffic load, were not considered carefully in the distributed algorithm. Besides, fog nodes need to send their states periodically to IoT nodes, which does not apply to the large-scale networks.

The main objective of our research is to minimize end-to-end delay and to distribute tasks between fog nodes by minimizing the number of offloading operations needed to assign a task to a suitable fog node. Our proposed approach is different from the works discussed above such as [4]–[6], [8], [9], and has no constraints on the type of network or the number of fog nodes or IoT nodes like [2], [3].

The authors in [10] studied the energy consumption problem with the constraint that the latency of IoT applications has to be more than a predefined threshold. Energy minimization was done by optimally assigning tasks to fog nodes. Another offloading mechanism was presented in [11], where the goal is to minimize the total response time of all tasks. This mechanism can work online to assign tasks to fog nodes. The work [12] also aimed at minimizing response time in a distributed way under the constraint that energy consumption must be less than a threshold. Moreover, this work focused also on task offloading from fog nodes to cloud servers.

Fog network combines all fog nodes in a system to construct a network. The authors in [13] presented a new framework to merge the fog nodes in one entity to construct a network. This network can manage the resources of fog nodes efficiently, where it can assign each task to a fog node that can meet the QoS requirements such as latency. In [14], the latency was also considered as the main objective, and the fog network is studied, where the tasks that are executed periodically are cached for future purposes. The task offloading algorithm presented in [15] is based on graph theory and allow each IoT device to build its fog network from the nearby fog nodes. When a new task is generated at one IoT device, it can offload it directly to the fog network. The work in [16] focused on one task offloading for the IoT devices that are used in smart cities. Moreover, task offloading in mobile cloud computing was discussed in [17], where no fog computing exists in the system. Therefore, an IoT device can offload tasks only to cloud servers.

Mobile edge computing (MEC) works in a similar way like fog computing that is adopted and developed for the cellular networks such as 5G [18]. In the structure of MEC, computing nodes are provided in the radio access network (RAN), which is referred to as Cloud-RAN. As a result, task offloading can be optimized by network operators. Task offloading is also an important issue in MEC and has increased attention in the literature. For example, the authors in [19] studied the problem of joint channel assignment and task offloading aiming at reducing latency and energy consumption. Similarly, a joint channel assignment and task offloading problem were studied in [20]. However, in this work, the decision is taken in a centralized model where the authors assumed that MEC nodes have no constraints on computation resources.

More recent works was presented in [21]–[25]. In [21], collaborative cloud-edge-end task offloading is proposed to handle the task offloading problem. The main goal is to

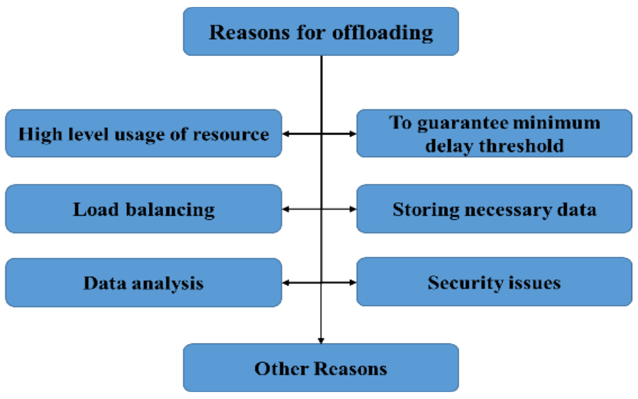

**FIGURE 2.** Criteria used in offloading.

increase the processing efficiency of the tasks considering the limited resources and communication's limitations. In this study, mobile devices partially process the tasks, and based on the tasks' priorities, these tasks are offloaded to the edge nodes and the cloud server. To organize task offloading, the authors proposed pipeline-based offloading tasks. The problem is modeled as a non-convex problem, which is hard to solve. Therefore, an approximation model was proposed.

Task offloading problem is also studied in [22], where, module placement method was proposed by classification and regression tree. This algorithm was proposed in order to select the best fog node to offload a task. This method considered the energy consumption when offload a task, and based on the energy consumption, the decision to offload a task is taken.

A block chain and learning-based method task offloading for vehicular fog computing was proposed in [23]. The aim is to reduce task offloading delay, queuing delay, and handover cost with incomplete information while simultaneously ensuring privacy, fairness, and security. A subjective logic-based trustfulness metric to quantify the possibility of task offloading success was designed. An online learning-based intelligent task offloading algorithm was proposed, which can learn the long-term optimal strategy and achieve a well-balanced tradeoff among task offloading delay, queuing delay, and handover costs.

An ant colony optimization and particle swarm optimization were used to efficient task offloading for IoT-based applications in fog computing [24]. Here, the main goal is to load the balance between fog nodes.

In [25], the aim is to provide a new offloading technique where latency for task offloading is optimized by taking Peer-to-Peer (P2P) technology as a basic mode of a network environment for fog computing and also taken P2P file-sharing protocol as a basic mode of offloading technique.

## III. IMPERATIVES OF TASK OFFLOADING

This section discusses reasons for needing the tasks to be offloaded, where the decision to offload a task depends on these reasons (or criteria) [26]. These criteria answer the question of why to offload a task. Figure 2 shows the most common reasons that require tasks to be offloaded, and they are explained in details below.

### A. HIGH-LEVEL USAGE OF THE RESOURCE

High-level usage of the resource happens when the usage level of device resources such as CPU or RAM exceeds a threshold. In this situation, some applications that occupy the largest portion of resources have to be offloaded to other devices such as fog nodes or cloud servers. If this happens in a fog node, the applications must be offloaded to other fog nodes or cloud servers. For example, video processing which needs a powerful CPU.

### B. TO GUARANTEE A MINIMUM DELAY THRESHOLD

When some applications offload tasks to the cloud server, if these applications are delay-sensitive, then these tasks need to be offloaded to nearby fog nodes to guarantee that the delay remains acceptable. The authors in [27] discussed in details task offloading to meet delay constraints. For example, in video streaming services, cloud servers need to offload the video to fog nodes that are close to end-users.

### C. LOAD BALANCING

Load balancing happens when there is a large number of end-users in a certain location, and offload their tasks to a sub-set of fog nodes, while other fog nodes have a smaller number of tasks to process. In this case, busy fog nodes may offload some tasks to other fog nodes to balance the load among them. Such a problem was studied in [28], where one fog node with a high-level load can distribute some tasks to other fog nodes. For example, a fog network manager in a datacenter distributes new arrival tasks among fog nodes to guarantee similar levels of occupation in them.

### D. STORING NECESSARY DATA

To store a large amount of data, end-users' devices, are not a good place because they are not reliable, and there is no enough capacity to store all data. Therefore, data that are needed to be stored is offloaded to the cloud server. In the case of data storage, even fog nodes maybe not a good choice.

### E. DATA ANALYSIS

In some scenarios, a manager in a cloud server may need to make some decisions on the system-level. Therefore, data from all end-node devices and even fog nodes need to be offloaded to cloud servers. In addition to data analysis, data may be offloaded to be organized or to remove redundant and replicated data. Some works studied tasks offloading in terms of data organization and data analysis, such as [28], [29].

### F. SECURITY ISSUES

Security issues are similar to storing data, where the end-user nodes are not the perfect place to keep data secure or to save privacy. For example, personal information may be offloaded from smartphones to cloud servers, such as Dropbox, OneDrive, and Google Drive. Another example is that the information related to patients in a body area network is offloaded from medical machinery to a data center to keep the privacy [30].

### G. OTHER CRITERIA

Task offloading can be done for other reasons such as availability or accessibility. In this situation, end-nodes may offload tasks to be available to other nodes and to be accessible from anywhere at any time.

## IV. TASK OFFLOADING SCENARIOS

In the previous section, the reasons that make it necessary to offload tasks were discussed. In this section, we study where the task offloading may happen. Task offloading can take place at almost all system components based on the reasons of why we need to offload tasks. In what follows, we classify the possible locations.

### A. FROM PERIPHERAL EQUIPMENT TO EDGE NODES

This scenario is simple and is related to special cases such as smart glasses or smartwatches that are designed to offload all gathered data to smartphones.

### B. FROM IOT NODES TO FOG NODES

#### 1) FACTORY SCENARIO

Factory scenario is the most common type of task offloading where IoT nodes do not have enough resources to process all data. For example, IoT nodes in industrial areas are nodes that provide sensors, weak CPUs, and low storage capacities. Therefore, these IoT nodes almost offload all tasks to fog nodes to process gathered data and to take decisions. Fog nodes may be local data centers.

#### 2) HEALTHCARE SCENARIO

This scenario can also be found in hospitals where sensor nodes that are responsible for patient monitoring offload tasks towards fog nodes that may be a central device in the nurses' room. Also, in the field of health surveillance, in the body area network, which consists of sensors installed in/on the patient's body, the same task of offloading type can be noted. Sensor nodes in this network send signals related to different organs of the body to sink node, which in its turn sends these signals to the local server representing a fog node.

#### 3) VEHICULAR AD-HOC NETWORK (VANET) SCENARIO

In VANET, vehicles send messages about street traffic, accidents, or other information directly to other vehicles or the fog nodes. Fog nodes in this scenario can be roadside units (RSUs), which are installed in predefined locations to monitor and control traffic. In this scenario, vehicles offload most tasks to RSUs because RSUs have better conditions to distribute messages and to broadcast alarms when needed. Some works studied this type of task offloading like [31] to [32]. In [31], the authors discussed real-time services in VANET, while the works in [32] and [33] studied the problem of distributing alarm messages when needed.

### C. FROM IOT NODES TO THE CLOUD

IoT nodes may offload tasks to the cloud server for many reasons. The common scenario is that IoT nodes offload tasks to store data, in body area networks, where fog nodes may not have enough capacity or reliability to store such important data [34]. Another example can be security issues, and in this case, it is preferable to save data in cloud servers than fog nodes. Processing purposes can also be a common example when fog nodes cannot perform tasks under some QoS requirements such as latency.

### D. FROM FOG NODES TO THE CLOUD

When many tasks are offloaded from IoT nodes to the fog nodes and consume their resources, in this case, fog nodes may offload some tasks to cloud servers. Storing and managing data can be another reason to offload tasks from fog nodes to cloud servers. In some scenarios, if there are no direct connections between IoT nodes and cloud servers, IoT nodes may offload a task to fog nodes first, and fog nodes, in turn, offload the task to the cloud.

### E. FROM FOG NODE TO OTHER FOG NODES

The main goal in this scenario is to balance the load between fog nodes. For example, one base station with a high load may decide to offload some tasks to other fog nodes with a lower load. This type of offloading can be done by fog nodes themselves or can be decided by the cloud that can have a complete view of the system. Another reason for this type of offloading may be accessibility or availability concerns. Some information is required for all IoT nodes, and if this information is stored only in some fog nodes; some other fog nodes may not have access to this information. Therefore, this information has to be offloaded to all fog nodes. However, for accessibility purposes, the offloaded information and the original one has to be synchronized if it is updated somewhere.

### F. FROM CLOUD TO FOG NODES

This scenario happens in different cases. For example, in multimedia services, cloud servers may offload a part of the video to some fog nodes to meet delay requirements. In this case, fog nodes have to provide the service instead of the cloud. Tasks are offloaded to the fog nodes that are as close as possible to the users. When there is a high-level interaction between cloud and end-users with no need for central processing, it is better to offload a task to nearby fog nodes. Interaction between RSUs and vehicles in VANET is a good example [35]. In 5G communication, when two connecting nodes located under the coverage of one base station, the cloud gives this fog node the right to manage the connection.

### G. MUTUAL OFFLOADING BETWEEN CLOUD AND FOG NODES

In the previous scenarios, task offloading is done from fog nodes to the cloud and vice versa. This scenario combines these two scenarios where information related to one task needs to be offloaded from cloud to fog nodes, and the fog nodes in turn offload other needed information to the cloud. Target tracking can be a good example, where fog

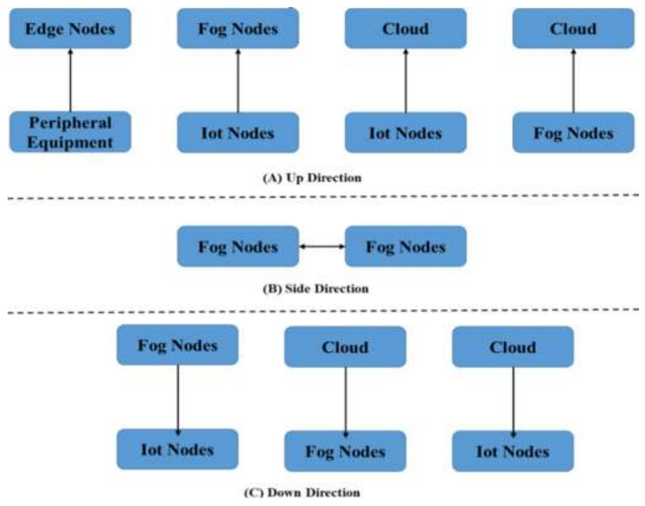

**FIGURE 3.** **Offloading scenarios in hybrid fog-cloud computing.**

nodes send the current location to the cloud, and the cloud analyzes this information and send some results to fog nodes to stop/start/continue tracking the target.

### *H. FROM CLOUD TO IOT NODES*

This is not very common, but it happens when the cloud needs to collect some information from the field or environment, for example, to get the coordinates of some events such as fire or explosion. In this case, cloud offloads a task to appropriate IoT nodes. Another example happens when the cloud needs to control something by controllers or actuators. Such types of task offloading were discussed in literature like [36]–[38].

To this end, three directions of offloading can be noted: Up direction, down direction, and Side direction. Figure 3 shows the types and directions in which task offloading can take place. In the case of cloud-to-fog nodes, cloud-to-IoT nodes, fog nodes-to-IoT nodes, and peripheral equipment to edge nodes are classified as Up direction. There is only one side-by-side case between fog nodes as explained in Section IV, down direction includes the cases of cloud-to-fog nodes, cloud-to-IoT nodes, and fog nodes-to IoT nodes.

## V. MOTIVATIONS AND SYSTEM DESCRIPTION

In this research, the main goal is to decide where to offload a task, to one fog node, or to the cloud servers. This decision can be made at IoT nodes, at the fog nodes, or even at the cloud servers. In [4], the decision is first made in fog nodes, where the IoT nodes directly offload the tasks to the fog nodes. The fog nodes then decide to perform the task, offloading it to other fog nodes, or offload it to the cloud servers. In the mentioned scheme, IoT nodes cannot send tasks directly to the cloud server. In [39], the offloading decision on a task is made in IoT nods, where IoT nodes decide to offload the task to either a fog node or the cloud servers. In [39], there is no decision making in fog nodes. The main drawback of [4] is that IoT nodes cannot directly offload tasks to the cloud servers, while the main challenge in [39] is that there is no offloading between fog nodes, which results in unbalanced service time.

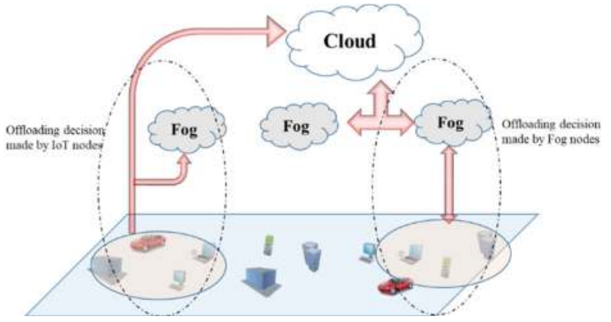
Cloud
Offloading decision made by IoT nodes
Fog
Fog
Fog
Offloading decision made by Fog nodes

**FIGURE 4.** **Possible locations of decision making.**

Motivated by the above discussion, we proposed HFCO, a new decision-making scheme where the decisions can be made in IoT nodes or in fog nodes, which results in better decisions regarding latency and load balancing. The main objectives can be summarized as follows:

1. Minimizing the service delay: the main goal is to minimize the service delay by making the best decision to offload a task to the cloud servers or a suitable fog node.
2. Minimizing the number of offloading operations: The service delay can also be improved by minimizing the number of offloading operations. For example, under some conditions, an IoT node may decide to offload a task to a fog node, which in turn decides to offload the task to other fog nodes or the cloud server. If the IoT node made a better decision from the beginning, service time could be improved dramatically. Therefore, making the optimal decision to where to offload the task is an important factor, which can reduce service time. Figure 4 depicts these two offloading models.
3. Load balancing: making the best decisions helps to select fog nodes with low utilization levels will balance the load between fog nodes, thereby reducing waiting time.
4. Exploiting the data about fog nodes: making the best decision requires many statistics to be known at IoT nodes, e.g. processing capability, utilization level, average waiting time, etc. For making good decisions (to offload to a fog node or the cloud), IoT nodes need some information about fog nodes.

## VI. SYSTEM MODEL

We considered one cloud computing server $\mathcal{C}$, a set of fog nodes $\mathcal{F} = \{F_1.F_2.\cdots.F_M\}$, and a set of IoT nodes $\mathcal{T} = \{T_1.T_2.\cdots.T_N\}$, where $M$ and $N$ are the number of fog nodes and IoT nodes in the considered area, respectively. Each IoT node $T_i$ periodically generates new tasks, where each task is referred to as $\omega_{h,i}$, where $h$ is the serial number of the tasks on IoT node $T_i$. Each task has QoS requirements represented by minimum acceptable delay $D_{h,i}$, and required resources represented by $R_{h,i}$. On the other side, each fog node $F_j$ has a determined amount of resources. Also, we consider the following assumptions:

1. Each fog node can estimate its currently available resources and calculate the response time based on the current tasks in the queue.

2. The IoT nodes have limitations in processing capacity, and most tasks need to be offloaded to the nearby fog nodes to be executed.

3. When a new task is generated by one IoT node, this IoT node asks the nearby fog nodes if they have enough resources to perform the task. The fog nodes that can perform the task send the estimated response time to the corresponding IoT node. This coordination is done rapidly via a special channel; therefore, the time duration of this coordination is neglected. This special channel can be allocated as a portion of the bandwidth, which is defined only for control packets.

Assuming that IoT nodes have choices to select the better fog nodes to offload their new tasks, an IoT node needs to know the states of all nearby fog nodes. On the other hand, an IoT node needs to decide if the new task has to be offloaded directly to cloud servers or a to fog node. This decision depends on the processing requirements and delay requirements of the task and the states of the fog nodes. Besides, fog nodes can offload some tasks to each other or to the cloud server.

Since the objective is to minimize service delay by offloading a task to the cloud servers or to a good fog node, the problem can be formulated as an optimization problem to find the optimal decisions. However, solving an optimization problem for each new task may take a long time, which is not practical. Markov decision process (MDP) can be used here, where, a reinforcement learning-based heuristic algorithm is provided to solve the MDP model to overcome the large state and action spaces.

The problem of selecting the best choice to offload a task is formulated as an MDP, which is widely used to formulate decision making in network structures as in [40] and [41]. MDP consists of quadruple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, in which $\mathcal{S}, \mathcal{A}, \mathcal{P}$ and $\mathcal{R}$ define a finite set of states, a finite set of actions, the transition probability matrix for each action in each state, and the corresponding reward obtained from each action in each state, respectively [42]. Since that both IoT nodes and fog nodes can make decisions, there are two decision-makers in this problem. Therefore, the set of state space, the set of action space, reward, and transition probability matrix must be defined for both types of nodes. The notations used in this paper are presented in Table 1.

## VII. FORMULATION

In the following formulation, we consider IoT nodes as decision-makers, similarly, the formulation of the fog nodes as decision-makers can be derived in the same way.

### A. STATE SPACE

The state of an IoT node $T_i$ at time slot $k$ is determined based on the available resources $\Psi^j(k)$ and the response time $\Theta^j(k)$ of each nearby fog node $F_j \in \mathcal{F}_i^n \subset \mathcal{F}$, and the response time of cloud server $\Theta^{\mathcal{C}}$. The resources at the cloud server $\mathcal{C}$ is always sufficient to handle any task. Therefore, we do not consider it here. However, the cloud server's response time is important to some tasks. Therefore, regarding the cloud server, we consider response time but not resources. Denote by $\psi^j_{r,r'}(k)$ to the probability that the available resource state of the fog node $F_j$ changes from the state $r$ to the state $r'$ in time slot $k$, the transition probability of the available resources state is defined as

$$\psi^j_{r,r'}(k) = \Pr\left\{\Psi^j(k+1) = r' | \Psi^j(k) = r\right\}, \quad \forall r, r' \in \mathcal{S}_\Psi \tag{1}$$

where $\mathcal{S}_\Psi$ is the set of the possible states of the available resources.

Similarly, if we denote by $\Theta^j(k)$ to the response time of the fog node $F_j$ in time slot $k$, and by $\theta^j_{d,d'}(k)$ to the probability that response time of $F_j$ changes from the state $d$ to $d'$ in time slot $k$. The transition probability of the response time state is defined as:

$$\theta^j_{d,d'}(k) = \Pr\left\{\Theta^j(k+1) = d' | \Theta^j(k) = d\right\}, \quad \forall d, d' \in \mathcal{S}_\Theta \tag{2}$$

where $\mathcal{S}_\Theta$ is the set of the possible states of the response times.

Therefore, the state of one fog node $F_j$ regarding available resources and response time in the time slot $k$ consists of a couple of sub-states and is represented as

$$\Delta^j(k) = \left(\Psi^j(k), \Theta^j(k)\right) \tag{3}$$

When one IoT node decides to offload a task, it must select one of the nearby fog nodes (or the cloud server). As a result, the state of an IoT node $T_i$ in time slot $k$ depends on the states of these fog nodes and the cloud server and is determined as

$$S_i(k) = \left[\Delta^1(k), \Delta^2(k), \ldots, \Delta^{N_i}, \Theta^{\mathcal{C}}\right] \tag{4}$$

where $N_i = \left|\mathcal{F}_i^n\right|$ is the number of the nearby node of $T_i$. Therefore, the state space of all IoT nodes will be

$$\mathcal{S}(k) = \{S_1(k), S_2(k), \ldots, S_N(k)\} \tag{5}$$

On the other side, the state of one fog node depends on the states of nearby fog nodes and is represented as

$$S_j(k) = \left[\Delta^1(k), \Delta^2(k), \ldots, \Delta^{N_j}\right] \tag{6}$$

where $N_j = \left|\mathcal{F}_j^n\right|$ is the number of nearby fog nodes of $F_j$.

Note that the response time of the cloud server is not considered in the state space of the fog nodes because for the fog node, it is important to balance the load between the fog nodes. While for IoT nodes, it is important to minimize response time in addition to balance the load between the fog nodes. Similarly, the state space of all fog nodes

$$\mathcal{S}_F(k) = \{S_1(k), S_2(k), \ldots, S_M(k)\} \tag{7}$$

### B. ACTION STATE

The action for each IoT node $T_i$ in time slot $k$ can be chosen from $N_i + 1$ actions represent the set of nearby fog nodes in addition to the cloud server. Therefore, the IoT node $T_i$ takes the action $a_i(k)$ in time slot $k$, where the set of all available actions is

$$\mathcal{A}_i(k) = \left\{a_1(k), a_2(k), \ldots, a_{N_i}(k), a_{\mathcal{C}}(k)\right\}, \tag{8}$$

**TABLE 1. Variables and functions for Q-Learning method.**

| *Variable/ function* | *Explanation* | *Variable/ function* | *Explanation* |
|---|---|---|---|
| $\mathcal{C}$ | *The cloud computing server* | $S_i(k)$ | *The state of an IoT node $T_i$ in time slot k includes all the member of $\mathcal{F}_i^n$* |
| $\mathcal{F}$ | *The set of fog nodes* | $S_j(k)$ | *The state of a fog node $F_j$ in time slot k includes all the member of $\mathcal{F}_j^n$* |
| $\mathcal{T}$ | *The set of IoT nodes* | $\mathcal{S}(k)$ | *The state-space of all IoT nodes* |
| $M$ | *The total number of fog nodes $\lvert\mathcal{F}\rvert$ in the considered area* | $\mathcal{S}_F(k)$ | *The state-space of all* fog *nodes* |
| $N$ | *The total number of IoT nodes $\lvert\mathcal{T}\rvert$ in the considered area* | $\Pr\{a\lvert b\}$ | *The probability of a condition to b* |
| $F_j$ | *The fog node number j* | $a_i(k)$ | *An available action for $T_i$ at time slot k* |
| $T_i$ | *The IoT node number $i$* | $A_j(k)$ | *An available action for $F_j$ at time slot k* |
| $\mathcal{F}_j^n$ | *The set of nearby fog nodes of fog node $F_j$* | $\mathcal{A}_i(k)$ | *The action set of $T_i$* |
| $\mathcal{F}_i^n$ | *The set of nearby fog nodes of the IoT node $T_i$* | $A_j(k)$ | *The action set of $F_j$* |
| $N_i$ | *The number of member of set $\mathcal{F}_i^n$* | $P_{ss'}^a$ $Pf_{ss'}^a$ | *The probability that the state changes from s in the time slot k to s′ in time slot k+1 when the action a is taken.* |
| $N_j$ | *The number of member of set $\mathcal{F}_j^n$* | $\rho_j(k)$ | *The normalized value of the available resources of $F_j$ in time slot k* |
| $\omega_{h,i}$ | *The task number h on IoT node $T_i$* | $\Theta^j(k)$ | *The normalized value of the response time of $F_j$ in time slot k* |
| $D_{h,i}$ | *The minimum acceptable delay for $\omega_{h,i}$* | $RWD_j(k)$ | *The obtained reward from $F_j$ in time slot k* |
| $R_{h,i}$ | *The required resources* | $T$ | *The future period in which the reward is calculated* |
| δ | *The finite set of states* | $RWD_i^k$ | *The cumulative discounted future reward for $T_i$ starting from k* |
| $A$ | *The finite set of actions* | β | *The discount factor* |
| $P$ | *The transition probability matrix for each action in each state* | π(s, a) | *Is the probability that policy π takes action in state s* |
| $R$ | *The set of the corresponding reward obtained from each action in each state* | $V^\pi(s)$ | *State value function* |
| $k$ | *The number of time slot* | $Q^\pi(s,a)$ | *Action value function* |
| $\Psi^j(k)$ | *The available resources on $F_j$ at time slot k* | $\alpha$ | *Learning rate* |
| $\Theta^j(k)$ | *The response time of each $F_j$ at time slot k* | | |
| $\Theta^{\mathcal{C}}$ | *The response time of $\mathcal{C}$* | | |
| $\psi_{r.r'}^j(k)$ | *The probability that $\Psi^j$ changes from the state $r$ to the state $r'$ in time slot k* | | |
| $\mathcal{S}_\Psi$ | *The set of the possible states of available resources $\psi^j$* | | |
| $\theta_{d.d'}^j(k)$ | *The probability that $\Theta^j$ changes from the state d to d′ in time slot k* | | |
| $\mathcal{S}_\Theta$ | *The set of the possible states of the response time $\Theta^j$* | | |
| $\Delta^j(k)$ | *The combined available resources and response time of one fog node $F_j$ in the time slot k* | | |

Similarly, in a certain time slot $k$, a fog node $F_j$ will take action from the set of the available actions $\mathcal{A}_j(k)$ that contains $N_j + 1$ actions represent the nearby fog nodes and the cloud server. When a fog node selects itself as the chosen action, this means that this fog node decides to keep the current tasks in its queue. In other words, no task offloading will be done. Similarly, the set of all available actions is

$$\mathcal{A}_j(k) = \left\{a_1(k), a_2(k), \ldots, a_{N_j}(k), a_{\mathcal{C}}(k)\right\} \tag{9}$$

### *C. TRANSITION PROBABILITY MATRIX*

In a time slot $k$, when an IoT node (or fog node) is in state $\boldsymbol{s} \in \boldsymbol{S_i}(\boldsymbol{k})$ and takes action $\boldsymbol{a} \in \boldsymbol{\mathcal{A}_i}(\boldsymbol{k})$, the state of the node will change to the state $\boldsymbol{s'}$ according to the following equation:

$$\boldsymbol{P_{ss'}^a} = \mathbf{Pr}\left[\mathbf{s}_i(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{s'} \lvert \mathbf{s}_i(\boldsymbol{k}) = \boldsymbol{s}, \boldsymbol{a_i}(\boldsymbol{k}) = \boldsymbol{a}\right] \tag{10a}$$

In the same way, the transition probability matrix for fog nodes is defined as

$$\boldsymbol{Pf_{ss'}^a} = \mathbf{Pr}\left[\mathbf{s}_j(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{s'} \lvert \mathbf{s}_j(\boldsymbol{k}) = \boldsymbol{s}, \boldsymbol{a_j}(\boldsymbol{k}) = \boldsymbol{a}\right] \tag{10b}$$

### D. REWARD

The reward for each fog node $F_j$ in time slot $k$ is defined based on the available resources $\Psi^j(k)$, and the response time $\Theta^j(k)$ of this fog node with related to other fog nodes. Generally, when an IoT node $T_i$ takes action $a \in \mathcal{A}_i(k)$ at state $s \in S_i(k)$ by selecting a fog node $F_j$ in time slot $k$ the following reward will be obtained in time slot $k+1$:

$$RWD_s^a(k+1) = \rho_j(k) - \delta_j(k) \tag{11}$$

where, $\rho_j(k) = \frac{\Psi^j(k)}{\sum_{F_{j'} \in \mathcal{F}_j^n \setminus \{F_j\}} \Psi_{j'}(k)}$, and $\delta_j(k) = \frac{\Theta^j(k)}{\sum_{F_{j'} \in \mathcal{F}_j^n \setminus \{F_j\}} \Theta_{j'}(k)}$ are the normalized values of the available resources and the response time, respectively. Based on this definition, the reward for each fog node increases when more resources are available, while it decreases by increasing the response time. In other words, an IoT node (or a fog node) chooses the fog node with more available resources and less response time. Since the available resources and response time are different in nature, we considered the normalized values of these parameters. The goal of the proposed algorithm is to choose the fog node that maximizes the cumulative discounted future reward. In other words, the aim is to select the optimal policy $\pi$ in a specific state s $(k)$ at time $k$ that can be represented as $\pi$ (s $(k)) = a(k)$. Since the aim is to consider the long-term reward, we compute the cumulative discounted future reward for $T_i$ and $F_j$ starting from $k$ according to the following formula

$$RWD_i^k = \mathbb{E}_\pi \left[ \sum_{t=0}^{T} \beta^t RWD_j(k+t+1) \right] \tag{12a}$$

$$RWD_j^k = \mathbb{E}_\pi \left[ \sum_{t=0}^{T} \beta^t RWD_j(k+t+1) \right] \tag{12b}$$

where, $\beta$ is the discount factor, and $T$ is the future time in which the reward is calculated. Based on Eq. (12), an IoT node (or a fog node) calculates the cumulative discounted future reward for all nearby fog nodes and set active the node with maximum reward, and the optimal fog node is specified as:

$$F_j^* = \arg \max_{F_j \in \mathcal{F}_i^n} \left\{ RWD_i^k \right\} \tag{13}$$

### E. Q-LEARNING-BASED TASK OFFLOADING ALGORITHM [1]

With large sizes of state space and action space, solving MDP by choosing the fog node with maximum reward becomes impractical. Therefore, we use Q-Learning in order to solve MDP to learn the optimal policy. Q-Learning is a model-free reinforcement learning (RL) which is used to select the optimal policy under a specific state. In this section, we present the Q-learning-based task offloading method to obtain the optimal policy. The IoT node $T_i$ acts as an agent to interact with the environment to obtain the best action. Accordingly, the environment will generate a corresponding reward. In what follows, we define some functions and variables regarding Q-learning. After that, these functions are transformed into Bellman equations.

[1] In this subsection, we consider the IoT nodes as decision-makers. The analysis can be then generalized for the fog nodes.

The aim for each IoT node $T_i$ is to learn a Markov policy, where each state is corresponded with a probability to take any available action, that $\pi : \mathcal{S} \times \mathcal{A} \longrightarrow [0,1]$, which maximize the expected discounted future reward starting from each state $s$. Therefore, we need to define the state value function $V^\pi$:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}\Big[R(k+1) + \beta.R(k+2) + \beta^2.R(k+3) \\ &\quad + \ldots | s(k) = s, \pi\Big] \\ &= \mathbb{E}\left[R(k+1) + \beta.V^\pi(s(k+1)) | s(k) = s, \pi\right] \\ &= \sum_{a \in \mathcal{A}} \pi(s,a) \left[R_s^a + \beta \sum_{s'} P_{ss'}^a V^\pi(s')\right] \end{aligned} \tag{14}$$

where, $R_s^a$ is calculated based on (11), and $\pi(s,a)$ is the probability that policy $\pi$ takes action $a$ in state $s$. The equation (14) is a Bellman equation that can be solved in different ways, such as Q-Learning, value iteration, or policy iteration. Therefore, we can determine the optimal policy based on the state value function:

$$\begin{aligned} V^*(s) &= \max_\pi V^\pi(s) \\ &= \max_{a \in \mathcal{A}} \mathbb{E}\left[R(k+1) + \beta.V^*(s(k+1)) | s(k) = s,\right. \\ &\quad \left. a(k) = a\right] \\ &= \sum_{a \in \mathcal{A}} \pi(s,a) \left[R_s^a + \beta \sum_{s'} P_{ss'}^a V^*(s')\right] \end{aligned} \tag{15}$$

According to the optimal Bellman equations given by (15), we can obtain the optimal policy $\pi^*$:

$$\pi^* = \arg_\pi \max_{a \in \mathcal{A}} \beta \sum_{s'} P_{ss'}^a V^*(s') \tag{16}$$

In Q-Learning, in addition to the state value function, we need to define action-value function $Q^\pi(s,a)$ for policy $\pi$, which is used to select the best action in each state and is defined as:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}\Big[R(k+1) + \beta.R(k+2) + \beta^2.R(k+3) \\ &\quad + \ldots | s(k) = s, a(k) = a, \pi\Big] \\ &= R_s^a + \beta \sum_{s'} P_{ss'}^a V^\pi(s') \\ &= R_s^a + \beta \sum_{s'} P_{ss'}^a \sum_{a'} \pi(s',a') Q^\pi(s,a) \end{aligned} \tag{17}$$

Also, equation (16) is a Bellman equation that can be solved by the methods mentioned above.

The optimal action-value function is

$$\begin{aligned} Q^*(s,a) &= \max_\pi Q^\pi(s,a) \\ &= R_s^a + \beta \sum_{s'} P_{ss'}^a \max_{a'} Q^*(s',a') \end{aligned} \tag{18}$$

By applying the policy $\pi$, we can obtain the Q-value represented by $Q^\pi(s,a)$. Therefore, $Q^*(s,a)$ is the Q-value obtained by the optimal policy $\pi^*$. As a result, the optimal policy can be obtained by:

$$\pi^* = \arg_\pi \max Q^*(s,a) \tag{19}$$

**Algorithm 1** The Proposed Algorithm to Select an Optimal Policy

```
Initialize: $\mathcal{S}(k), \mathcal{S}_F(k), \mathcal{A}(k), T, N, M, \alpha, \beta$.
If a new task arrives at IoT node $T_i$ (or a Fog node $F_j$)
    For $k = 0$ to $T$
        Select an action $a_i(k) \in \mathcal{A}_i(k)$ using $\varepsilon - greedy$ policy.
        Receive $\Psi^j(k)$ and $\Theta^j(k)$ from the selected fog node $F^j$.
        Calculate the reward $RWD_j(k)$ based on (12).
        Update $Q(s_i(k), a_i(k))$ based on (20).
        Update the $V(s_i(k))$ using (15).
        Update the $\pi(s_i(k), a_i(k))$ via (19).
    End for
End if
```

The Q-Learning algorithm learns to choose the optimal action in each state depending on the Q-value function. These values determine the best policy for choosing the best fog node to offload a task. Obtaining the values of these functions is the main factor of this algorithm. Whereas, the equation of updating function is defined as follows:

$$Q^{\pi}(s(k), a(k)) = (1-\alpha) Q(s(k), a(k)) + \alpha \left[ RWD_i^k + \beta . \max_{a \in \mathcal{A}} Q(s(k+1), a(k+1)) \right] \quad (20)$$

where, $\alpha$ is the learning rate, $Q(s(k), a(k))$ is the old value, and $\max_{a \in \mathcal{A}} Q(s(k+1), a(k+1))$ is the estimate of optimal future value.

To explain the overall Q-Learning operations for selecting the best choice, we introduce the following algorithm (see Algorithm 1). The algorithm takes as input the state set of IoT nodes $\mathcal{S}(k)$, the state set of fog nodes $\mathcal{S}_F(k)$, and the action set $\mathcal{A}(k)$. We aim to select the optimal action that represents the best fog node or the cloud to offload a task. Therefore, the output of the algorithm gives the best policy.

For each new task generated at one IoT node, a new packet should be sent to the nearby fog nodes to ask about performance metrics. Nearby fog nodes send back the current state for the requested IoT. After that, the IoT node needs to make a decision and inform the selected fog node with the decision. To this end, $|\mathcal{F}^n|+2$ additional control packets are exchanged.

In the same way, when a new task arrives to a fog node $F_j$ from an IoT node or from another fog node, the fog node takes the role of decision maker to offload a task (or some tasks) from the queue. Similarly, the considered fog node run the Q-Learning algorithm, where it takes as input the state set of fog nodes $\mathcal{S}_F$ (k), and the action set $\mathcal{A}$(k).

To describe the overall sequential decision-making process between IoT nodes and fog nodes, we present the following flowchart (see Figure 5). The process starts at an IoT node when a new task is generated. Then, the IoT nodes decide to offload the task. If it does not require a large amount of resources, it may handle the task locally. On the other hand, if the task requires a large amount of resources, the IoT node may offload the task to nearby fog node or to the cloud depending on the latency requirement. When a task is received at a fog node, the fog decides to keep the task in the queue or to offload it to other fog nodes or to the cloud server.

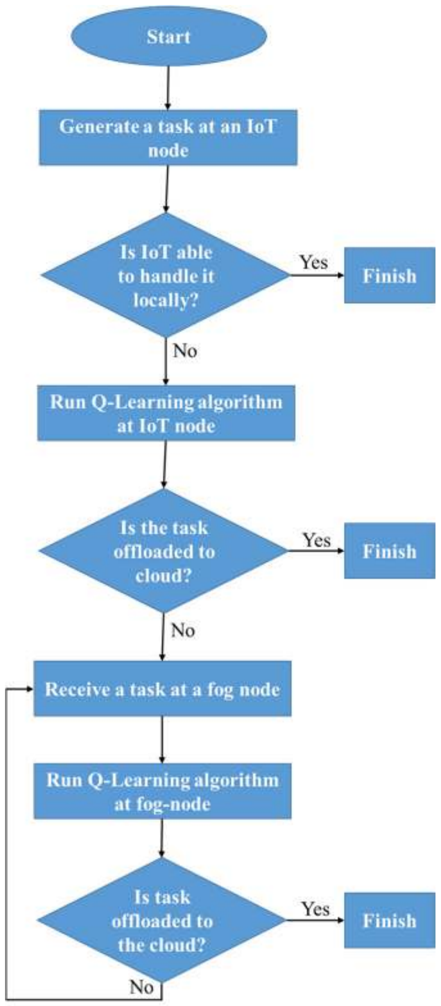
Start
Generate a task at an IoT node
Is IoT able to handle it locally?
Yes
Finish
No
Run Q-Learning algorithm at IoT node
Is the task offloaded to cloud?
Yes
Finish
No
Receive a task at a fog node
Run Q-Learning algorithm at fog-node
Is task offloaded to the cloud?
Yes
Finish
No

**FIGURE 5.** Sequential decision-making process.

## VIII. PERFORMANCE EVALUATION

### A. SIMULATION SETUP

The proposed algorithm is evaluated by computer simulations and compared with other methods that were proposed in this area. MATLAB is used to simulate network connections and offloading scenarios. Many papers used MATLAB to evaluate the performance of offloading methods like [4], and [39]. Similar tools like C++/C# were also used to simulate task offloading methods like [43].

Many topologies are considered in the evaluation process, where the number of IoT nodes changes in the range (50,300), and the number of fog nodes changes between 5 and 30. In the same way, we consider the link speed of wired and wireless links of 100Mbps and 2Mbps, respectively. The wireless communication between the IoT nodes and the nearby fog nodes is assumed to be through IEEE 802.11. The duration of simulation time is set to 3600 seconds. These configurations are used in literature to evaluate task offloading methods [4].

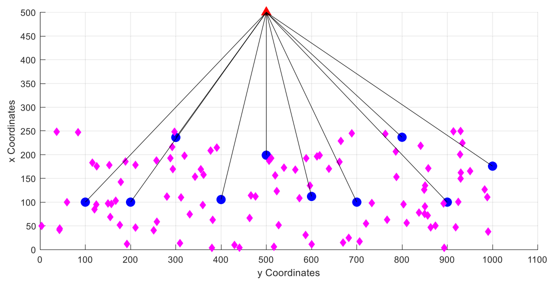

**FIGURE 6.** A snapshot of the considered topology.

Each fog node is provided with 25 CPUs with each has 3.8GHz of speed, and 128 GB of RAM. The tasks generated by the applications running on IoT nodes require different requirements range between 1GHz to 10GHz of processing speed and between 2GB to 10GB of RAM. Each task takes between 1 second to 30 seconds to be executed if it met the minimum requirements at the chosen fog node. The task will take longer time if the assigned resources are lower than the minimum requirements. Each IoT node run many applications, where each application generates its own tasks in random times. The response time of each fog node depends on the tasks that are currently existed in the queue.

In the evaluation scenario, we consider ten fog nodes to serve 100 IoT nodes (Figure 6). The fog nodes are connected with each other by multi-hop connections, while each fog node has a direct connection to the cloud server (shown as lines in Figure 6).

We compared the proposed algorithm named HFCO with F-RAN presented at [4], COG presented at [39], FSYNC [43], and the optimal theoretical results. These algorithms are compared regarding average delay, number of non-served tasks, load balancing, and blocking probabilities. The optimal solution is obtained by centrally solving the problem assuming that we have a perfect prediction of the future generated tasks and the available resources at all fog nodes during an infinite time horizon [40]. In what follows, we briefly explain these works for more understanding.

In [4], the authors introduced a framework to model service delay in IoT-fog-cloud applications. With the aim of delay minimizing, they tried to select the optimal offloading policy, which considered IoT-to-cloud and fog-to-cloud interaction, and fog-to-fog communication. Besides, they considered different kinds of request that have different service times. The proposed framework was not limited to a particular architecture but could be applied to different structures.

An allocation mechanism for fog computing resources in IoT systems is proposed in [39]. The aim is to make the offloading decision for each new task. Each IoT user equipped with multi-RAT is able to offload its tasks to either different fog nodes or to the remote cloud servers. Since each fog node has determined resources, IoT users might compete with each other to offload tasks to good fog nodes. To tackle this problem, the authors first used a processor sharing method to allocate the computing resources of fog nodes and manage their limited resources. They then adopted

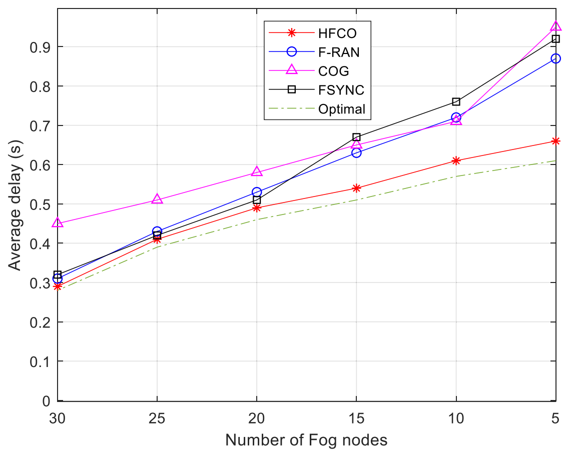

**FIGURE 7.** The average delay in different approaches.

a game theoretic approach to model the competition among IoT users.

An architecture for computation and storage offloading based on fog computing is proposed in [43]. This architecture can help both end users and the cloud. The authors designed a fog-based synchronization algorithm to minimize the communication cost and reduce the latency from the end users to the cloud. In addition, a new scheme with security consideration was provided, where Reed-Solomon code was introduced to protect the privacy of users.

### B. AVERAGE DELAY

The delay includes waiting time in IoT nodes, the time needed to offload the task towards a fog node (or directly toward the cloud) which may execute it, forwards it to another fog node, or towards the cloud. In other words, the delay time is the time interval between generating the task at an IoT user and start executing it. For delay-sensitive applications, the delay time is critical. To evaluate it, we consider 100 IoT nodes, and decrease the number of fog nodes from 30 down to 5 and evaluate the average delay in each method. Figure 7 shows the results of a simulation. When sufficient fog nodes numbers (30) is available, therefore, all methods achieve average delay (about 0.3 sec except COG) close to each other. Because fog nodes are available, and even if the task allocation methods are not optimal, fog nodes are not busy and can execute the task rapidly. However, by decreasing the number of fog nodes, the superiority of our approach becomes clearer, especially with 15 fog nodes and less. In these cases, fog nodes become busier, and our good task allocation approach finds a better choice to where and when to offload the task. On the other hand, the delay in our FHCO is close to the optimal for different numbers of fog nodes.

### C. LOAD BALANCING

Load balancing aims at distributing tasks between fog nodes to minimize response time and to avoid bottleneck situation that may happen in some parts of the network. Load balancing also guarantees that most of the tasks can be processed in fog nodes to avoid offloading them to the cloud, which increases

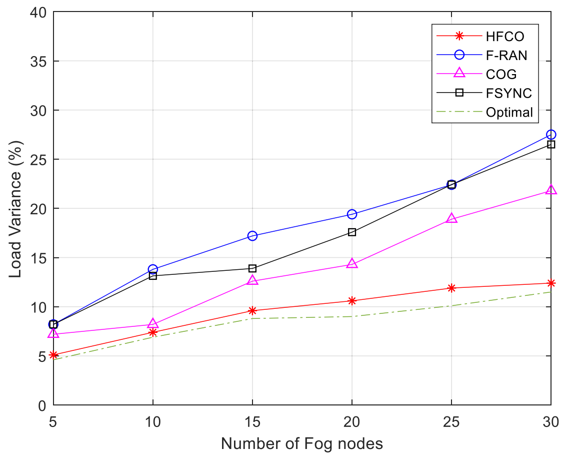

**FIGURE 8.** **Load variance with the number of fog nodes.**

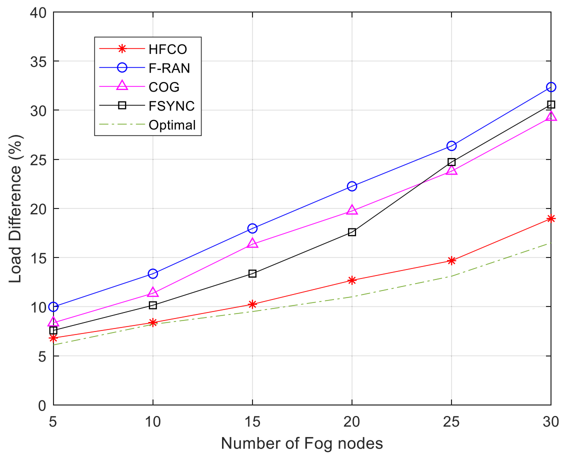

**FIGURE 9.** **Load difference with the number of fog nodes.**

the delay and the overhead. Similar to the previous scenario, we evaluate the proposed approach with other offloading methods considering 100 IoT nodes while increasing the number of fog nodes from 5 to 30. Two criteria are evaluated in terms of load balancing: variance and the load difference between levels of business of fog nodes. Figure 8 shows the load variance in each method. Considering that the load variance reflects the difference in load overall fog nodes, the large value of variance shows that there is less load balancing. From figure 8, we can see that all methods have a good variance when only five fog nodes exist because all the fog nodes are almost busy. By increasing the number of fog nodes, the variance between fog nodes become bigger, and the proposed approach can achieve less load variance and subsequently, better load balancing. Only the optimal solution has some superiority over the proposed HFCO.

Another load-balancing criterion is considered as the difference in load between the fog node with the maximum load and the fog node with minimum load. Figure 9 shows the load difference in different methods. This figure is similar to the previous one showing the difference in load distribution over different fog nodes. However, this one shows larger values since the maximum difference is considered here instead of variance.

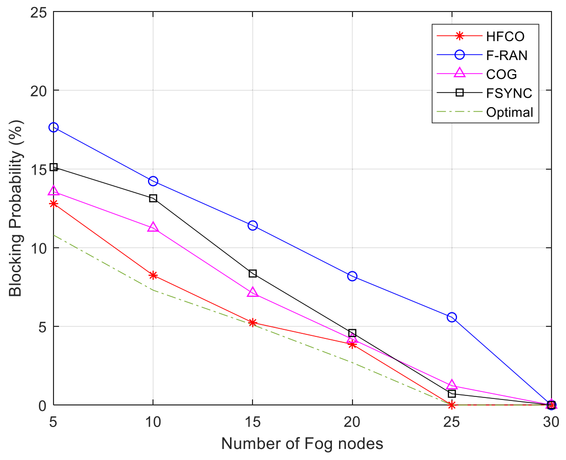

**FIGURE 10.** **Blocking probability with the number of fog nodes.**

## D. BLOCKING PROBABILITY

Blocking probability represents the percentage of tasks that incorrectly completed. This can happen when the task execution time is greater than the maximum allowed time. The maximum allowed time varies from task to task. To calculate this probability, the ratio of tasks that are not successfully completed to all tasks is calculated. As shown in figure 10, increasing the number of fog nodes will reduce the blocking probability because it provides additional resources, that is in turn, serve additional tasks. The blocking probability in the proposed HFCO is close to the optimal solution and is better than other algorithms. Differences between the algorithms decreases with the increase in the number of fog nodes, which means that the proposed algorithm works better in conditions of insufficient resources in the system.

## E. NUMBER OF BENEFICIAL USERS

For IoT users, especially for delay-sensitive applications, it is better to offload tasks towards fog nodes instead of the cloud, which results in larger delays. In this section, we evaluated the number of IoT users that can successfully offload their tasks to the fog nodes. Under some conditions, such that there are not enough resources available at nearby fog nodes or delay requirements, IoT users have to offload tasks to the cloud server. For this evaluation, we considered 300 IoT users and increased the number of fog nodes from 5 to 30. Figure 11 shows that with many fog nodes up to 10, all methods can accept mostly the same number of IoT users to be serviced via fog nodes. However, when the number of fog nodes increases, the difference between methods appears clearer. Figure 11 shows the superiority of the proposed approach over the other ones. This is because our approach distributes the tasks generated by IoT users fairer. On the other side, if tasks are not distributed fairly between fog nodes, some of the fog nodes reach the maximum limit of acceptance. In contrast, others have lower levels of utilization, which, in turn, reduces the number of beneficial IoT users. The proposed HFCO is always close to the optimal solution.

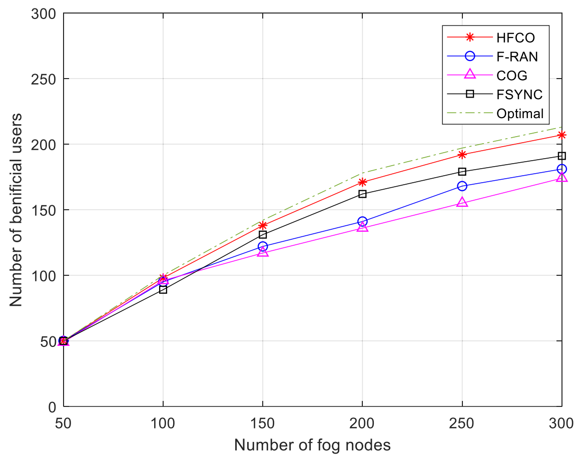

**FIGURE 11. Number of beneficial users with the number of IoT nodes.**

## IX. CONCLUSION

In this paper, a new task offloading approach was proposed to determine the optimal decision on where and when to offload a task; toward a specific fog node or to the cloud server. The problem is studied and analyzed as a Markov decision process. In the proposed MDP, two decision-makers have been considered, where IoT users can decide to which fog node they need to offload their tasks, while fog nodes may decide to offload some tasks to other fog nodes or to the cloud servers to balance the tasks between fog nodes. In order to handle large-size of state space and action space, Q-learning-based algorithm is derived to obtain the optimal policy. Simulation results show that the proposed approach achieves better load balancing and minimizes the delay time compared to other works. Future works in this area can consider the problem with no need to exchange the state of fog nodes using tools like semi-MDP, which reduces the communication overhead and the time needs to make the decision.

## REFERENCES

[1] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," 2015, *arXiv:1502.01815*. [Online]. Available: http://arxiv.org/abs/1502.01815

[2] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 94–100.

[3] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centres in the framework of fog computing," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 4, pp. 1–18, 2016.

[4] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[5] Y.-J. Ku, D.-Y. Lin, C.-F. Lee, P.-J. Hsieh, H.-Y. Wei, C.-T. Chou, and A.-C. Pang, "5G radio access network design with the fog paradigm: Confluence of communications and computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 46–52, Apr. 2017.

[6] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Mar. 2017.

[7] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 3, pp. 107–112, Jan. 2014.

[8] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, Jan. 2018.

[9] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.

[10] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[11] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[12] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[13] G. Lee, W. Saad, and M. Bennis, "An online secretary framework for fog network formation with minimal latency," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[14] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2017, pp. 1–6.

[15] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[16] D. Bruneo, S. Distefano, F. Longo, G. Merlino, A. Puliafito, V. D'Amico, M. Sapienza, and G. Torrisi, "Stack4Things as a fog computing platform for smart city applications," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 848–853.

[17] H. Shah-Mansouri, V. W. S. Wong, and R. Schober, "Joint optimal pricing and task scheduling in mobile cloud computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5218–5232, Aug. 2017.

[18] V. W. S. Wong, *Key Technologies for 5G Wireless Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[19] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[20] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[21] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability," *IEEE Trans. Cognit. Commun. Netw.*, early access, Aug. 20, 2020, doi: 10.1109/TCCN.2020.3018159.

[22] D. Rahbari and M. Nickray, "Task offloading in mobile fog computing by classification and regression tree," *Peer-to-Peer Netw. Appl.*, vol. 13, pp. 104–122, Feb. 2020.

[23] H. Liao, Y. Mu, Z. Zhou, M. Sun, Z. Wang, and C. Pan, "Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing," *IEEE Trans. Intell. Transp. Syst.*, early access, Jul. 21, 2020, doi: 10.1109/TITS.2020.3007770.

[24] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.

[25] S. Maity and S. Mistry, "Partial offloading for fog computing using P2P based file-sharing protocol," in *Progress in Computing, Analytics and Networking*, vol. 1119, H. Das, P. K. Pattnaik, S. S. Rautaray, and K.-C. Li, Eds. Chicago, IL, USA: AISC, 2020, pp. 293–302.

[26] C. Huang, R. Lu, and K.-K.-R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[27] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Gener. Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018.

[28] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K.-R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.

[29] A. Elgazar, K. Harras, M. Aazam, and A. Mtibaa, "Towards intelligent edge storage management: Determining and predicting mobile file popularity," in *Proc. 6th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud)*, Mar. 2018, pp. 23–28.

[30] M. Aazam, E.-N. Huh, and M. St-Hilaire, ''Towards media inter-cloud standardization–evaluating impact of cloud storage heterogeneity,'' *J. Grid Comput.*, vol. 16, no. 3, pp. 425–443, Sep. 2018.

[31] J. Zhou, D. Gao, and D. Zhang, ''Moving vehicle detection for automatic traffic monitoring,'' *IEEE Trans. Veh. Technol.*, vol. 56, no. 1, pp. 51–59, Jan. 2007.

[32] D. Zhao, Y. Dai, and Z. Zhang, ''Computational intelligence in urban traffic signal control: A survey,'' *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 42, no. 4, pp. 485–494, Jul. 2012.

[33] M. Aazam and E.-N. Huh, ''E-HAMC: Leveraging fog computing for emergency alert service,'' in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2015, pp. 518–523.

[34] K. M. Kramer, D. S. Hedin, and D. J. Rolkosky, ''Smartphone based face recognition tool for the blind,'' in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol.*, Aug. 2010, pp. 4538–4541.

[35] S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin, and A. Hassan, ''Vehicular ad hoc networks (VANETS): Status, results, and challenges,'' *Telecommun. Syst.*, vol. 50, no. 4, pp. 217–241, Aug. 2012.

[36] A. P. Jayasumana, Q. Han, and T. H. Illangasekare, ''Virtual sensor networks—A resource efficient approach for concurrent applications,'' in *Proc. 4th Int. Conf. Inf. Technol. (ITNG)*, Apr. 2007, pp. 111–115.

[37] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, ''Wireless sensor network virtualization: A survey,'' *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 553–576, 1st Quart., 2016.

[38] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibanez, ''Internet of vehicles: Architecture, protocols, and security,'' *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3701–3709, Oct. 2018.

[39] H. Shah-Mansouri and V. W. S. Wong, ''Hierarchical fog-cloud computing for IoT systems: A computation offloading game,'' *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[40] M. Ghanem, M. Sabaei, and M. Dehghan, ''A novel model for implicit cooperation between primary users and secondary users in cognitive radio-cooperative communication systems,'' *Int. J. Commun. Syst.*, vol. 31, no. 6, p. e3524, Apr. 2018.

[41] C. Luo, G. Min, F. R. Yu, M. Chen, L. T. Yang, and V. C. M. Leung, ''Energy-efficient distributed relay and power control in cognitive radio cooperative communications,'' *IEEE J. Sel. Areas Commun.*, vol. 31, no. 11, pp. 2442–2452, Nov. 2013.

[42] K. L. Hoffman, M. Padberg, and G. Rinaldi, *Encyclopedia of Operations Research and Management Science*. Norwell, MA, USA: Kluwer, 2001.

[43] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, ''Fog-based computing and storage offloading for data synchronization in IoT,'' *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4272–4282, Jun. 2019.

**SAIF ALJANABI** received the B.S. degree in computer and control system engineering from the University of Technology, Baghdad, Iraq, in 2008, and the M.Sc. degree in information technology from Guru Gobind Singh Indraprastha University, New Delhi, India, in 2011. He is currently pursuing the Ph.D. degree with Razi University, Kermanshah, Iran. His research interests include computer networks and communications.

**ABDOLAH CHALECHALE** received the B.S. degree in electrical engineering and the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 1991 and 1994, respectively, and the Ph.D. degree from Wollongong University, NSW, Australia, in 2005. He is currently with RAZI University, Kermanshah, Iran. His research interests include multimedia processing and communication, artificial intelligence, and human–computer interaction.