Received September 27, 2019, accepted October 20, 2019, date of publication October 30, 2019, date of current version November 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950443

Energy Consumption Optimization With a Delay Threshold in Cloud-Fog Cooperation Computing

GUANGSHUN LI[1,2], JIAHE YAN[1], LU CHEN[3], JUNHUA WU[1], QINGYAN LIN[1], AND YING ZHANG[1]

[1]Department of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China
[2]Department of Computer, The Hong Kong Polytechnic University, Hong Kong
[3]Department of Information Security, Naval University of Engineering, Wuhan 430033, China

Corresponding author: Lu Chen (ieucl@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 61672321, Grant 61771289, and Grant 61832012.

ABSTRACT With the rapid development of the Internet of Things (IoT), the number of mobile terminal devices is increasing. Massive data are generated by mobile terminal devices, resulting in high delay and high energy consumption. In most cases, however, a low delay means high energy consumption. To balance energy consumption and delay, we adopt a tradeoff strategy that can realize optimal energy consumption with a delay threshold in this paper. First, we introduce the role of the delay threshold in reducing delay. Then, we describe the delay and energy consumption of the mobile terminal layer, fog node layer and cloud server layer with queue theory. Nonlinear programming is used to solve the energy optimization problem by calculating the optimal workload of each layer. We design a cloud-fog cooperation scheduling algorithm to reduce energy consumption. A task offloading algorithm is also designed to complete tasks when their nodes leave. The experimental results show that the energy consumption is reduced by approximately 22%, while the delay is 12.5% less than the first come first served (FCFS) approach.

INDEX TERMS Energy consumption optimization, fog computing, tasks scheduling.

I. INTRODUCTION

With the development of the Internet of Things, the number of mobile terminal devices shows a trend of exponential increment [1]. Mobile terminal devices have restrictions in size, battery life and computing capability, which results in poor computing performance [2]. To alleviate the computing burden on mobile terminal devices, the cloud-based structure provides a promising opportunity. Computation-intensive tasks can be transported to the cloud for execution to improve the performance of applications and reduce the energy consumption of mobile terminal devices [3]. The traditional central cloud, such as the Amazon EC2 cloud, Microsoft Windows Azure or Rackspace, has considerable storage, rich computational resources and good service capability, [4], [5]. However, traditional cloud computing is far from the terminal user, which results in degraded service for delay-sensitive applications. Moreover, extraordinarily large volume data generated by mobile terminal devices pose a heavy burden on traditional cloud computing, which results in unbearable transmission delay [6]–[8].

To overcome the above disadvantages, Bonomi created the term ''fog computing'' in 2012, also known as ''cloud at the edge'' [9]. The fog layer acts as a bridge between the mobile terminal layer and the cloud layer, which is composed of fog nodes (such as routers and switches). Each fog node is equipped with the facility of storage, computing, and wireless communication [10]. Fog computing has advantages of low delay, mobility support, and location/context awareness [11], [12], which can meet the needs of applications with strict requirements of interactive response and high resilience. However, fog nodes are limited in size, power, capacity, and only serve a small portion of users [13]. Therefore, fog computing is not a substitution but complements cloud computing [1].

The requirement for cooperation between fog computing and cloud computing is inspired by IoT applications [14]. For example, fog nodes deployed at a highway can provide delay-sensitive services to drivers. These fog nodes send delay-tolerant but computation-intensive services to the cloud for processing [15]. Running too many tasks on fog nodes can

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

reduce delay but may increase the energy consumption of fog nodes. However, transporting tasks to the cloud can save energy of fog nodes but may increase delay. Therefore, it is important to propose a tradeoff strategy between delay and energy [4].

In this paper, we propose a tradeoff strategy between delay and energy with a certain delay threshold. The main contributions are as follows:

1) We model a mathematical framework of a fog-cloud cooperation system with queue theory. We model the energy consumption function and delay function of three system layers and present a joint energy and delay optimization problem.
2) To realize our joint optimization, we provide an easy and universal method for minimizing the upper bound of the delay threshold and use nonlinear programming to calculate the optimal workload of each layer for energy consumption optimization.
3) The cloud-fog cooperation scheduling algorithm is designed to reduce energy consumption when considering new tasks generated by IoT application, and a task offloading algorithm is designed to complete tasks when their nodes leave.
4) The experimental results show that energy consumption is reduced by approximately 22%, while the delay is 12.5% less than the first come first served (FCFS) approach.

The rest of this paper is organized as follows. In Section II, we discuss the related work. We describe the mathematical framework of the fog-cloud cooperation system in Section III. In Section IV, we design a cloud-fog cooperation scheduling algorithm. In Section V, we introduce the task offloading algorithm. The experimental results and conclusions are presented in Section VI and Section VII, respectively.

II. RELATED WORK

The emergence of cloud computing had established a trend of requesting services from cloud center. Singh *et al.* identified opportunities and obstacles in cloud computing [16]. To address the computational burden and transmission delay of cloud computing, a paradigm named fog computing has attracted great attention [17]. In [9], [10], Bonomi *et al.* proposed fog computing and noted its critical roles in real applications. Fog computing is not a substitution but complements cloud computing [1]. Recently, fog computing has been expanded to a series of real applications scenarios, including vehicular networks, smart grids, and smart cities.

In [18], the authors proposed HyFog, a new fog computing hybrid task offloading framework in which devices can choose three options for task execution, including local mobile execution, device-to-device (D2D) offloading execution and cloud offloading execution. However, where to execute the tasks and how to allocate workload are attractive yet challenging topics. In [19], Zhu *et al.* studied a task scheduling problem and aimed to minimize energy consumption in fog computing. A single objective optimization strategy has been adopted in some studies. Ahn *et al.* studied the energy consumption of fog computing and cloud computing, aiming to improve energy efficiency [20]. Li *et al.* [21] proposed a network architecture model that combines cloud computing and fog computing. They solved the delay optimization problem using the Kruskal algorithm and Lagrange multiplier method. Li *et al.* [22] proposed a resource scheduling method to improve the efficiency of resources in fog computing. Li *et al.* [23] proposed a user-oriented spectral clustering scheduling algorithm based on the k-means algorithm to improve the satisfaction of users in a fog computing environment. Yu *et al.* proposed a connected k-coverage working sets construction algorithm (CWSC) based on Euclidean distance to prolong the lifetime of sensor networks [24]. Qi *et al.* proposed a novel cloud service cost optimization method named CS-COM by considering the user's job size, service invocation and service quality level [25]. In [26], Liu *et al.* studied a mutiobjective optimization strategy in fog computing, including delay, energy consumption, and equipment offloading cost. The authors used a numerical conversion algorithm to assign a weight factor to each objective goal and formulated a joint optimization objective to minimize the energy consumption, delay and equipment offloading cost. However, equipment offloading cost optimization may have negative effects on energy consumption optimization or delay optimization. It is important to guarantee energy consumption optimization and delay optimization in cloud and fog computing. In [27], Deng *et al.* studied the tradeoff strategy between delay and energy when allocating tasks in cloud computing. They formulated a workload allocation problem and approached this problem by decomposing the primal problem into three subproblems. Hoang *et al.* proposed the fog-based region and cloud (FBRC) framework [28]. The energy consumption formula was taken as the objective function, and the maximum delay was set as the constraint condition. To find the appropriate upper bound of delay, they took delay as the objective function again, which increased the complexity of the problem. Meng *et al.* minimized energy consumption with a given delay constraint and noted that delay is an uncertain attribute and is related to many factors [29]. Therefore, it is difficult to quantify and express the delay formula.

III. CLOUD-FOG COOPERATION SYSTEM

As shown in Fig. 1, we assume that the cloud-fog cooperation system is composed of N mobile terminal devices, M fog nodes, and K cloud servers. Each mobile terminal device can communicate with others by wireless channels, a fog node provides services for several mobile terminal devices and a cloud server is responsible for several fog nodes. To save communication cost, we assume that the monitoring center responsible for scheduling tasks is in the cloud. Due to the different power and capacity, we consider the traffic model at the mobile terminal device as an M/M/1 queue, the traffic model at the fog node as an M/M/C queue, and the traffic model at the cloud server as an M/M/∞ queue when we model the cloud-fog system with queue theory [30]. If the task arrival

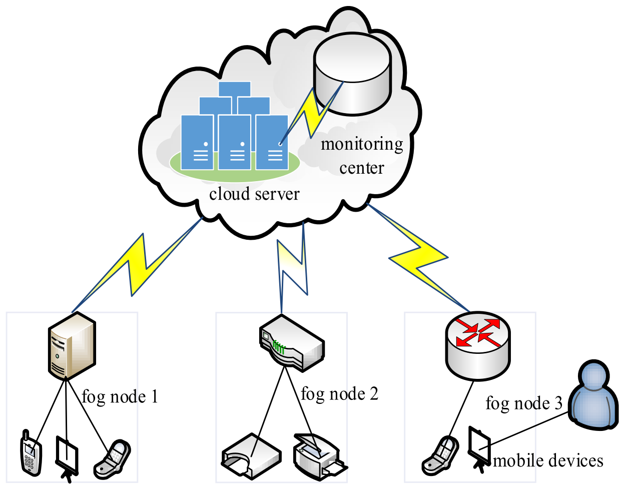


FIGURE 1. The cloud-fog computing architecture.

rate is less than the service rate of the mobile terminal device, then all of the generated tasks are processed at the mobile terminal device. Otherwise, the mobile device offloads tasks to the fog node for execution. If the task arrival rate is more than the service rate of the fog node, the tasks are further transported to the cloud server.

We assume that the tasks follow a Poisson process with an average arrival rate of λ. We can describe a task by some attributes, including the number of tasks t, the task input length I_t, the deadline of the task d_t, the task execution flag u_t, and ψ_t is the computing units required by the task. A task can be described as the following tuple: $\langle I_t, d_t, u_t, \psi_t \rangle$.

In this section, we introduce our cloud-fog cooperation system. First, we introduce the role of the delay threshold. Then, we describe delay and energy consumption with queue theory. Finally, we present the optimization problem and use nonlinear programming to solve it. Some important notations used in this paper are summarized in Table 1.

A. DELAY THRESHOLD DEFINITION

If we require $D_{Total} \leq D_{\max}$, we can minimize $D_{\max}$ to reduce delay. However, the value of $D_{\max}$ will affect task completion and workload allocation. We need to guarantee task completion quality when we minimize $D_{\max}$. The method for minimizing $D_{\max}$ will determine the complexity of the optimization problem. It is important to select a low-complexity and feasible method. The scheduling to minimize lateness (STML) algorithm is designed to minimize maximum delay, which adopts the greedy strategy of the earliest deadline priority and has low complexity. We conducted an experiment to show the role of STML. From Fig. 2, we can see the maximum delay of different scheduling algorithms. The maximum delay of STML is 38, while the max delays of FCFS and shortest job first (SJF) approaches are 49 and 40, respectively. The STML scheduling algorithm can help us find the minimum $D_{\max}$. We discuss task completion and workload allocation in detail in Section VI.

In system initiation, every mobile terminal device sorts the tasks in the queue according to the STML scheduling algorithm and synchronizes its maximum delay with the monitoring center. The monitoring center selects the largest maximum delay as $D_{\max}$ and broadcasts to every mobile terminal device. In this way, we provide an easy and universal method for minimizing the delay threshold.

TABLE 1. Summary and notations.

Symbol	Definition
$E^i_{mt}, E^j_{fog}, E^k_{cloud}$	energy consumption of mobile terminal device i, fog node j, cloud server k (a unit of power)
$D^i_{mt}, D^j_{fog}, D^k_{cloud}$	delay of mobile terminal device i, fog node j, cloud server k (a unit of time)
E_{Total}, D_{Total}	energy consumption, delay of the system
$D_{\max}$	upper bound of delay threshold (a unit of time)
λ	tasks arrival rate #((request)/a unit of time)
μ	tasks service rate #((request)/a unit of time)
$X^i_{mt}, X^j_{fog}, X^k_{cloud}$	workload assigned to mobile terminal device i, fog node j, cloud server k #(request)
$X^{\min}_{mt}, X^{\min}_{fog}, X^{\min}_{cloud}$	minimum workload of mobile terminal device layer, fog node layer, cloud server layer
$X^*_{mt}, X^*_{fog}, X^*_{cloud}$	optimal workload of mobile terminal device layer, fog node layer, cloud server layer

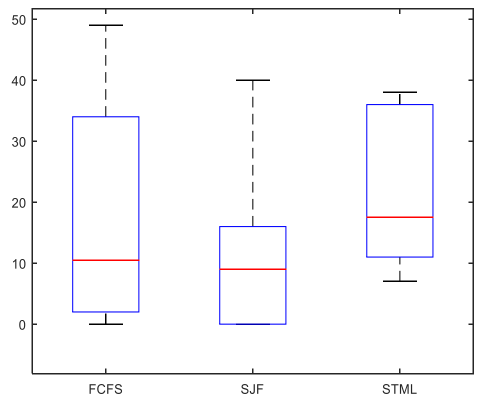


FIGURE 2. Max delay comparison.

B. DELAY DESCRIPTION AND ENERGY DESCRIPTION

1) MOBILE TERMINAL DEVICES

We assume that the service rate μ of a mobile terminal device i follows the exponent process since its task queue is M/M/1. Additionally, the tasks generated from mobile terminal devices follow a Poisson process with an average arrival rate λ. P_{mt} is the power of mobile terminal device i, and T_{mt} is the working time. The energy consumption E^i_{mt} for processing tasks at mobile terminal device i can be described as follows:

$$E^i_{mt} = T_{mt} \times P_{mt} = \frac{X^i_{mt}}{\mu - \lambda} \times P_{mt}. \quad (1)$$

Because tasks executed at mobile terminal devices have a small communication delay, we only consider computing delay. According to queue theory, the delay is expressed as follows:

$$D_{mt}^{i} = \frac{\lambda}{\mu(\mu - \lambda)}. \quad (2)$$

2) FOG NODES

For a fog node j, its task queue is M/M/C. The energy consumption of a fog node can be modeled by a function of the number of computations, which is a monotonic increasing and strictly convex function. The piecewise linear function and quadratic function are two alternatives [1]. For simplicity without loss of generality, we select a quadratic function. The energy consumption of a fog node E_{fog}^{j} is a function related to workload X_{fog}^{j}. We have

$$E_{fog}^{j} = aX_{fog}^{j2} + bX_{fog}^{j} + e. \quad (3)$$

Next, we discuss the delay of fog nodes. The delay of fog nodes is composed of computing delay and communication delay. Assume that the computing delay D_{fog}^{comp} is related to the waiting time. According to queue theory, we have

$$D_{fog}^{comp} = \frac{L_q}{\lambda} \times X_{fog}^{j}, \quad (4)$$

where X_{fog}^{j} is the workload assigned to fog node j, and L_q is the average queue length. As long as the task is not executed by mobile terminal devices, there will be communication delay that is related to the input length of the tasks. We define the transmission delay function as follows:

$$F_{comm}(I_t) = \begin{cases} \gamma I_t & u_t \in cloud \\ \varepsilon I_t & u_t \in fog \end{cases}, \quad (5)$$

where I_t is the input length of task t ($\gamma \gg \varepsilon$). Then, the communication delay of the fog node is $D_{fog}^{comm} = \varepsilon I_t$.The delay of the fog node is composed of computing delay and communication delay, which can be expressed as:

$$D_{fog}^{j} = D_{fog}^{comp} + D_{fog}^{comm}. \quad (6)$$

3) CLOUD SERVERS

For a cloud server k, its task queue is M/M/∞. When the allocated workload increases, more cloud servers power on. When the workload decreases, some servers turn off for energy savings. The energy consumption of cloud server E_{cloud}^{k} is related to the on/off state, the on-state machine number, and the workload, which is as follows.

$$E_{cloud}^{k} = \sigma_k n_k (a_k X_{cloud}^{k} + b_k), \quad (7)$$

where a_k and b_k are positive constants. σ_k denotes the on/off state of the cloud server k, where 1 indicates that the cloud server is on, and 0 represents off. And n_k is the number of on-state machines on the cloud server. Because cloud servers have rich computational resources, the computing delay can be ignored, so the delay is mainly the communication delay. Refer to (5), we have

$$D_{cloud}^{k} = \gamma I_i. \quad (8)$$

4) PROBLEM OPTIMIZATION

The total energy consumption of the system E_{Total} is:

$$E_{Total} = \sum_{i \in N} E_{mt}^{i} + \sum_{j \in M} E_{fog}^{j} + \sum_{k \in K} E_{cloud}^{k}, \quad (9)$$

where $0 < i \le N, 0 < j \le M, 0 < k \le K$. The total delay of the system can be expressed as

$$D_{Total} = \sum_{i \in N} D_{mt}^{i} + \sum_{j \in M} D_{fog}^{j} + \sum_{k \in K} D_{cloud}^{k}. \quad (10)$$

We can realize optimal energy consumption with a certain delay threshold by solving nonlinear programming as follows:

$$\begin{aligned} &\min \ E_{Total} \\ &s.t. \ D_{Total} \le D_{\max} \\ &\quad \sum_{i \in N} X_{mt}^{i} + \sum_{j \in M} X_{fog}^{j} + \sum_{k \in K} X_{cloud}^{k} = C \\ &\quad X_{mt}^{\min} < X_{mt}^{i} < C \\ &\quad X_{fog}^{\min} < X_{fog}^{j} < C \\ &\quad X_{cloud}^{\min} < X_{cloud}^{k} < C, \end{aligned} \quad (11)$$

where C is the total workload. It may reduce delay; however, it may increase energy consumption when tasks are executed on fog nodes. In contrast, offloading tasks to the cloud server can save the energy of fog nodes but increase the transmission delay. To balance the workload, we set a minimum workload for each layer. $X_{mt}^{\min}$ is the minimum workload of the mobile terminal device layer, $X_{fog}^{\min}$ is the minimum workload of the fog node layer, and $X_{cloud}^{\min}$ is the minimum workload of the cloud server layer. By solving the above nonlinear programming problem, we can obtain the optimal workload of each layer. X_{mt}^{*} is the optimal workload of the mobile terminal device layer, X_{fog}^{*} is the optimal workload of the fog node layer, and X_{cloud}^{*} is the optimal workload of the cloud server layer.

Fig. 3 is the task processing chart in our cloud-fog cooperation system. After tasks arrive, tasks are allocated to each layer by using our tradeoff. However, numerous new tasks are often generated in IoT applications. In this scenario, we propose a fog-cloud cooperation scheduling algorithm to reduce energy consumption when processing new tasks (Section IV). Moreover, the IoT system is dynamic, and one node may leave the system when there are unfinished tasks in its queue. In this case, we propose the tasks offloading algorithm to complete tasks when their nodes leave (Section V). The monitoring center in the cloud is responsible for the scheduling tasks and monitoring devices conditions.

IV. CLOUD-FOG COOPERATION SCHEDULING ALGORITHM

In IoT application scenarios, some latency-sensitive tasks must be responded to immediately. The tasks can be executed by the allocated layer when the workloads are not exceeded. However, the tasks cannot be executed directly when the layer

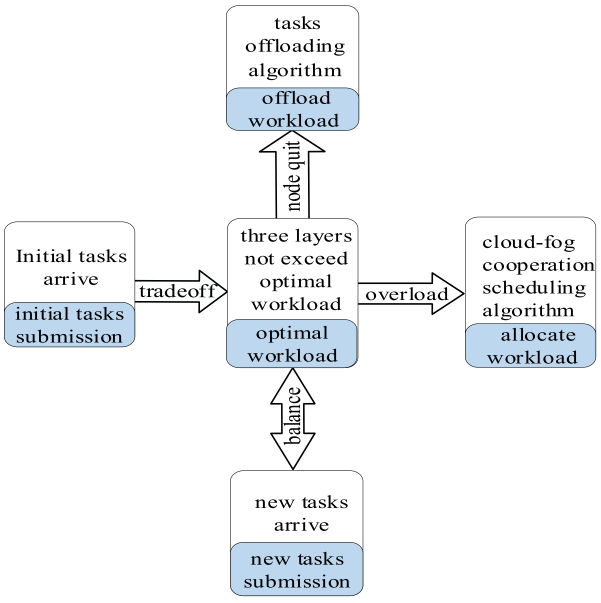


FIGURE 3. **The task processing chart.**

workloads exceed. Therefore, we design a cloud-fog cooperation scheduling algorithm to reduce energy consumption. The approach is summarized in Algorithm 1.

Although the cloud server has rich computational resources, we cannot allocate all new tasks to the cloud server. Where to execute a new task is related to the energy consumption and delay tolerance of the new task. In the cloud-fog cooperation scheduling algorithm, u_t is the task execution flag; when the flag value is 1, the task is executed on the mobile terminal device layer, a value of 2 indicates

TABLE 2. **Algorithm 1.**

Cloud-Fog Cooperation Scheduling Algorithm

```
Sort n tasks by deadline so that $d_1 \le d_2 \le ...... \le d_n$
while $(u_t = 0)$
{
   if  $\psi_t < \psi^a_{mt}$
      $u_t = X_{mt} + \psi_t \le X^*_{mt}\ ?\ 1:01$
   else if  $\psi^a_{mt} < \psi_t < \psi^a_{fog}$
      $u_t = X_{fog} + \psi_t \le X^*_{fog}\ ?\ 2:02$
   else  $\psi_t > \psi^a_{cloud}$  or  $\psi_t > \psi^a_{fog}$
      $u_t = 3$
   switch$(u_t)$
   {
      case 01: $u_t = X_{fog} + \psi_t < X^*_{fog}\ ?\ 2:3$
      case 02: $u_t = 3$
   }
   update $E_{Total}$
   return $u_t$
}
```

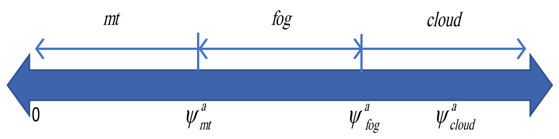


FIGURE 4. **Range of task execution.**

that the task is executed on the fog node layer, and a value of 3 indicates that the task is executed on the cloud server layer. If the task t is not allocated, the value of u_t is 0. Energy consumption is related to computing units required by the task. ψ^a_{mt}, ψ^a_{fog} and ψ^a_{cloud} represent the average computing units required by the task performed at mobile terminal devices, fog nodes and cloud servers, respectively. ψ_t denotes the computing units required by task t. If ψ_t is less than ψ^a_{mt}, we allocate the task to the mobile terminal device layer for execution. It must be determined whether the current workload plus task t exceeds the optimal workload before it is allocated. The task can be allocated to the mobile terminal device layer if the total workload of the mobile terminal devices does not exceed the optimal workload; otherwise, set u_t to 01 and wait for further processing. If ψ_t is more than ψ^a_{mt} and less than ψ^a_{fog}, the task is transported to the fog nodes for execution. We determine whether the current workload plus task t exceeds the optimal workload before it is allocated. The task can be allocated to the fog layer if the total workload of the fog nodes does not exceed the optimal workload; otherwise, set u_t to 02 and wait for further processing. If ψ_t is more than ψ^a_{fog} or ψ^a_{cloud}, the task must be assigned to the cloud server for execution. Next, in the case of $u_t = 01$, the task can be allocated to the fog layer if the total workload of the fog nodes does not exceed the optimal workload; otherwise, the task should be allocated to the cloud layer. In the case of $u_t = 02$, the task should be allocated to the cloud layer. Finally, the value of u_t is returned.

Theorem 1: Scheduling tasks according to the cloud-fog cooperation scheduling algorithm can not only minimize the maximum delay but also meet the requirements of energy consumption.

Prove: We sort tasks from small to large with the STML algorithm and obtain the minimal value of $D_{\max}$. We next prove that the cloud-fog cooperation scheduling algorithm can make the energy consumption of the system meet the requirements of the objective optimization. From Fig. 4, for a new task, if the computing units required by the task are in $(0, \psi^a_{mt}]$, we allocate the task to the mobile terminal device layer. If the computing units required by the task are in $\left(\psi^a_{mt}, \psi^a_{fog}\right]$, we allocate the task to the fog node layer. If the computing units required by the task are in $\left(\psi^a_{fog}\ , \psi_t\right]$, we allocate the task to the cloud server layer.

E^a_{mt}, E^a_{fog}, E^a_{cloud} represent the average energy consumption of previous tasks performed at mobile terminal devices, fog nodes and cloud servers respectively. When the power of the device is constant, the energy consumption is proportional to the computing units required by the task. The mobile terminal device layer energy consumption distribution is in $(0, E_{mt}^{a}]$, the fog node layer energy consumption distribution is in $(E_{mt}^{a}, E_{fog}^{a}]$, and the cloud server layer energy consumption distribution is in $(E_{fog}^{a}, \varpi]$, ϖ is the upper limit of the cloud server energy consumption. We can conclude that the energy consumption of a new task is not more than the average energy consumption at the mobile terminal device layer and the fog node layer. The energy consumption of a new task may be more than the average energy consumption at the cloud layer. For a large task, if its energy consumption is more than $E_{mt}^{a}, E_{fog}^{a}, E_{cloud}^{a}$, we should allocate the task to the cloud server layer since the energy consumption of a large task is closest to E_{cloud}^{a}.

V. TASK OFFLOADING ALGORITHM

As is known, the IoT network is dynamic, and one node could leave the system when it runs out of power. If there is an unfinished task in the node's queue, the task will be offloaded to other nodes for execution. Therefore, we designed a task offloading algorithm to ensure that the task could be completed when their nodes leave. The task offloading algorithm is given as Algorithm 2.

TABLE 3. Algorithm 2.

```
Task Offloading Algorithm

if ($u_t \in \{mt\}$){
  $n_1 = mtnode\_set()$
  for($i = 1; i \le n_1; i++$)
   {if $C_{idle}^{i} > \psi_t$
      $u_t = 1$
    else
      $u_t = X_{fog} + \psi_t \le X_{fog}^{*}\ ?\ 2:3$}
}
if ($u_t \in \{fog\}$){
  $n_2 = fognode\_set()$
  for($j = 1; j \le n_2; j++$)
   {if $C_{idle}^{j} > \psi_t$
      $u_t = 2$
    else
      $u_t = 3$}
}
Return $u_t$
```

Assume there are n_1 mobile terminal devices and n_2 fog nodes. If task t needing to be offloaded belongs to the mobile terminal device layer, first, we determine whether the mobile terminal device layer has idle computing resources to complete task t. δ_i is the current CPU utilization rate and C_i is the total computing cell of the mobile terminal device i. The idle computing resource of the mobile terminal device i is $C_{idle}^{i} = (1 - \delta_i)C_i$. When the mobile terminal device i has enough idle computing resources to complete task t, we allocate the task to mobile terminal device i for execution. Otherwise, we consider allocating task t to the fog node layer or cloud server layer for execution. We should allocate task t to the fog node layer when the current number of tasks in the fog node layer plus task t does not exceed the optimal workload. Otherwise, task t is allocated to the cloud server layer. In the same way, if task t belongs to the fog node layer, we first determine whether fog node j can accept this task. If yes, we allocate the task t to fog node j. Otherwise, the task is allocated to the cloud server. The task offloading algorithm is used to ensure that tasks can still be executed when the execution node exits.

Theorem 2: The time complexity of this task offloading algorithm is $O(n)$.

Prove: Assuming there is a task t to be offloaded, it could be at the mobile terminal device layer or fog node layer. Due to rich computational resources, the cloud server layer does not offload to the mobile terminal device layer and fog node layer. Our choice is to offload the task to the layer that the task belongs to. Therefore, in the worst case, the algorithm only needs to traverse the mobile terminal device layer and fog node layer, and the time complexity is $O(n)$.

TABLE 4. Value of important parameters.

Parameter	Value	Parameter	Value
μ	4.6	a	1
λ	3.2	b	2.4
P_{mt}	23	e	3.5
$X_{mt}^{\min}$	[3 5 10 20 30]	a_k	[3.206 4.485 2.370]
$X_{fog}^{\min}$	[10 20 30 50 80]	b_k	$[68\ 53\ 70]\times10^{-2}$

VI. SIMULATION

Simulation results are presented in this section to validate the effectiveness of our method. For simplicity but without loss of generality, we consider the scenario with seven mobile terminal devices, three fog nodes and one cloud server in the fog-cloud computing system. It can be extended to more mobile terminal devices, fog nodes, and cloud servers, with similar results. Some important parameters used in the simulation are summarized in Table 4, referring to [1], [4] and [30]. To reflect the performance of our method at different workloads, we selected five groups of tasks, and their total workloads are 30, 50, 90, 150, and 200, in which the minimum workload of the mobile terminal device layer and fog layer is 3, 5, 10, 20, and 30 and 10, 20, 30, 50, and 80, respectively. More or fewer tasks can be achieved similarly. The lengths of the tasks are generated randomly because we cannot predict the length of a task in reality. We conduct extensive simulations using MATLAB 2015a on a computer equipped with a Pentium Dual-Core CPU running the Windows operating system to validate the performance of our proposed method.

First, we investigate the impact of the scheduling method on the upper bound of the delay threshold $D_{\max}$. Because

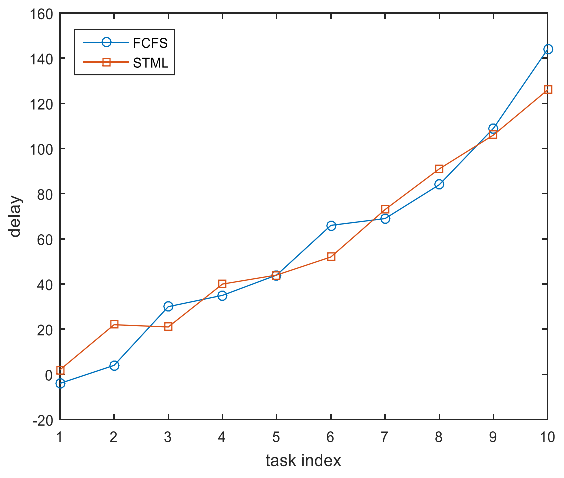


FIGURE 5. Delay of FCFS and STML.

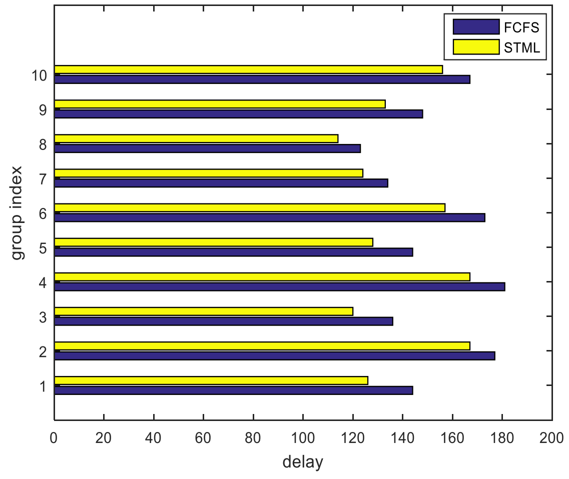


FIGURE 6. Max delay of FCFS and STML.

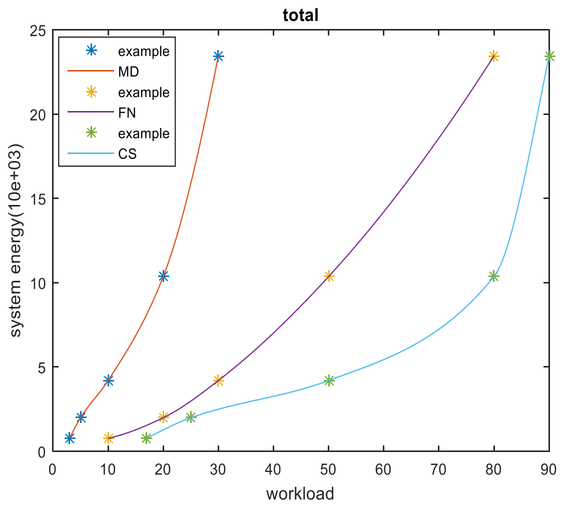


FIGURE 7. The energy consumption of the system without considering the delay threshold.

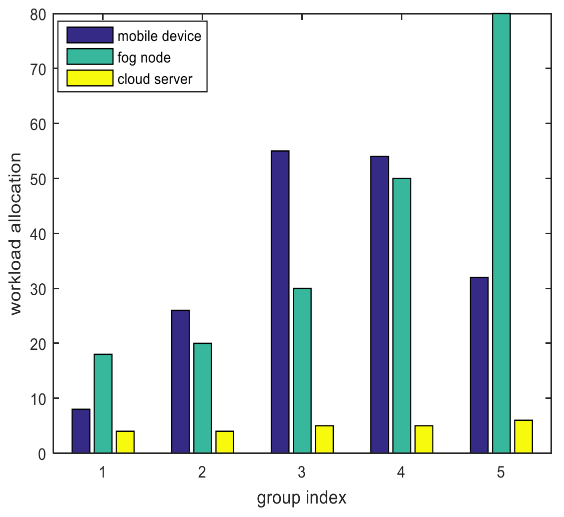


FIGURE 8. Workload allocation of $D_{\max} = 50$.

FCFS is most commonly used in fog or cloud computing, we compare STML with FCFS. As Fig. 5 shows, a group of tasks are scheduled by FCFS and STML. The x-coordinate represents the sign of the tasks, and the y-coordinate represents the delay. As shown in Fig. 5, the maximum delay of STML is approximately 126, and the maximum delay of FCFS is approximately 144, which means that the upper bound of delay threshold $D_{\max}$ can be reduced from 144 to 126. In other words, STML minimizes the global maximum delay by approximately 12.5%. Moreover, we divide 100 tasks into 10 groups and schedule them by FCFS and STML. The maximum delay of each group is presented in Fig. 6. We can see that the reduction in the max delay is stable.

Then, we study how the delay threshold affects our method. Fig. 7 shows the energy consumption of the system without considering the delay threshold. We select five groups as examples, in which the total workload is 30, 50, 9, 150, and 200. The sample points at the same horizontal line represent the workload allocated to each layer when the system energy consumption is constant. We find that the workload allocated to the mobile terminal device layer is 3, 5, 10, 20, and 30 (indicated by blue stars), the workload allocated to the fog node layer is 10, 20, 30, 50, and 80 (indicated by yellow stars), and the workload allocated to the cloud server layer is 17, 25, 50, 80, and 90 (indicated by green stars), respectively. With the increase in total workload, the workload growth trend of mobile terminal devices and fog nodes does not change much, and it increases linearly, while the slope of the workload of the cloud server fluctuates. Moreover, the workload of the mobile terminal device layer and fog node layer is equal to the minimum workload we set, and the cloud server performs the highest workload. As is known, processing tasks on the cloud can save the energy of mobile terminal devices and fog nodes, but it unavoidably increases delay. In this case, we do not consider delay, so the cloud processes most of the workload. Our three-layers model meets the characteristics of the system.

To reflect how the delay threshold affects workload allocation and energy consumption, we choose two scenarios with small and normal delay thresholds. The scenario of a large delay threshold is equivalent to without considering delay,

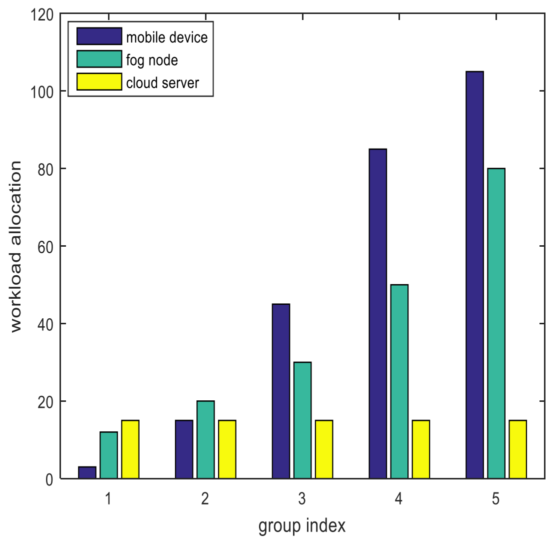


FIGURE 9. Workload allocation of $D_{\max} = 150$.

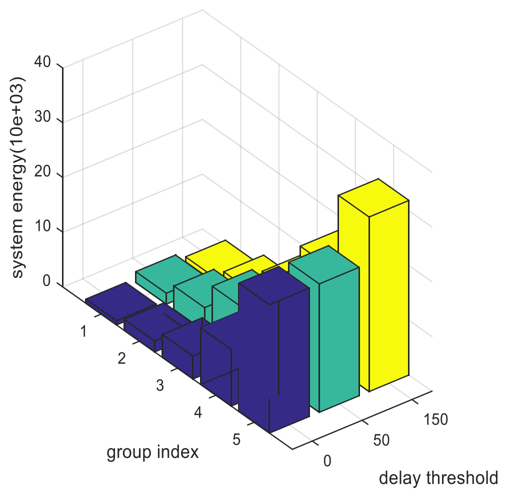


FIGURE 10. Energy comparison at different $D_{\max}$.

which we have discussed above. As Fig. 6 shows, the average max delay of STML is approximately 150, so the normal delay threshold is $(-\infty, 150]$. We set the small delay threshold as $(-\infty, 50]$, which is one-third of the normal delay threshold. Fig. 8 is the workload allocation when $D_{\max} = 50$. In Fig. 8, when the total workload is 30, 50, 90, 150, and 200, the workload allocated to the mobile terminal device layer is 8, 26, 55, 54, and 32, the workload allocated to the fog node layer is 18, 20, 30, 50, and 80, and the workload allocated to the cloud server layer is 4, 4, 5, 5, and 6, respectively. We can see that the mobile terminal device layer and fog layer perform most of the workload. While the workload allocated to the cloud changes little, the workload allocated to mobile terminal devices rises and then falls, and the workload allocated to fog nodes rises because a small delay threshold has higher requirements for task delay, which results in processing tasks at the local layer. It is worth noting that the sum of workload allocated to each layer is not the total workload when the total workload is 150 and 200, which means there

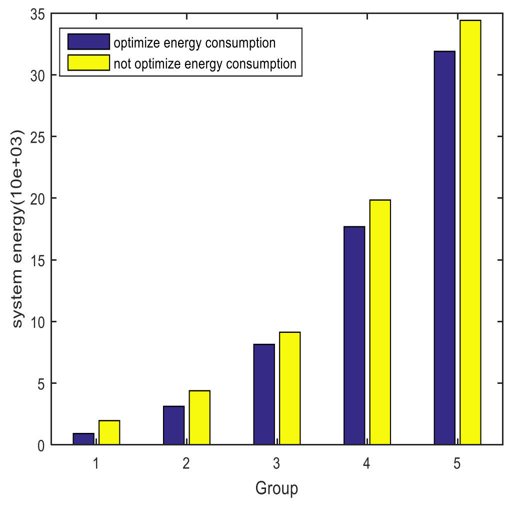


FIGURE 11. Energy consumption comparison of $D_{\max} = 150$.

are tasks not completed. This is mainly because of the power constraint. The power of mobile terminal devices and fog nodes is far from the cloud, and they cannot complete too many tasks without assistance from the cloud. Fig. 9 is the workload allocation when $D_{\max} = 150$. In Fig.9, when the total workload is 30, 50, 90, 150, and 200, the workload allocated to the mobile terminal device layer is 3, 15, 45, 85, and 105, the workload allocated to the fog node layer is 12, 20, 30, 50, and 80, and the workload allocated to the cloud server layer is 15, 15, 15, 15, and 15, respectively. We can conclude that the workload of the cloud layer is increased approximately three times to alleviate the energy consumption of the mobile terminal device layer and fog layer. It is more important that all tasks are finished in this scenario. From above, we can clearly observe the necessity of determining a proper delay threshold for task allocation and completion.

In addition, energy consumption at different $D_{\max}$ value is shown in Fig. 10. We can conclude that the higher the delay requirement, the larger the energy consumption. The energy consumption of the situation where $D_{\max} = 50$ and the total workload is 150 and 200 should be higher than others, but there are many unfinished tasks, so we do not see this in Fig. 10. Therefore, the normal delay threshold we set is a proper delay threshold.

Finally, we compare our energy consumption optimization with the nonoptimization of energy consumption. In Fig. 11, an obvious energy consumption reduction with a normal delay threshold can be realized using our method. When the total workload is 30, 50, 90, 150, and 200, the energy consumption reduction rate is approximately 53%, 28%, 10%, 10%, and 7.3%, respectively. Then, the average energy consumption reduction rate is approximately 22%. Therefore, our energy consumption optimization is effective. However, the energy consumption reduction rate decreases when the total workload increases; $D_{\max} = 150$ can ensure that all tasks finish on time. The delay-intensive requirement is reached by processing at mobile terminal devices or fog nodes, which increases the energy consumption of the system when total the workload increases.

VII. CONCLUSION

In this paper, we adopt a tradeoff strategy that can realize optimal energy consumption with a delay threshold. We model a three-layers fog-cloud cooperation system by describing energy and delay function with queue theory. We solve the energy optimization problem by nonlinear programming, and solve the delay optimization problem by STML approach. We designed two detailed algorithms to reduce new task's energy consumption and guarantee task completion. The experimental results show that energy consumption is reduced by approximately 22%, while the delay is 12.5 less than FCFS approach. For future work, we intend to further optimize the energy consumption and delay in cloud-fog computing where fog nodes are heterogeneous. In that case, the energy consumption description and delay description of fog nodes need to be carefully investigated.

REFERENCES

[1] R. Deng, R. Lu, T. H. Luan, H. Liang, and C. Lai, "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.

[2] S. Wan, Y. Zhao, T. Wang, Z. Gu, Q. H. Abbasi, and K.-K. R. Choo, "Multi-dimensional data indexing and range query processing via Voronoi diagram for Internet of Things," *Future Gener. Comput. Syst.*, vol. 91, pp. 382–391, Feb. 2019.

[3] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang, T. Umer, and S. Wan, "An edge computing-enabled computation offloading method with privacy preservation for Internet of connected vehicles," *Future Gener. Comput. Syst.*, vol. 96, pp. 89–100, Jul. 2019.

[4] L. Liu, Z. Chang, S. Mao, T. Ristaniemi, and X. Guo, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[5] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, "Fog-computing-enabled cognitive network function virtualization for an information-centric future Internet," *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 48–54, Jul. 2019.

[6] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Trans. Netw. Sci. Eng.*, to be published.

[7] X. Lin, J. Li, J. Wu, H. Liang, and W. Yang, "Making knowledge tradable in edge-AI enabled IoT: A consortium blockchain-based efficient and incentive approach," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2019.2917307.

[8] H. Liang, J. Wu, S. Mumtaz, J. Li, X. Lin, and M. Wen, "MBID: Micro-blockchain-based geographical dynamic intrusion detection for V2X," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 77–83, Oct. 2019.

[9] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. ACM 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[10] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Cham, Switzerland, Springer, 2014, pp. 37–42.

[11] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Trans. Ind. Informat.*, to be published.

[12] Z. Guan, Y. Zhang, L. Zhu, L. Wu, and S. Yu, "EFFECT: An efficient flexible privacy-preserving data aggregation scheme with authentication in smart grid," *Sci. China-Inf. Sci.*, vol. 62, no. 3, pp. 1–14, Mar. 2019.

[13] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[14] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[15] S. K. Datta, C. Bonnet, and J. Haerri, "Fog computing architecture to enable consumer centric Internet of Things services," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Madrid, Spain, Jun. 2015, pp. 1–2.

[16] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, Feb. 2014.

[17] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Warsaw, Poland, 2014, pp. 1–8.

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

[19] T. Zhu, T. Shi, Z. Cai, X. Zhou, and J. Li, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, Jun. 2019.

[20] S. Ahn, M. Gorlatova, and M. Chiang, "Leveraging fog and cloud computing for efficient computational offloading," in *Proc. IEEE MIT Undergraduate Res. Technol. Conf. (URTC)*, Cambridge, MA, USA, Nov. 2017, pp. 1–4.

[21] G. Li, J. Wang, J. Wu, and J. Song, "Data processing delay optimization in mobile edge computing," *Wireless Commun. Mobile Comput.*, vol. 2018, Feb. 2018, Art. no. 6897523.

[22] G. Li, Y. Liu, J. Wu, D. Lin, and S. Zhao, "Methods of resource scheduling based on optimized fuzzy clustering in fog computing," *Sensors*, vol. 19, no. 9, p. 2122, 2019.

[23] G. Li, S. Xu, J. Wu, and H. Ding, "Resource scheduling based on improved spectral clustering algorithm in edge computing," *Sci. Program.*, vol. 2018, Jul. 2018, Art. no. 6860359.

[24] J. Yu, X. Deng, G. Wang, X. Gu, and D. Yu, "CWSC: Connected k-coverage working sets construction algorithm in wireless sensor networks," *AEU-Int. J. Electron. Commun.*, vol. 67, no. 11, pp. 937–946, 2013.

[25] L. Qi, J. Yu, and Z. Zhou, "An invocation cost optimization method for Web services in cloud environment," *Sci. Program.*, vol. 2017, May 2017, Art. no. 4358536.

[26] L. Liu, Z. Chang, T. Ristaniemi, and X. Guo, "Multi-objective optimization for computation offloading in mobile-edge computing," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2017, pp. 832–837.

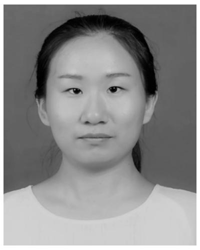
[27] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3909–3914.

[28] D. Hoang and T. D. Dang, "FBRC: Optimization of task scheduling in fog-based region and cloud," in *Proc. IEEE Trustcom /BigDataSE/ICESS*, Sydney, NSW, Australia, Aug. 2017, pp. 1109–1114.

[29] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, 2017.

[30] T. Mori, Y. Utsunomiya, T. Okuda, and X. Tian, "Queueing theoretic approach to job assignment strategy considering various inter-arrival of job in fog computin," in *Proc. IEEE 19th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2017, pp. 151–156.

GUANGSHUN LI received the Ph.D. degree from Harbin Engineering University, China, in 2008. He has been a Visiting Scholar with Hong Kong Polytechnic University, since 2019. He is currently an Associate Professor with the School of Information Science and Engineering, Qufu Normal University, China. He has already published more than 50 articles. His research interests include wireless networks, the IoT, and Bigdata.

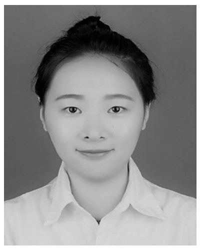


JIAHE YAN received the bachelor's degree in computer science and technology from Qufu Normal University, China, in 2017, where she is currently pursuing the M.Sc. degree. Her current research interests include mobile edge computing, task scheduling, and the IoT.

LU CHEN received the Ph.D. degree from Wuhan University, China. She is currently an Associate Professor with the Department of Information Security, Naval University of Engineering, China. Her research interest includes trusted computing.

JUNHUA WU received the Ph.D. degree from Harbin Engineering University, China, in 2009. She is currently an Associate Professor with the School of Information Science and Engineering, Qufu Normal University, China. She has already published more than 40 articles. Her research interests include wireless networks, edge computing, and the IoT.

QINGYAN LIN received the bachelor's degree in computer science and technology from Qufu Normal University, China, in 2017, where she is currently pursuing the M.Sc. degree. Her current research interests include mobile edge computing, task scheduling, and resource allocation.



YING ZHANG received the bachelor's degree in computer science and technology from Qufu Normal University, China, in 2017, where she is currently pursuing the M.Sc. degree. Her current research interests include the Internet of Vehicles, edge computing, stackelberg game, wireless resource allocation, heterogeneous computing systems, game theory, and mobile computing.