Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4218696, 9 pages
https://doi.org/10.1155/2022/4218696

Research Article

Fragmented Task Scheduling for Load-Balanced Fog Computing Based on Q-Learning

Mian Muaz Razaq, Shahnila Rahim, Byungchul Tak, and Limei Peng

School of Computer Science and Engineering, Kyungpook National University, Daegu 41566, Republic of Korea

Correspondence should be addressed to Limei Peng; auroraplm@knu.ac.kr

Received 28 December 2021; Accepted 26 January 2022; Published 10 March 2022

Academic Editor: Xiaojie Wang

5G and beyond (B5G) applications generate tremendous computing-intensive, latency-sensitive, and privacy-sensitive tasks, which differ from the legacy cloud computing tasks, requiring more sophisticated scheduling strategies. We must satisfy the stringent service requirements, particularly privacy preservation that has not been sufficiently considered in the past. Meanwhile, we need to balance the tasks offloaded to different edge nodes to avoid overwhelming some fog nodes, which may degrade the overall performance. To appropriately schedule the privacy-sensitive tasks while balancing the traffic load, we define IoT tasks according to their security need, processing time, and real-time requirement and segment IoT tasks into smaller pieces based on their privacy levels. The sliced tasks are scheduled to multiple fog nodes with satisfactory security reputations to avoid a compromised fog node handling a whole task. Meanwhile, we consider the constraint of the response time of all available fog nodes before scheduling IoT tasks to avoid chaos task scheduling that may overwhelm some fog nodes. Regarding this, we propose a reinforcement learning (RL) model in which the agent tends to satisfy the required latency and security requirements while avoiding overloading some fog nodes to minimize the average delay. The numerical results demonstrate that the proposed approach performs well in a better-balanced load and less performance violation in latency and security.

1. Introduction

With the mature of 5G technologies that support pervasive Internet connections anywhere at any time, the number of Internet users will reach 5.3 billion, almost 71 percent of the population, in 2023 [1]. Most of them use wireless devices, such as smartphones, tablets, smartwatches, home meters, and wearable devices, proliferating numerous mobile IoT data that require real-time and secure services. These services are pretty different from the legacy cloud data services, thus soliciting the fog nodes at the edge rich in computing, storage, and bandwidth resources, to satisfy the emerging service demands of IoT data [2, 3].

Efficiently offloading IoT tasks to fog nodes while satisfying the QoS requirement with minimized fog resources is of primary concern. Motivated by the fact that attacking cloud servers results in significant data leakage, some literature has considered fragmenting IoT tasks to increase privacy measures. For instance, authors in [4] sliced an IoT task into three pieces with different sizes and schedule them to the cloud, fog, and local machine, respectively. Each of the sliced pieces lacks a part of crucial information, thus increasing the privacy of overall data even if a portion of data is being compromised. To avoid data leakages from location-based services (LBS) and keep the actual data trajectory, dummy trajectories were sent along with encrypted data to LBS for processing and getting results at fog nodes [5]. A decentralized blockchain-based technique is proposed in [6] utilizing the edge nodes to assist the burdens of cluster heads formed, aiming to minimize the system latency while maximizing the data safety of the user.

Another crucial issue is to avoid overwhelming some fog nodes with numerous IoT tasks while leaving some other fog nodes underutilized, which would degrade the overall performance of fog networks. Local fog managers and SDN controllers have been utilized to balance the load both locally

and globally by splitting each fog node into several levels, with each level indicating a load threshold at that fog node. Fog managers monitor the load levels of fog nodes belonging to their respective clusters and distribute the tasks accordingly [7]. Some works have considered offloading the task to the nearest available fog node for processing first and then to the cloud if the fog node is overloaded [8].

To satisfy the privacy requirement while balancing the load of fog nodes, this paper fragments each IoT task according to their security level and offloads each sliced task piece to the fog node with the least response time and satisfactory security reputation. For this purpose, we categorize the fog nodes and IoT tasks into three different security levels, i.e., low, medium, and high. The processing capacity of fog nodes and processing requirement of IoT tasks are categorized in a similar way. A fog node with a higher processing capacity can process IoT tasks in less time than the one with low processing capacity.

To achieve the above ideas, this paper proposes to deploy a reinforcement learning agent at one of the edge servers in the fog layer. Before scheduling an IoT task to any fog node or cloud, the agent slices the task into several fragments based on the security requirement and size of the IoT task. Only one sliced piece can be scheduled to the same fog node. If the agent schedules an IoT task piece to a fog node with a lower security level than the fragment requires, it is considered a security violation, and the agent will receive a substantial negative penalty for such action. Moreover, the agent also monitors the required response time to serve the task piece by all the available fog nodes at that time step and selects the node with a lesser response time, thus avoiding overloading only the nearest fog nodes. Regarding this, we propose a Q-learning-based algorithm, which guides the agent to schedule the incoming IoT task pieces to fog nodes or cloud, considering the security, real-time, and bandwidth requirements while avoiding overwhelming some fog nodes.

The remainder of this paper is organized as follows. Section 2 introduces the related works. Section 3 presents the preliminaries, including the system model and description, the reinforcement learning environment, and the proposed Q-learning-based algorithm. Section 4 defines the simulation environment and discusses the simulation results. Finally, Section 5 concludes the work.

2. Related Works

Minimizing the delay while scheduling IoT tasks to fog nodes has been the research focus of many existing works. Chiti et al. in [9] proposed a distributed algorithm to optimally select fog nodes based on matching theory, aiming to minimize the maximum total task completion time to achieve efficient task offloading for real-time applications with considerations on communications and computational costs. Tran-Dang et al. proposed the adaptive resource-aware task offloading scheme to minimize the average delay by selecting an optimal policy to determine the most appropriate fog node with available resources for task offloading. Yousefpour et al. reduced the service delay when offloading the tasks to fog nodes by considering a generalized model with flexible topology [10]. Wang et al. have proposed an imitation learning-based approach to provide the most updated information to the vehicular network by minimizing the Age-of-Critical-Information (AoCI) taking advantage of the edge nodes [11].

Mobile fog computing using unmanned aerial vehicles (UAVs) as fog nodes has been widely discussed in the past few years. In [12], Ning et al. considered offloading tasks to UAV-based edge servers that are deployed to minimize the total cost. Authors in [13] considered grouping the IoT users to different clusters and providing computing services through UAVs serving as mobile edge nodes. In [14], Wang et al. proposed an imitation learning-based approach to deploy UAVs as mobile edge nodes and addressed the issue of deploying UAVs belonging to different service providers in a shared area such that the profits of the owners along with the utilities of users are maximized.

Some literature also addressed the energy efficiency issue while offloading the tasks to fog nodes. Zhang et al. proposed an energy-minimizing algorithm that selects a fog node for task offloading, considering the energy consumption, history about the average energy usage of fog nodes, and priorities of fog nodes [15]. Yang et al. maximized the energy efficiency of fog networks consisting of fog nodes with different computing resources by proposing the maximal energy-efficient task scheduling algorithm [16]. Jalali et al. compared the nanodata centers (nDCs) used in fog computing with the centralized cloud data centers. They pointed out that energy efficiency is impacted more by the application types, the type of access networks attached to the nanoservers, and the active and idle time of nanoservers, rather than the number of hops [17].

3. System Description and System Model

3.1. System Description. A three-layer integrated fog and cloud framework is considered as shown in Figure 1. The bottom IoT layer consists of IoT devices such as tablets and smartphones, which generate IoT tasks with different requirements in terms of their nature, such as real time or not, security and privacy preservation level, and bandwidth-intensity.

The middle fog layer comprises fog nodes with different capabilities, such as different security credits, computing and storage capability, and mobility. Specifically, their security reputation and processing capability can be categorized into high, medium, and low. The processing ability of a fog node affects the task execution time when a task goes from the task waiting queue to the processing stage. On the other hand, the reinforcement learning (RL) agent is deployed in the edge node, which is responsible for scheduling and offloading IoT tasks to other fog nodes or the cloud for processing.

The top cloud layer is assumed to have sufficient computing and storage resources to serve all the IoT tasks but cannot satisfy the real-time requirement due to its long distance from the IoT users. Serving an IoT task with the real-

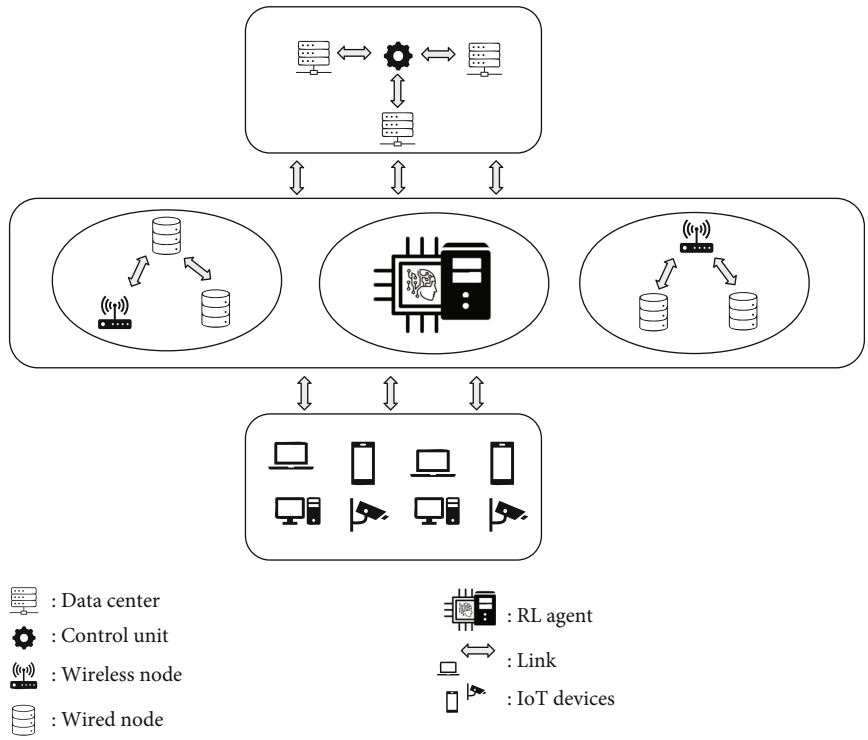

Figure 1: Three layered integrated cloud fog framework.

time requirement by the distant cloud is deemed as a delay violation.

Fog nodes can serve IoT tasks with real-time requirements and security levels less than the security reputation of the fog nodes. We slice a single IoT task into smaller pieces based on their security requirement to help in privacy preservation. A fog node with an equal or higher security reputation than the security level of an IoT task piece can provide service. Moreover, a single fog node is not allowed to serve more than one piece of a sliced IoT task. IoT task pieces can be offloaded to either the cloud or fog nodes flexibly only if their requirements can be satisfied. The objective of task offloading is to minimize the average end-to-end (E2E) delay, avoiding overwhelming some fog nodes to balance the traffic load among fog nodes. We treat the average response time of the whole iCloudFog as the vital metric for evaluating the network load balance. The reason behind this is that if there are some overloaded fog or cloud nodes, their response time may become very considerably large, leading to a high average response time.

The response time is determined by the E2E delay, consisting of the processing time, queuing delay, propagation time, and transmission time when serving an IoT task. Therefore, we always consider the E2E delay before scheduling any task and avoid offloading IoT task segments to nodes requiring long response time, which in turn helps balance the overall traffic load. Based on the above assumptions, this paper tackles the problem of IoT task scheduling by proposing an intelligent solution based on Q-learning that meets the requirements of security, real-time, etc., of IoT tasks while minimizing the average E2E delay of iCloudFog when offloading IoT tasks for load balancing.

3.2. System Model. Table 1 shows the notations of sets, parameters, variables, and their descriptions used for describing the system. Cloud with sufficient computing, storage, and network resources is represented by C and can process all but the real-time tasks. We use the sets of $F = \{F_1, F_2, \cdots, F_j\}$ and $T = \{T_1, T_2, \cdots, T_i\}$ to denote the fog nodes and IoT tasks, respectively, where j and i are integer indexes.

Each fog node F_j in the set F is a four tuple, i.e., $F_j = \{\mathrm{FR}_j, \mathrm{FS}_j, \mathrm{FP}_j, \mathrm{RP}_j\}$, where FR_j, FS_j, FP_j, and RP_j are the available computing/storage resources, security level, processing power, and response time of the fog node j. Specifically, the value of 0, 1, or 2 for FS_j means fog node j has low, medium, or high security reputation; the value of 0, 1, or 2 of FP_j indicates low, medium, or high processing capability of fog node j.

Similarly, each IoT task element in the set T is a three tuple characterized by quality-of-service (QoS) requirements, i.e., $T_i = \{\mathrm{SL}_i, \mathrm{RT}_i, \mathrm{RC}_i\}$, where SL_i, RT_i, and RC_i denote the requirements of security level, real-time service, and required computing/storage resources. Specifically, the value of 0, 1, or 2 for SL_i indicates the low-, medium-, and high security requirement; RT_i is one when task i has the real-time requirement and zero vice versa. The value of 0, 1, or 2 for RC_i indicates task i is low, medium, or high computing-intensive. DE_i denotes the maximum delay requirement of task$i$.

4. RL Model for Load Balancing and Privacy-Aware Task Offloading

The objective is to minimize the average delay of task offloading to balance the traffic load of fog nodes and meanwhile satisfy various QoS requirements, especially the

Table 1: Sets, parameters, and variables.

C	Cloud computing server
F	Set of fog nodes
T	Set of IoT tasks
$S(k)$	State space of system at time slot k
$A(k)$	Action space of system at time slot k
R	Possible states of available resources
RT_i	Binary variable. One indicates IoT task i is real-time task; zero, vice versa; $i \in T$
dv_{ij}	Binary variable. One indicates there is a delay violation while serving task i at fog node j; $i \in T\, j \in F$
sv_{ij}	Binary variable. One indicates there is a security violation while serving task i at fog node j; $i \in T\, j \in F$
TR_i	Required resources of IoT task i; $i \in T$
SL_i	Security requirement of IoT task i; $i \in T$
RC_i	Computing required of IoT task i; $i \in T$
TF_i	Number of fragments of IoT task i; $i \in T$
DE_i	Maximum delay threshold of IoT task i; $i \in T$
FR_j	Available resources of fog node j; $j \in F$
FS_j	Security level of fog node j; $j \in F$
FP_j	Processing power of fog node j; $j \in F$
RP_{ij}	Response time of fog node j for IoT task i; $i \in T\, j \in F$
RP_c	Response time of cloud
RR_j	Reward based on response time of fog node j; $j \in F$
RS_j	Reward based on security adherence j; $j \in F$
Td_{ij}	Transmission delay; $i \in T\, j \in F$
Pd_{ij}	Propagation delay; $i \in T\, j \in F$
Pr_{ij}	Processing delay; $i \in T\, j \in F$
Ts_i	Task size; $i \in T$
BW_{ij}	Link bandwidth; $i \in T\, j \in F$
Ds_{ij}	Distance between fog node j and source of IoT task i; $i \in T\, j \in F$
c	Speed of light
λ_j	Arrival rate of tasks at fog node j; $j \in F$
μ_j	Service rate of tasks at fog node j; $j \in F$
RWD_s^a	Reward received by agent after taking an action a in state s; $a \in A(k), s \in S(k)$
RWD_j^k	Cumulative future reward; $j \in F$
F_j^*	Optimal fog node j; $j \in F$
α	Learning rate
γ	Discount factor

privacy requirement. We assume that a fog node with a high traffic load suffers from a long response time to handle new tasks, and hence, the response time of a fog node determines the probability of it being selected to provision a new IoT task. The problem is formulated as a Markov decision process (MDP) model which can be solved by the reinforcement learning- (RL-) based algorithm proposed in this paper. The proposed RL MDP model is represented by a quadruple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{R}$ define a finite set of states, a finite set of actions, a transition probability matrix for every action in each state, and the reward associated with every state [18], respectively.

4.1. State. The state of fog node j at time slot k is determined based on its available resources FR_j, security level FS_j, response time FP_j, and the response time of cloud RP_c. Based on the assumption that the cloud always has sufficient

resources to serve tasks, the cloud resources do not affect the system state and thus is not considered here. Nonetheless, the response time of the cloud is vital for task scheduling and contributes to determining the system state. The state of fog node j consisting of a couple of substates at time slot k is represented as follows:

$$\Delta^j(k) = \left(\mathrm{FR}_j, \mathrm{FS}_j, \mathrm{RP}_j\right). \tag{1}$$

The system state space is comprised of the states of all fog nodes and the response time of the cloud, which is represented by Equation (2), where n is the total number of fog nodes.

$$S(k) = \left[\Delta^1(k), \Delta^2(k), \Delta^3(k) \cdots \Delta^n(k), \mathrm{RP}_c\right]. \tag{2}$$

Assume the state of available resource of fog node j changes from r to r' in time slot k and S_R is the set of possible states of the available resources; the transition probability of the available resource state, i.e., $P^j_{r,r'}(k)$, is defined as follows:

$$P^j_{r,r'}(k) = \Pr\left[\mathrm{FR}_j(k+1) = r' \mid \mathrm{FR}_j(k) = r\right] \quad \forall r, r' \in S_R. \tag{3}$$

4.2. Action. The system action space consists of all the actions that the agent can take at a particular state. The action for IoT task T_i in time slot k can be chosen from a combination of available fog nodes and cloud C while avoiding selecting the same fog node. This is based on the assumption that an IoT task can be sliced into pieces and each node cannot provision more than one of the pieces. The number of the combination based on the fog nodes and cloud is defined as follows:

$$N = \begin{pmatrix} n+1 \\ \mathrm{TF}_i \end{pmatrix} = \frac{n+1!}{\mathrm{TF}_i!((n+1) - \mathrm{TF}_i)!}, \tag{4}$$

where $n+1$ means the total n available fog nodes plus the cloud. TF_i is the number of fragments that IoT task i has been sliced. Hence, the action space can be represented as follows:

$$A_i(k) = [a_1(k), a_2(k), a_3(k), \cdots, a_N(k)], \tag{5}$$

where N is the number of combination of fog nodes and cloud obtained from Equation (4).

4.3. State Transition. When the agent takes an action $a \in A_i(k)$ in state S of the system at time slot k, the system will change its state from s to s', and the transition probability matrix of the system is defined as follows:

$$P^a_{s,s'} = \Pr\left[s(k+1) = s' \mid s(k) = s, a(k) = a\right]. \tag{6}$$

4.4. Reward. When the reinforcement learning (RL) agent explores the environment, it learns from the reward fed back by the environment when it takes some action from the action set in a particular state, resulting in the system's transition to the next state [19]. The reward of our agent on each time step k is determined by the response time of fog node j and its security level deciding how well it can serve an IoT task with security requirement, denoted by $\mathrm{RWD}^a_s(k)$ as follows:

$$\mathrm{RWD}^{\mathrm{a}}_{\mathrm{s}}(k) = \mathrm{RR}_j(k) + \mathrm{RS}_j(k), \tag{7}$$

$$\mathrm{RS}_j(k) = c_2 - c_3 * \mathrm{sv}_{ij}, \tag{8}$$

$$\mathrm{RR}_j(k) = -\mathrm{RP}_j(k) - c_1 * \mathrm{dv}_{ij}, \tag{9}$$

where $\mathrm{RS}_j(k)$ is the negative or positive reward received by the agent based on the security level of fog node j and security requirement of IoT task i; c_1, c_2, and c_3 are constant values where $c_3 > c_1 > c_2$; dv_{ij} and sv_{ij} are binary parameters, where $\mathrm{dv}_{ij} = 1$ indicates a delay violation, meaning the delay requirement of IoT task (fragment) i cannot be satisfied by fog node j, and $\mathrm{sv}_{ij} = 1$ indicates a security violation, meaning the security credit of fog node j is lower than IoT task (fragment) i requires and vice versa; $\mathrm{RR}_j(k)$ is the negative or positive reward that the agent gets based on the response time of fog node j; and $\mathrm{RP}_j(k)$ therein is the response time of fog node j for provisioning IoT task i at time slot k.

$$\mathrm{RP}_j(k) = \mathrm{Td}_{ij} + \mathrm{Pd}_{ij} + \mathrm{Pr}_{ij}, \tag{10}$$

where Td_{ij}, Pd_{ij}, and Pr_{ij} are the transmission delay, propagation delay, and processing delay, respectively, when scheduling task i to fog node j, as defined in Equations (11), (12), and (13). Transmission delay Td_{ij} is calculated by dividing task size Ts_{ij} over the link bandwidth BW_{ij}; propagation delay Pd_{ij} is calculated by dividing the distance between fog node j and the source device of task i, i.e., Ds_{ij}, over the light speed c; processing delay Pr_{ij} considers the task arrival rate λ_j and service rate μ_j at fog node j.

$$\mathrm{Td}_{ij} = \frac{\mathrm{Ts}_i}{\mathrm{BW}_{ij}}, \tag{11}$$

$$\mathrm{Pd}_{ij} = \frac{\mathrm{Ds}_{ij}}{c}, \tag{12}$$

$$\mathrm{Pr}_{ij} = \frac{\lambda_j}{2\mu_j}\left(\mu_j - \lambda_j\right). \tag{13}$$

Based on Equations (7), (8), and (9), the agent will receive a positive reward c_2 if it schedules the task to a fog node that satisfies the task security requirement. Nonetheless, if it schedules the task to a fog node with lower security level than the task requirement, causing a security violation, i.e., $\mathrm{sv}_{ij} = 1$, the agent will be penalized with a large negative reward c_3.

In addition, the agent will receive a positive reward equal to the response time of fog node j, i.e., RP_j, if there is no delay violation when scheduling IoT task i. In case of delay violation, i.e., $\mathrm{dv}_{ij} = 1$, the agent will then select the fog node

Input: $S(k), A(k), R, T, F, C$
Output: Delay violations, security violations, optimal Q-table values and policy
1 **while** All IoT tasks $i \in T$ are scheduled **do**
2 Obtain TF_i by slicing the IoT task i based on security requirement TS_i and task size Ts_i;
3 Use ε – greedy policy Equation (16) to select action $a_i \in A(k)$;
4 Get response time RP_{ij} of selected fog node j using (10).
5 Get $\mathrm{FR}_j, \mathrm{FS}_j, \mathrm{FP}_j$ of selected fog node j.;
6 Calculate reward RWD_j^k using (14)
7 Use (17) to update the Q-table;
8 Update the available resource state R, system states $S(k)$, time step k
9 **end while**

Algorithm 1: Q-Learning Based Load Balanced Task Scheduling.

Table 2: Parameter configuration.

Parameter	Value
Episodes	10000
ε	0.5
α	0.4
γ	0.6
DE_i	50-500 ms
BW_{ij}	1 Gbps
Ds_{ij}	5-1000 m
FR_j	75-85 units
TR_i	5-9 units

with the least response time at that particular time step k, avoiding scheduling the task to a busy fog node with heavy load and thus balancing the system load. The considerable negative penalty for delay violation forces the agent not to schedule task i to a fog or cloud node whose response time exceeds the delay requirement of IoT task i.

To avoid greedy decisions when achieving the load-balanced environment, the long-term reward is considered rather than short-term rewards by considering the expected cumulative future discounted reward for fog node j at time step k, i.e., RWD_j^k, computed as follows:

$$\mathrm{RWD}_j^k = E_\lambda \left[\sum_{t=0}^{T} \beta^t \mathrm{RWD}_s^a (k+t+1) \right], \tag{14}$$

where T is the future time slot when the reward is determined and β is the discounting factor. Therefore, the agents try to select those actions that can maximize the sum of rewards it will receive [19]. Specifically, based on Equation (14), the agent determines the discounted cumulative future reward for all the fog nodes in its vicinity and activates the fog node with the highest reward. The optimal fog node can be defined as follows:

$$F_j^* = \arg \max_{F_j \in F_i^n} \left\{ \mathrm{RWD}_j^k \right\}. \tag{15}$$

4.5. Q-Learning Policy. We use the most used RL method, i.e., Q-learning, to obtain the policy. The goal is to enable the agent to learn from the environment and select the best action under a particular state to maximize its rewards. The policy $\pi(s)$ dictates the agent which action to select at some particular state. To keep a balance between exploration and exploitation, we use ε – greedy policy [19] in our model as follows:

$$\pi(s) = \begin{cases} a \in \mathrm{argMax} Q_i(s, a_i), & \text{probability of } 1-\varepsilon, \\ a \text{ random action } \mathrm{a}_i, & \text{otherwise}, \end{cases} \tag{16}$$

where ε is the probability of taking a random action in an ε – greedy policy. Exploration allows the agent to explore the environment by trying multiple actions and learning the best results for those actions. After learning next time, i.e., $k+1$, it exploits those values to schedule optimally. In Q-learning algorithm, the Bellman equation [19] is used to update the state-action paired values of the Q-table:

$$\begin{aligned} Q^\pi(s(k), a(k)) &= (1-\alpha) Q(s(k), a(k)) \\ &\quad + \alpha \Big(\mathrm{RWD}_j^k + \gamma \max Q(s(k+1), a(k+1)), \end{aligned} \tag{17}$$

where α is the learning rate that determines how much our agent cares about the previously learnt information. If $\alpha = 1$, the agent will override the most recent Q value rather than learning from past values. A too-small α will result in slow learning; $\alpha = 0$ will restrict the agent from updating the old values. γ is the discount factor, defining the nature of the agent regarding long-term or short-term rewards while making decisions.

4.6. Q-Learning-Based Algorithm for Scheduling. This section introduces the proposed Q-learning-based algorithm as shown in Algorithm 1 in the pseudocode. It takes as input the system state space $S(k)$, action space $A(k)$, states of available resources R, set of IoT tasks T, set of fog nodes F, and

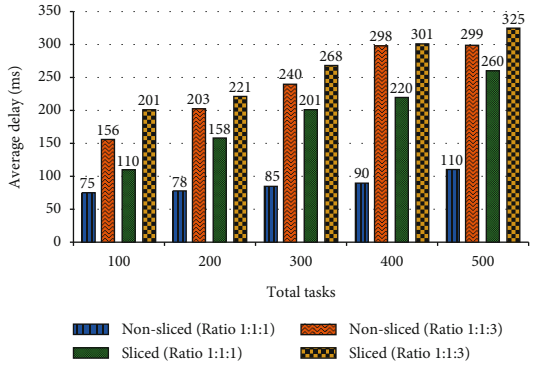

Figure 2: Average delays of sliced vs. nonsliced tasks with different ratios of security requirements.

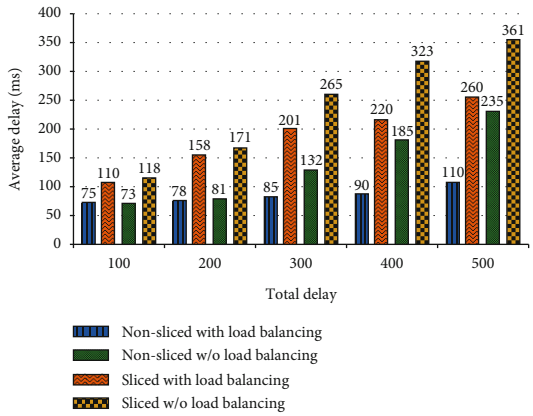

Figure 3: Average delays of sliced vs. nonsliced tasks with and without load balancing approach.

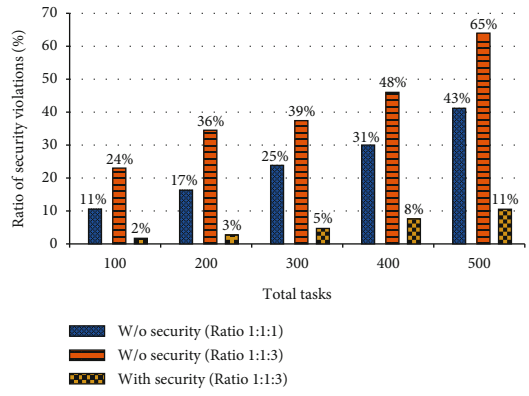

Figure 4: Security violations with and without security constraints for different ratios of security requirements.

cloud C. The objective of the algorithm is to select an optimal action for the agent so that the best fog node or cloud for offloading the IoT task is picked. Regarding this, the output of our algorithm is the optimal policy and Q-table values.

For IoT task i, the agent first converts the task into TF_i number of fragments based on the security requirements and task size as shown in line 2. The agent then takes an

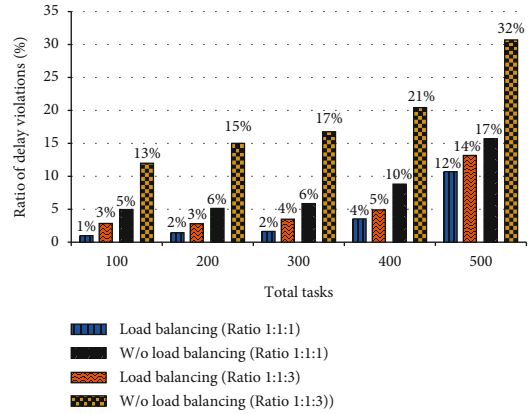

Figure 5: Delay violations with and without load balancing for different ratios of security requirement.

action by selecting a combination of the fog nodes or cloud to serve all the fragments of IoT task from the available fog nodes and cloud server. The agent receives the response time, available resources, security level, and processing power of each selected fog node and then calculates the reward based on the response time and security adherence the selected fog node can provide to the IoT task. Then, the agent updates the Q-table values, available resource state, and system state.

5. Performance Evaluation

5.1. Simulation Environment. We used a Jupyter Notebook on an Intel Core i7 system with 16-GB RAM and an Nvidia GTX-1650 dedicated graphics card to evaluate the Q-learning algorithm. Table 2 presents the parameters and their values used for simulation. Assume nine fog nodes and one cloud server in the environment. The number of tasks varies from 100 to 500. The maximum delay threshold DE_i for IoT tasks is set in a range of 50 ms to 500 ms. Link bandwidth BW_{ij} of the link from IoT task i source to fog nodes j is assumed to be 1 Gbps, while the link between fog nodes and cloud is assumed to be 4 Gbps [20]. The sizes of IoT task Ts_i are distributed as 10% ranging from 500 Mb to 1 Gb, 20% ranging from 500 Kb to 1 Mb, and 70% ranging from 1 to 50 Mb. The distance Ds_{ij} between IoT task source devices and fog nodes is assumed to be within 1000 m. The resource units of fog nodes FR_j range from 75 to 85, while the resource units required by the IoT tasks range from 5 to 9.

6. Results and Analysis

Figure 2 evaluates the system load balancing in terms of the average delay under different numbers of IoT tasks, i.e., 100 to 500, for schemes with and without IoT task slicing based on their security requirements. We consider the ratio of IoT tasks with low, medium, and high security levels as 1 : 1 : 1 and 1 : 1 : 3, respectively. It can be noticed that the average delay of the proposed approach is larger as compared to the approach that does not consider the IoT task slicing.

The reason is that the task, when sliced, should be scheduled to different fog nodes, suffering from more complexity due to satisfying the delay and security requirement of each IoT task piece and thus leading to longer overall average delay. Nonetheless, the delay is still within the required range of general real-time tasks. Moreover, the delay for the tasks with a higher ratio (i.e., schemes of 1 : 1 : 3) of high security requiring tasks is larger than the one with equal distribution of low, medium, and high security requiring tasks (i.e., schemes of 1 : 1 : 1). This is due to security constraints in place, since the agent must schedule the high security requiring tasks to the fog nodes with high security levels only, which are lesser in number. Hence, the tasks must be queued first at those fog nodes, which add to the average delay.

Figure 3 compares the average end-to-end delay for schemes with and without IoT task slicing and load balancing considerations. It is evident that the task offloading schemes taking load balancing into consideration, i.e., nonsliced with load balancing and sliced with load balancing, have a lesser average delay than those without a load balancing approach, i.e., nonsliced w/o load balancing and sliced w/o load balancing. The difference is not that much for a small number of tasks, but as the number of tasks increases from 300 to 500, the difference becomes more considerable.

Figure 4 shows the performance of security violations for schemes with and without security considerations. The result figure shows that if not consider the security requirement, the agent will end up scheduling higher or medium security requiring tasks to fog nodes with lesser security credit, reflected by high ratio of security violations with increasing IoT task numbers, which threatens the task privacy. Moreover, the security violations increase when the ratio of high security requiring tasks increases from 1 : 1 : 1 (33%) to 1 : 1 : 3 (60%). On the other hand, our proposed model performs much better with even 60% high security requiring tasks.

Figure 5 shows the ratio of delay violations for schemes with and without load balancing considerations for IoT tasks with low to medium to high security ratios of 1 : 1 : 1 and 1 : 1 : 3, respectively. For both ratios, it is evident that our proposed model with security constraints and load balancing approach performs better with fewer delay violations than the approach that does not consider load balancing.

7. Conclusion

This paper proposed an IoT task offloading algorithm based on Q-learning that forces the agent to schedule the incoming IoT tasks to fog nodes with lightweight traffic loads and satisfactory security credit; while in task scheduling, the agent fragmentedanIoT task first based on its security requirement and size and then scheduled the task fragments to fog nodes with equal or higher security credits than the task fragments' security level. During this process, the agent tried to balance the load of the network by considering the response time of fog nodes for each particular IoT task, which reduced the overall average delay while ensuring security constraints.

Data Availability

The data of our paper is generated in our own lab based on the requirements of our simulation scenarios and we have not used any public data. Nonetheless, we declare that once our paper is accepted, we will share our data by uploading it to the submission portal as a supportive file or anywhere required by the WCMC journal.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (Ministry of Education) (Grant No. 2020R1I1A3072688).

References

[1] T. Cisco and A. Internet, "Cisco: 2020 CISO benchmark report," *Computer Fraud Security*, vol. 2020, no. 3, p. 4, 2020.

[2] G. Peralta, M. Iglesias-Urkia, M. Barcelo, R. Gomez, A. Moran, and J. Bilbao, "Fog computing based efficient IoT scheme for the Industry 4.0," in *Proceedings of the 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics, ECMSM 2017*, pp. 1–6, Donostia, Spain, 2017.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC'12- Proceedings of the 1st ACM Mobile Cloud Computing Workshop*, pp. 13–15, Helsinki, Finland, 2012.

[4] T. Wang, J. Zhou, X. Chen, G. Wang, A. Liu, and Y. Liu, "A three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 3–12, 2018.

[5] J. Z. TianWang, M. Z. A. Bhuiyan, H. Tian, Y. Cai, Y. Chen, and B. Zhong, "Trajectory privacy preservation based on a fog structure for cloud location services," *IEEE Access*, vol. 5, pp. 7692–7701, 2017.

[6] Z. Ning, S. Sun, X. Wang et al., "Blockchain-enabled intelligent transportation systems: a distributed crowdsensing framework," *IEEE Transactions on Mobile Computing*, vol. 1233, pp. 1–17, 2021.

[7] A. J. Kadhim and J. I. Naser, "Proactive load balancing mechanism for fog computing supported by parked vehicles in IoV-SDN," *China Communications*, vol. 18, no. 2, pp. 271–289, 2021.

[8] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *International Journal of Information Technology and Computer Science*, vol. 8, no. 4, pp. 1–10, 2016.

[9] F. Chiti, R. Fantacci, and B. Picano, "A matching theory framework for tasks offloading in fog computing for IoT systems," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5089–5096, 2018.

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

[11] X. Wang, Z. Ning, S. Guo, M. Wen, and V. Poor, "Minimizing the age-of-critical-information: an imitation learning-based scheduling approach under partial observations," *IEEE Transactions on Mobile Computing*, vol. 1233, p. 1, 2021.

[12] Z. Ning, Y. Yang, X. Wang et al., "Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing," *IEEE Transactions on Mobile Computing*, vol. 1233, pp. 1–1, 2021.

[13] Z. Ning, P. Dong, M. Wen et al., "5G-enabled UAV-to-community offloading: joint trajectory design and task scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3306–3320, 2021.

[14] X. Wang, Z. Ning, S. Guo, M. Wen, L. Guo, and V. Poor, "Dynamic UAV deployment for differentiated services: a multi-agent imitation learning based approach," *IEEE Transactions on Mobile Computing*, vol. 1233, pp. 1–16, 2021.

[15] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M. T. Zhou, "FEMTO: fair and energy-minimized task offloading for fog-enabled IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4388–4400, 2019.

[16] Y. Yang, K. Wang, G. Zhang, C. Xu, X. Luo, and M. T. Zhou, "Maximal energy efficient task scheduling for homogeneous fog networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*no. 5, pp. 274–279, HI, USA, 2018.

[17] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, 2016.

[18] C. Tofallis, S. I. Gass, and C. M. Harris, "Encyclopedia of operations research and management science," *Journal of the Operational Research Society*, vol. 48, no. 7, pp. 759-760, 1997.

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, 2nd edition, 2018.

[20] J. Hecht, "Faster fiber links for data centers," 2019, December 2021, https://spectrum.ieee.org/tech-talk/telecom/internet/fasterfiber-links-for-data-centers.