Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.059616

ARTICLE

An Asynchronous Data Transmission Policy for Task Offloading in Edge-Computing Enabled Ultra-Dense IoT

Dayong Wang[1,*], Kamalrulnizam Bin Abu Bakar[1], Babangida Isyaku[2] and Liping Lei[3]

[1]Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, 81310, Johor, Malaysia

[2]Department of Computer Science, Faculty of Information Communication Technology, Sule Lamido University, Kafin Hausa, 741103, Jigawa, Nigeria

[3]School of Political Science and Public Administration, East China University of Political Science and Law, Shanghai, 201620, China

*Corresponding Author: Dayong Wang. Email: wangdayong@graduate.utm.my

Received: 13 October 2024 Accepted: 14 November 2024 Published: 19 December 2024

ABSTRACT

In recent years, task offloading and its scheduling optimization have emerged as widely discussed and significant topics. The multi-objective optimization problems inherent in this domain, particularly those related to resource allocation, have been extensively investigated. However, existing studies predominantly focus on matching suitable computational resources for task offloading requests, often overlooking the optimization of the task data transmission process. This inefficiency in data transmission leads to delays in the arrival of task data at computational nodes within the edge network, resulting in increased service times due to elevated network transmission latencies and idle computational resources. To address this gap, we propose an Asynchronous Data Transmission Policy (ADTP) for optimizing data transmission for task offloading in edge-computing enabled ultra-dense IoT. ADTP dynamically generates data transmission scheduling strategies by jointly considering task offloading decisions and the fluctuating operational states of edge computing-enabled IoT networks. In contrast to existing methods, the Deep Deterministic Policy Gradient (DDPG) based task data transmission scheduling module works asynchronously with the Deep Q-Network (DQN) based Virtual Machine (VM) selection module in ADTP. This significantly reduces the computational space required for the scheduling algorithm. The continuous dynamic adjustment of data transmission bandwidth ensures timely delivery of task data and optimal utilization of network bandwidth resources. This reduces the task completion time and minimizes the failure rate caused by timeouts. Moreover, the VM selection module only performs the next inference step when a new task arrives or when a task finishes its computation. As a result, the wastage of computational resources is further reduced. The simulation results indicate that the proposed ADTP reduced average data transmission delay and service time by 7.11% and 8.09%, respectively. Furthermore, the task failure rate due to network congestion decreased by 68.73%.

KEYWORDS

Bandwidth allocation; edge computing; internet of things; task offloading; reinforcement learning

1 Introduction

The rapid development of the Internet of Things (IoT) and the continuous complexity of application scenarios have indeed led to a significant increase in the computational load on terminal devices (TDs) [1,2]. This led to the issue of slow response times for application services becoming particularly pronounced due to the insufficient computational capacity of TDs [3]. Task offloading technology allows TDs to submit computational tasks to edge-computing networks or cloud platforms, reducing the computational load on the TDs [4,5].

The most commonly applied task offloading paradigm involves TDs offloading computation-intensive tasks to cloud platforms or edge-computing networks [6]. TDs submit task offloading requests to nearby computing networks via wireless access networks and upload the data associated with the computational tasks [7]. As the number of terminal devices continues to rise, numerous computation-intensive jobs have begun to compete for resources in the edge network [8]. This trend has placed immense pressure on the network services at the wireless access layer, leading to network congestion [9]. Compared to cloud computing, edge-computing systems have relatively limited bandwidth resources. The coexistence of numerous devices poses challenges for bandwidth allocation. Additionally, reducing data processing latency is also a pressing issue that needs to be addressed [10].

Indeed, both computational and transmission delays play a critical role in influencing the overall efficacy of task offloading [11]. However, existing research has focused extensively on optimizing computational resource allocation, while considerations for transmission optimization remain relatively limited. Concurrent data uploads can lead to severe wireless network congestion, particularly in densely populated networks [12]. Therefore, the completion time of tasks increases due to the failure of task data to arrive at the corresponding virtual machine (VM) promptly. Thus, it is essential to optimize the data transmission process for task scheduling based on effective network bandwidth allocation strategies. However, such problems have been proven NP-hard due to their complex optimization requirements and dynamic environments [13]. This means that it is challenging to find an algorithm that can guarantee an optimal solution within polynomial time.

The common bandwidth allocation mechanism involves assigning different priorities to the data streams of tasks [14,15]. However, this can result in low-priority data experiencing significant transmission delays [16]. Some studies [17,18] utilize M/M/1 queues to model task processing. However, this requires ensuring that the computing nodes have sufficient resources to handle incoming data at all times. This is challenging to guarantee in edge-computing networks. Additionally, heuristic bandwidth allocation methods, while providing near-optimal solutions, often lack scalability.

AI-based network resource scheduling methods can effectively handle more complex transmission demands and environmental constraints, thereby optimizing overall network performance [19,20]. However, existing research in this area primarily focuses on optimizing IoT data transmission and abstract data communication scheduling. Some AI-based resource scheduling methods aimed at task offloading take into account the varying transmission demands of task data and the constraints of available bandwidth. Nonetheless, they all lack efficient mechanisms for scheduling the network transmission of task data to align with the dynamic allocation of VM resources. To address the aforementioned issues, we propose an asynchronous data transmission policy for task offloading in edge-computing-enabled IoT. This policy takes into full account the delay associated with the arrival of task data at the target VM, thereby reducing resource wastage by dynamically allocating the limited available network bandwidth. Specifically, our main contributions are listed as follows:

- Deconstructed the factors influencing task offloading efficiency and focused on the optimization of task data transmission in high-density task offloading scenarios. Subsequently, this NP-hard mixed integer nonlinear programming (MINLP) optimization problem was transformed into a Markov decision process (MDP) that can be solved using reinforcement learning techniques. Furthermore, the discrete nature of VM allocation was differentiated from the continuous characteristics of network bandwidth allocation to adapt to various reinforcement learning neural networks.
- Proposed an Asynchronous Data Transmission Policy (ADTP) that jointly considers the matching of tasks with computational resources and network resource constraints to dynamically allocate network bandwidth. Additionally, ADTP incorporates Quality of Experience (QoE) feedback to refine the bandwidth allocation strategy. Consequently, task data can be uploaded promptly to the corresponding VMs for processing, while simultaneously reducing the computational cost of the scheduling algorithm.
- Established a simulation environment and conducted extensive comparative experiments. The simulation results demonstrate that the proposed Asynchronous Data Transmission Policy (ADTP) utilizes limited network resources more effectively. Both the success rate of task execution and response time were enhanced due to the timely upload of task data to the corresponding virtual computing nodes.

The rest of this paper is organized as follows. Section 2 explained the relevant prior works. Section 3 describes the system model and formulates the optimization problem of task data transmission. Section 4 presents the proposed data transmission policy for IoT task offloading. The performance evaluation of the proposed policy is presented in Section 5. Finally, we provide conclusions and future research works in Section 6.

2 Related Work

Data transmission optimization based on network resource allocation has been investigated across various levels in tandem with the advancement of IoT and task offloading technologies. However, the majority of existing research fails to adequately address the optimization of data transmission within the context of task offloading scenarios.

In [21], channel resources are allocated based on the size of the task data to avoid transmission conflicts among IoT nodes. However, data size alone does not fully represent the multidimensional requirements for data transmission. In [22], the proposed transmission optimization method prioritizes data based on task deadlines; however, this prioritized transmission approach does not support parallel data transfers. In [23], a weighted fair queuing (WFQ) mechanism with delay constraints is employed to reduce overall latency. Nonetheless, fairness is not the requisite mode for task offloading data transmission. If resources for either communication or computation are insufficient, tasks must wait in the queue. Each upstream queue can only process upload tasks sequentially.

Compared to the aforementioned simple data transmission optimization schemes, heuristic and meta-heuristic algorithms can handle relatively complex data transmission constraints. In [24], an Index-Based Transmission Scheduling (IBTS) approach is employed to optimize the transmission of task data after offloading decisions. However, this scheme only considers data size and transmission rate, neglecting the timing constraints for task data arrival at the server. In [25], the proposed Grey Wolf Optimization (GWO) meta-heuristic method is utilized to optimize data transmission and reduce wireless communication energy consumption for terminal devices. Nonetheless, this method suffers from slow convergence and a tendency to get trapped in local optima.

AI-based data transmission schemes can manage a vast array of environmental parameters and task constraints, demonstrating robust capabilities for optimizing task data transmission in complex scenarios. In [26], an improved K-means algorithm is proposed for the automatic clustering of NB-IoT terminals, followed by the generation of a prioritized transmission sequence for task data. However, this method overlooks the impact of the delay in task data arrival at the edge server on task completion time. In [27], the proposed approach utilizes deep neural networks (DNN) to learn data transmission characteristics for each time slot and dynamically adjust the allocation of wireless communication resources. However, this method primarily focuses on optimizing network throughput, latency, and the conflict rate of data transmissions.

The advancement of deep reinforcement learning (DRL) techniques offers improved solutions for optimizing task data transmission due to their ability to automatically adapt to dynamic changes in the network environment. In [28], multiple Deep Q-Learning Networks (DQN) are employed for the dynamic scheduling of network transmission states and data transmission requirements, optimizing channel allocation and throughput. Similarly, DDQN has also been applied to dynamically allocate time slot resources across multiple channels [29]. However, these approaches primarily optimize at the packet transmission level and lack awareness of the overall structure of task data.

To address such independent optimization challenges, hybrid solutions that integrate multiple techniques have been proposed. However, existing methods struggle to flexibly coordinate the discrete nature of the VM selection problem with the continuous nature of the bandwidth allocation issue [30–32]. Moreover, the expansion of network scale and the proliferation of numerous application tasks impose constraints on reinforcement learning agents in tackling high-dimensional problems, as this may give rise to the curse of dimensionality. In [33], a hierarchical DDPG (HDDPG) framework was proposed, which first optimizes the data communication process and subsequently addresses the computational resource allocation problem. However, the use of two synchronously operating DDPG neural networks introduces relatively high computational costs.

To address these challenges, we propose an Asynchronous Data Transmission Policy (ADTP) for task offloading in edge-computing enabled ultra-dense IoT environment.

3 System Model and Problem Formulation

In this section, an edge-computing enabled IoT task-offloading network architecture is presented. Additionally, we describe the system model including the task computing model and the data communication model for IoT task offloading. Subsequently, the problem of minimizing task completion time is formulated as a Markov Decision Process (MDP).

We consider a typical edge-computing enabled IoT network architecture as shown in Fig. 1. Similar scenario designs are widely adopted [34]. There is a set of TDs communicating with the edge-computing network via bandwidth-limited wireless connections. The communication between the edge-computing network and the cloud platform utilizes Wide Area Network (WAN) connections. Within the edge-computing network, multiple data centers are interconnected through high-speed networks. The orchestrator in the edge-computing network is responsible for allocating virtual machine resources and wireless bandwidth to tasks from TDs based on task offloading requests and the operational state of the network. Consequently, the completion time of a task is equal to the transmission time of the task data plus the execution time of the task on the virtual machine. Each Access Point (AP) has the same total available bandwidth. The ADTP dynamically adjusts the bandwidth allocated for the transmission of each task data in each time slot. Fig. 1 illustrates the system architecture of task offloading in Multi-access Edge Computing (MEC).

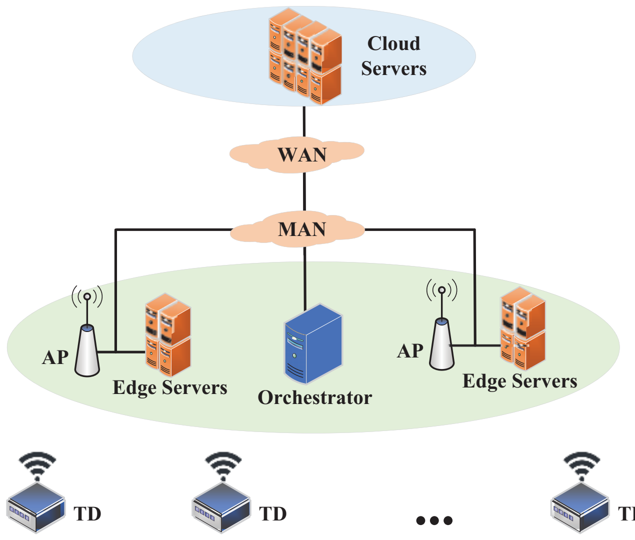

Figure 1: The system architecture of task offloading in MEC

To describe the problem under investigation, let the system's operating time be T with time slots $t \in \{1, 2, \ldots, T\}$, where each time slot has a duration of Δt. The total number of tasks and virtual machines are denoted as $I \in \mathbb{Z}^+$ and $J \in \mathbb{Z}^+$, respectively. Additionally, each task waiting to be offloaded has multiple attributes defined as $\tau_i = (C_i, D_i, S_i)$, where C_i represents the computational resources required by the task (in MIPS), D_i denotes the size of the task data (in bytes), and S_i indicates the time sensitivity of the task. Furthermore, each computing node is defined as $v_j = \left(R_j, S_j, P_j, C_{util,j}\right)$, where R_j represents the computational resources (in MIPS), S_j denotes the data storage resources (in bytes), P_j indicates the type of platform where the VM is located (1 for edge, 2 for cloud), and $C_{util,j}$ represents the resource utilization. Thus, the offloading decision variable d_{ij} indicates whether the task is assigned to a virtual machine j, this is a common definition in this type of research [20,22]:

$$d_{ij} = \begin{cases} 1, & \text{if } \tau_i \text{ is assigned to } v_j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Accepted offloaded tasks will be allocated computational resources $R_{ij}(t)$ and network bandwidth resources $B_{ij}(t)$. However, not all tasks can be completed within the specified time due to various uncertainties. Therefore, we define the failure state of a task as:

$$F_i(t) = \begin{cases} 1, & \text{if task } i \text{ fails} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Consequently, the average task failure rate can be expressed as:

$$\overline{F}(t) = \frac{1}{N}\sum_{i=1}^{N} F_i(t) \tag{3}$$

In this case, the remote server operates in timesharing mode, which means multiple tasks can be executed in parallel on the server providing computational resources. The computational delay of the offloaded tasks can be expressed by:

$$D_c(i,j) = \frac{C_i}{R_j \cdot \left(1 - C_{util,j}\right)} \tag{4}$$

where C_i represents the computational resource demand of the task, and $C_{util,j}$ denotes the resource utilization for VM$_j$. In addition to the computation time associated with tasks, another significant delay that contributes to the overall completion time is the transmission delay of task data. In the context of this study, multiple TDs are competing for the limited bandwidth of the wireless access network to upload their data to the edge-computing infrastructure. However, the transmission rate for each data transfer is subject to temporal fluctuations due to device mobility and intricate contention conflicts. Consequently, it is imperative to dynamically optimize the scheduling of available communication resources for each task's data within each time slot. Fig. 2 illustrates the fundamental principle of bandwidth adjustment for multiple data streams across time slots.

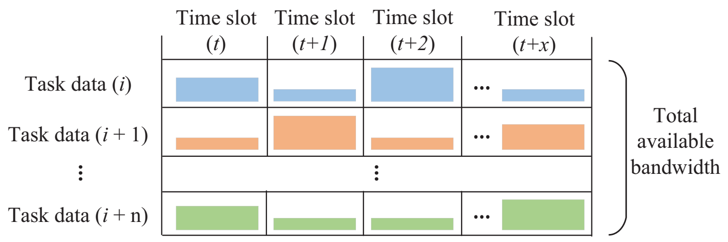

Figure 2: The fundamental principle of bandwidth adjustment for multiple data streams

Here, the downstream network transmission delay is neglected due to the size of the task computation results being significantly smaller than the size of the uploaded data, and the downstream bandwidth typically being larger than the upstream bandwidth [35–37]. The delay associated with the data transmission over the network can be denoted by:

$$D_t(i) = \frac{D_i}{B_{ij}(t)} + D_{wan}(i) \tag{5}$$

where $D_{wan}(i) = 0$ when the task is assigned to the edge server. For cloud computing, where data is transmitted over the Wide Area Network (WAN), the delay is modeled using an M/M/1 queue system due to the more complex nature of the transmission [38]. Thus, service delays can be expressed as:

$$D_s(i,j) = D_c(i,j) + D_t(i) \tag{6}$$

In high-density IoT networks, the substantial volume of upload data streams from terminal devices (TDs) competes for the limited available bandwidth of the wireless access network. The data associated with tasks should be transmitted to the matched computing nodes at appropriate times. Therefore, the optimization objective for bandwidth resource allocation is to minimize the service time of tasks while improving resource utilization: min $\sum_{i=1}^{N} D_s(i,j)$, while also enhancing the Quality of Experience (QoE): max $U\left(R_j, QoE(t)\right)$. Consequently, the objective function can be expressed as:

$$\min \mathcal{O} = w_1 \sum_{i=1}^{N} D_s(i,j) + w_2 \left(1 - \frac{1}{M} \sum_{j=1}^{M} U_j\right) - w_3 QoE(t) - w_4 \sum_{i=1}^{N} S_i$$

$$\text{s.t. C1: } \sum_{i=1}^{N} d_{ij} C_i \le R_j \cdot \left(1 - C_{util,j}\right) \forall j \tag{7}$$

$$\text{C2: } C_{util,j} < 1 \forall j$$

$$\text{C3: } \textstyle\sum_{j=1}^{M} d_{ij} = 1 \forall i$$

where w_1, w_2, w_3, w_4 are weighting coefficients and $w_1 + w_2 + w_3 + w_4 = 1$. The allocation of weights should be optimized according to the requirements of different application scenarios. Constraints C1 and C2 denote that the scheduler will not oversubscribe resources to tasks waiting for offloading. Constraints C3 specifies that each task can only be offloaded to a single computing node. For clarity, the main notations are shown in Table 1. It is worth mentioning that the model can be enhanced by adding more parameters and constraints for different application contexts.

Table 1: List of main notations

Symbol	Definition	Symbol	Definition
I	Total number of tasks	τ_i	The i-th task
J	Total number of VMs	t	Time slot
C_i	Computing resources required	v_j	The j-th VM
D_i	Size of the task data	S_i	Time sensitivity of the task i
R_j	Computing resources of the VM	S_j	Storage space for VM j
P_j	The type of platform	$C_{util,j}$	Resource utilization of the VM
d_{ij}	Task offloading decision	F_i	Failed task
R_{ij}	Allocated computing resources	B_{ij}	Allocated network resources
D_c	Task calculation delay	D_t	Data transmission delay
D_{wan}	Data transmission delay on WAN	QoE	Quality of experience
$s_t, s_t^{'}$	State space	$a_t, a_t^{'}$	Action space
r, r'	Reward function	w	Weight coefficient of the objective

4 Proposed Asynchronous Data Transmission Policy (ADTP) Based on Reinforcement Learning

In this section, we address the optimization problem presented in Section 3. An asynchronous resource allocation-based data transmission optimization strategy is proposed, which aims to minimize the average completion time of tasks. This optimization method first utilizes Deep Q-Network (DQN) to determine the virtual machine to which task offloading requests are assigned. Subsequently, it employs a Deep Deterministic Policy Gradient (DDPG) approach to dynamically generate bandwidth allocation decisions for task data transmission. Fig. 3 illustrates the proposed Asynchronous Data Transmission Protocol (ADTP) logical framework. Furthermore, the VM allocation decisions are event-driven and are not repeated in every time slot to reduce computational load and energy consumption. In contrast, adjustments to wireless communication resources occur in every time slot due to the mobility of the TD and the rapid changes in network conditions.

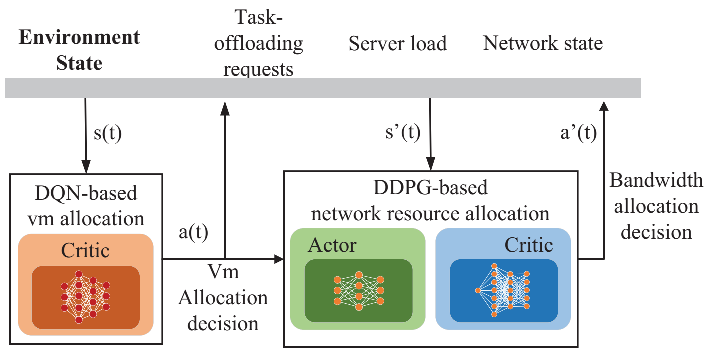

Figure 3: The architecture of CRADTP

For VM allocation, we employ the widely recognized DQN-based approach to facilitate offloading decisions [39–41]. The inputs to the DQN encompass the operational state of the edge network, the resource requirements of the tasks, and the failure rates associated with task offloading. The output generated by the decision-making algorithm is an allocation vector that delineates the offloading of tasks to designated VMs. The reward for the algorithm is derived from the weighted sum of the service time and user satisfaction of the offloaded tasks, thereby incentivizing optimal performance in task allocation. The state-action space and reward definitions for the reinforcement learning algorithm are as follows:

State Space: Let S represents the system state space. The current system state space $s \in S$ is expressed as:

$$s_t = \left\{d_j^i(t), R_j(t), \overline{F}(t), D_s(i,j), C_{utili,j}(t), C_i, D_i\right\} \tag{8}$$

where $d_j^i(t)$ denotes the distribution of the task on the virtual machine based on ongoing offloading decisions. $R_j(t)$ is the available MIPS on the server j. $\overline{F}(t)$ is the average task failure rate. $D_s(i,j)$ is the service time. $C_{utili,j}(t)$ denotes the computational resource utilization of VM_j. C_i is the Computing resources required for the task i. D_i is the data size of task i.

Action Space: The action space for the DQN algorithm will be the selection of a VM for a particular task. Thus, the action $a \in A$ can be given by:

$$a_t = \left\{d_{ij}\right\} \tag{9}$$

where a_t is the action to allocate task i to VM_j.

Reward: The reward function will be designed to encourage the allocation of tasks to the servers that can minimize the expected completion time.

$$r(s,a) = -\left(\beta_1 D_s(i,j) + \beta_2 vQoE(t)\right) \tag{10}$$

where β_1 and β_2 stand for the weight parameters, which indicate the preferences for service time and satisfaction. Here, $\beta_1, \beta_2 \in [0, 1]$ and $\beta_1 + \beta_2 = 1$.

Overall, the DQN iteratively learns and generates offloading decisions for the matching of tasks and VMs. The pseudocode for the algorithm is presented in Algorithm 1.

Algorithm 1: DQN-based VM allocation for task offloading

Input: Computational resource network state and resource requirements for task offloading: $\left\{d_j^i(t), R_j(t), \overline{F}(t), D_s(i,j), C_{utili,j}(t), C_i, D_i\right\}$

Output: Matching decisions between the tasks requesting offloading and the virtual machines providing computational resources: $\left\{d_{ij}\right\}$

```
1   Initialize DQN with parameters $\theta$ randomly and empty replay memory
2   for each time step $t$ do
3       Observe current state s
4       if a new request arrives or task computation is completed then
5           for each task-offloading request $\tau_i$ do
6               Choose an action $a$ based on the Q-value
7               Output decision
8               Observe the reward and next state
9               Store transition in replay memory
10              if length(replay_memory) > batch_size then
11                  Sample a batch of data from memory
12                  Train the Q-network and update
13              end if
14          end for
15      end if
16  end for
```

As illustrated in Algorithm 1, the neural network and replay memory are initialized in the first line. The second line initiates a loop to iteratively execute the core functionalities of the algorithm at each time step. Lines 3 and 4 evaluate the necessity for offloading decisions and algorithm updates based on the current state of the edge network. The decision-making component remains inactive when no new offloading requests are received and no task computations are completed, thereby minimizing computational resource usage and energy consumption. Lines 6 to 9 execute the generation and output of resource-matching decisions based on reinforcement learning. Finally, Lines 10 to 12 are devoted to the periodic updating of the neural network to enhance its adaptability to scheduling optimizations. The core unit of the algorithm operates based on DQN, allocating VM resources for incoming task offloading requests in each time slot. Therefore, the algorithm has a time complexity of O(N).

In the context of the scenarios addressed in this study, we adopt the widely recognized DDPG-based [42–44] approach to dynamically allocate bandwidth in each time slot, capitalizing on its self-learning capabilities and proficiency in generating continuous values. The state-action space and reward definitions for the reinforcement learning algorithm are as follows:

State Space: The state space for the DDPG algorithm will represent the current state of the network regarding bandwidth allocation, which includes:

$$s_t' = \left\{a_t, R_j(t), \overline{F}(t), D_s(i,j)\right\} \tag{11}$$

where a_t is the decision made by the DQN for offloading tasks to the VM. $R_j(t)$ denotes the computational resources of VM_j. $\overline{F}(t)$ represents the average task failure rate. $D_s(i,j)$ is the service time of the task τ_i.

Action Space: The action space for the DDPG algorithm will be the bandwidth allocation for each task in each time slot.

$$a_t^{'} = \left\{B_{ij}(t)\right\} \tag{12}$$

where $a_t^{'}$ is the bandwidth allocated to task τ_i in time slot t.

Reward: The reward function for the DDPG algorithm will encourage efficient bandwidth allocation that minimizes the total transmission time for all tasks.

$$r'(s', a') = -\left(\delta_1 D_s(i,j) + \delta_2 vQoE(t) + \delta_3 C_{net}(t)\right) \tag{13}$$

where δ_1, δ_2 and δ_3 represent the weight parameters, which specify the preferences for service time, satisfaction and network utilization. Here, $\delta_1, \delta_2, \delta_3 \in [0, 1]$ and $\delta_1 + \delta_2 + \delta_3 = 1$.

The pseudocode for the algorithm is presented in Algorithm 2.

Algorithm 2 : DDPG-based bandwidth allocation

Input: Allocation decisions for computing nodes, the operational condition of the wireless access network, and the actual completion efficiency of the tasks: $\left\{a_t, R_j(t), \overline{F}(t), D_s(i,j)\right\}$

Output: The available bandwidth allocated for the upload data of each approved offloaded task in each time slot: $\left\{B_{ij}(t)\right\}$

```
1    Initialize actor_network and critic_network with parameters $\theta^{\mu}$ and $\theta^{Q}$
2    Copy parameters of the main network to target network $\theta^{\mu'}$ and $\theta^{Q'}$
3    Iniyislize replay_memory
4    Set learning_rates, discount_factor, soft_update_factor, batch_size, update_ frequency
5    for each episode do
6        for each time step do
7            for each task do
8                Make an action $a'(t) = \theta^{\mu}(s'(t))$ with the parameter $\theta^{\mu}$
9                Output the bandwidth allocation decision $a'(t)$
10               Obtain the reward $r'$ and next state $s'(t+1)$
11               Store the transition tuple in experience replay_memory
12           end for
13           if length(replay_memory) > batch_size then
14               Randomly sample from replay_memory
15               Compute target Q-value, loss and perform policy gradient
16               Update Critic network parameters
17               Compute policy gradient and Update Actor parameters
18           end if
19           if $t$ mod update_frequency $== 0$ then
20               Update the parameters $\theta^{Q'}$, $\theta^{\mu'}$ of the actor's target network and the critic's target
                 network
21           end if
22       end for
23   end for
```

As illustrated in Algorithm 2, the Actor-Critic networks are created in Lines 1 and 2. Correspondingly, various hyper-parameters and the buffer for experience replay are initialized in Lines 3 and 4. Lines 5 to 7 establish three nested loops to iterate through each training epoch, each time slot, and each task. Lines 8 to 11 are dedicated to generating bandwidth allocation decisions and saving the replay memory records. Lines 13 to 18 conditionally execute model training. Finally, Lines 19 to 21 update the target networks when the update conditions are met. The time complexity of the DDPG part in the algorithm is linear. When the algorithm operates continuously in each time slots, the overall time complexity is O(N).

5 Performance Evaluation

In this section, we verify the performance of the proposed task data transmission policy through simulation experiments. We use EdgeCloudSim [34] as the foundational environment for task offloading simulation. Additionally, we extend the network transmission and virtual machine allocation modules of EdgeCloudSim. Thus, the proposed ADTP and the comparison algorithms can be implemented. The resource allocation module, based on deep neural networks, is implemented in PyTorch 3.8 and provides functional enhancements to EdgeCloudSim via a custom developed interface. The computer hardware comes with inter I7 CPU, 16 GB RAM to support the operation of the simulation environment. The detailed parameters of the experiments are shown in Table 2.

Table 2: Simulation parameters

Parameters	Value
Iterations	50
Simulation time (minutes)	60
Number of VMs on cloud/edge network	4/28
Core of VM on cloud/edge network	4/2
MIPS of VM on cloud/edge network (MIPS)	10,000/1000
Ram of VM on cloud/edge network (MB)	32,000/2000
Storage of VM on cloud/edge network (Byte)	1,000,000/50,000
Bandwidth of WAN/WLAN/MAN (Mbps)	15/100/1000
Number of TDs	100∼1200
Task length of AR/HA/HCA/IA (MI)	2000/400/3000/750
Data upload of AR/HA/HCA/IA (KB)	2100/1850/3000/725
Data download of AR/HA/HCA/IA (KB)	25/20/250/2500

In the context of task offloading, the most significant factors affecting user experience are delay and failure rate. Therefore, we concentrated on evaluating task data transmission delay, average service time, and the effects of various factors on task offloading failure rates to demonstrate the advantages of ADTP. To visually evaluate the performance of the proposed ADTP, the following benchmark strategies are simulated for comparison:

- **Bandwidth Sharing (BS):** This serves as the baseline. In this scheme, all tasks engage in unrestricted competition for data uploads.
- **Relative-execution Deadline First (RDF):** In this scheme, the priority of each task's data flow is dynamically adjusted based on deadlines when bandwidth resources become scarce [45].
- **Double Deep Q-Network (DDQN):** In this scheme, the bandwidth resources of the AP are divided into multiple virtual channels of identical performance. A deep learning network collects information on data transmission requests and network bandwidth conditions to dynamically optimize bandwidth allocation [46].
- **Asynchronous Data Transmission Policy (ADTP):** Proposed dynamic data transmission strategy based on asynchronous reinforcement learning to facilitate optimized scheduling of task data flows and enhance the overall performance of task offloading.

Network latency is the primary focus of this study. Fig. 4 compares the network latency of various solutions. The baseline free competition scheme exhibits lower performance due to a high incidence of conflicts during the competition process, resulting in a significant number of data transmission failures. In contrast, the RDF approach strives to ensure that multiple packets belonging to the same task are uploaded with priority. Furthermore, the DDQN-based solution is capable of comprehensively gathering information regarding data transmission requests and network bandwidth conditions to dynamically optimize bandwidth allocation, thereby demonstrating superior performance. However, it lacks awareness of task execution processes within the virtual machines (VMs). Additionally, the fixed-bandwidth allocation of network resources into virtual channels introduces a challenge related to excessive scheduling granularity. In contrast, the Asynchronous Data Transmission Policy (ADTP) holistically considers task offloading decisions and network conditions to deliver refined network resource allocations. Thus, this approach exhibits significantly higher performance.

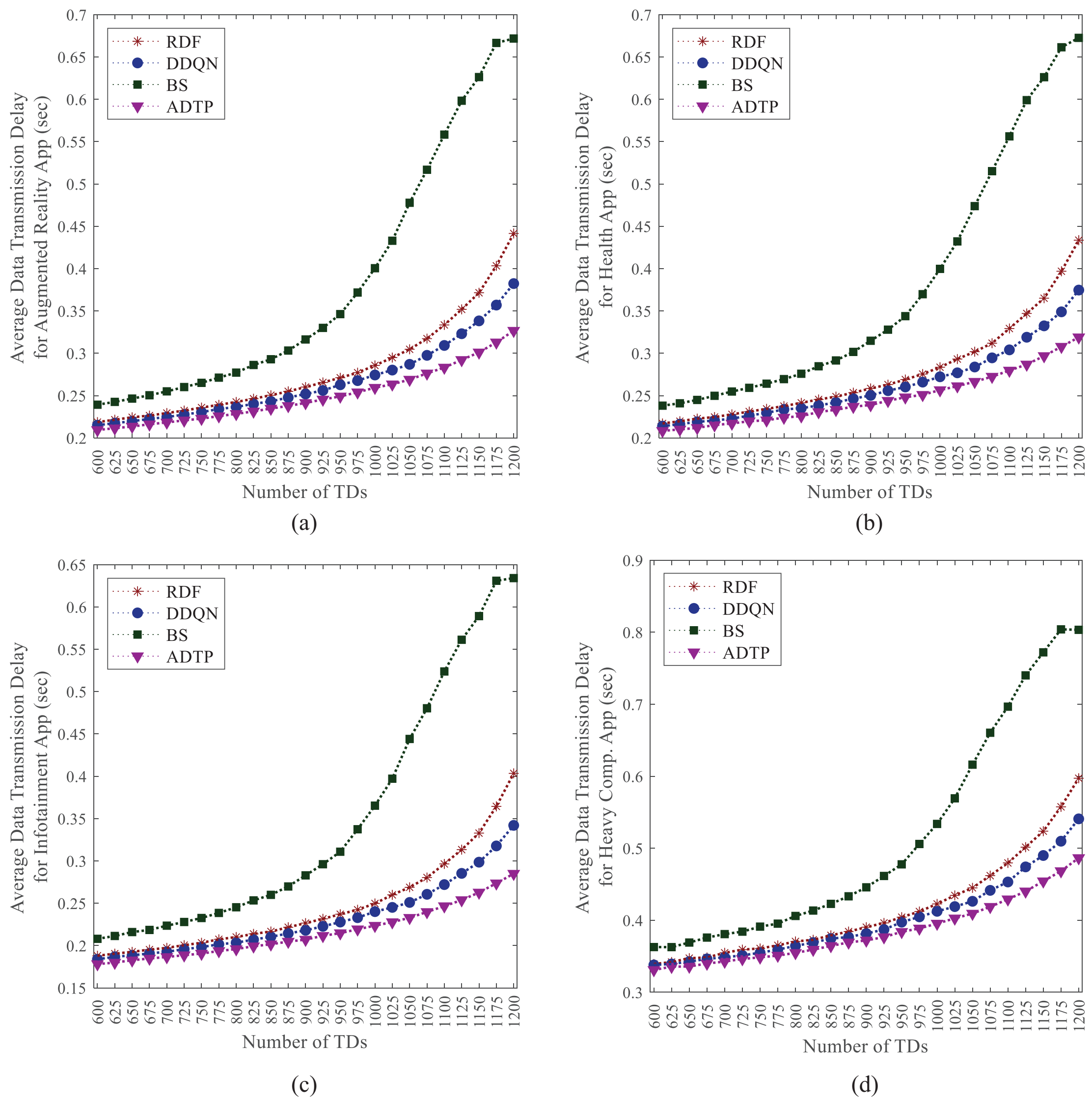

Figure 4: Average data transmission delay for (a) Augmented reality App; (b) Health App; (c) Infotainment App; (d) Heavy Comp. App

Additionally, network latency and computational latency together influence the overall service time for tasks. Fig. 5 presents a comparative analysis of service times for each application, further highlighting the relative advantages of the ADTP. This superiority is attributable to the fact that network transmission latency constitutes a critical component of overall service time. As the duration of task data transmission diminishes, there is a corresponding reduction in service time. It is noteworthy that the disparities in service times across the various schemes remain relatively minimal when the number of mobile devices is limited, as the data flow within the wireless network is not significantly congested. Conversely, the influence of data transmission scheduling on service time becomes markedly more pronounced as the number of mobile devices increases.

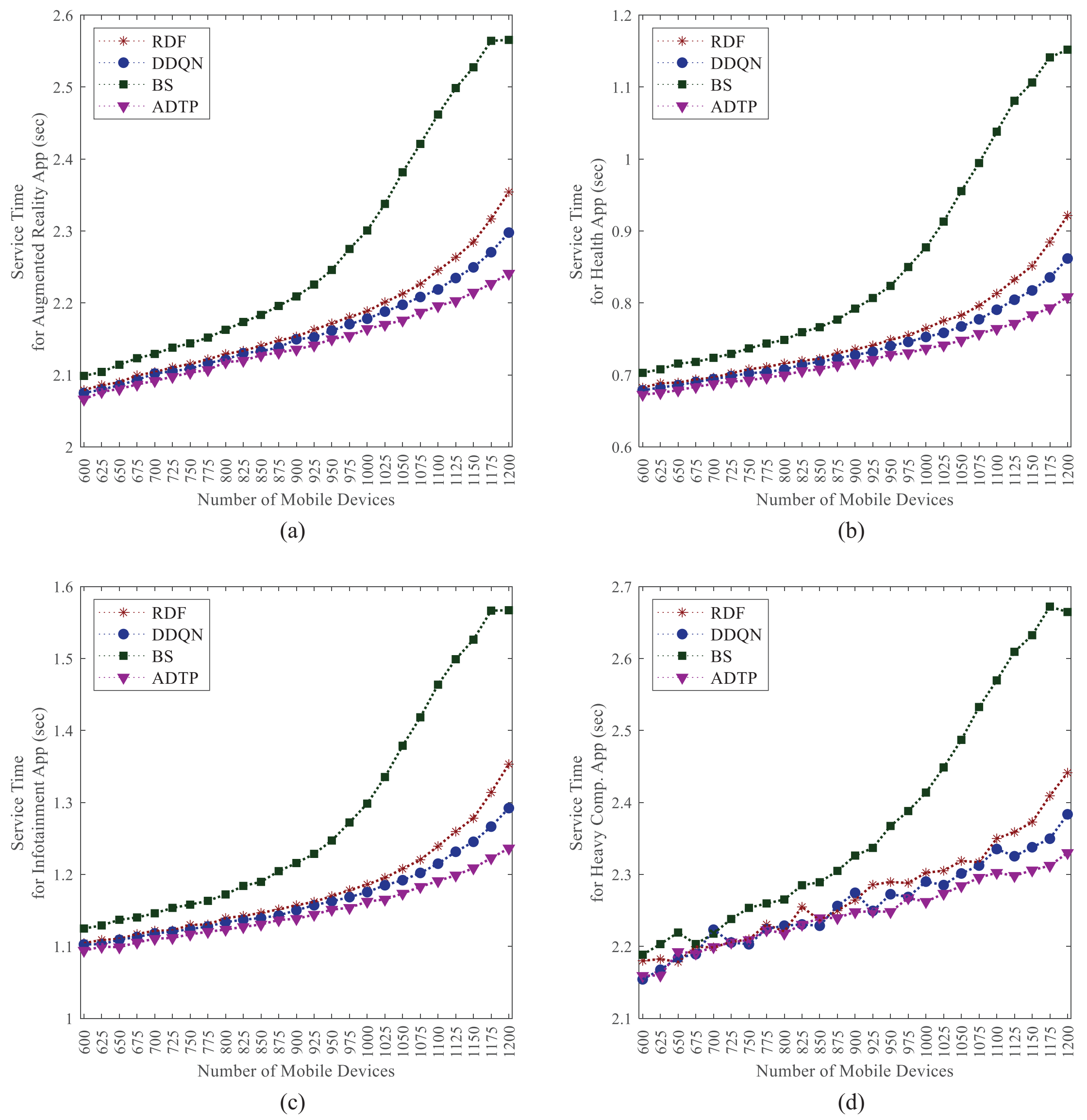

Figure 5: Service time for (a) Augmented reality App; (b) Health App; (c) Infotainment App; (d) Heavy Comp. App

While many factors can lead to task offloading failures, the primary challenges stem from inadequate computational resources and limited network transmission capabilities. As shown in Fig. 6, both data upload conflicts and resource limitations of VMs are equally significant contributors to task failures when the number of TDs is relatively low. However, as the number of TDs increases, inefficient mechanisms for network resource allocation give rise to communication-related issues, which subsequently become the predominant factor leading to task failures. In this context, the proportion of task failures attributable to computational resource constraints exhibits a consistent decline. Conversely, the ADTP demonstrates effective data transmission optimization capabilities, which results in computational blocking becoming the main reason for task failures.

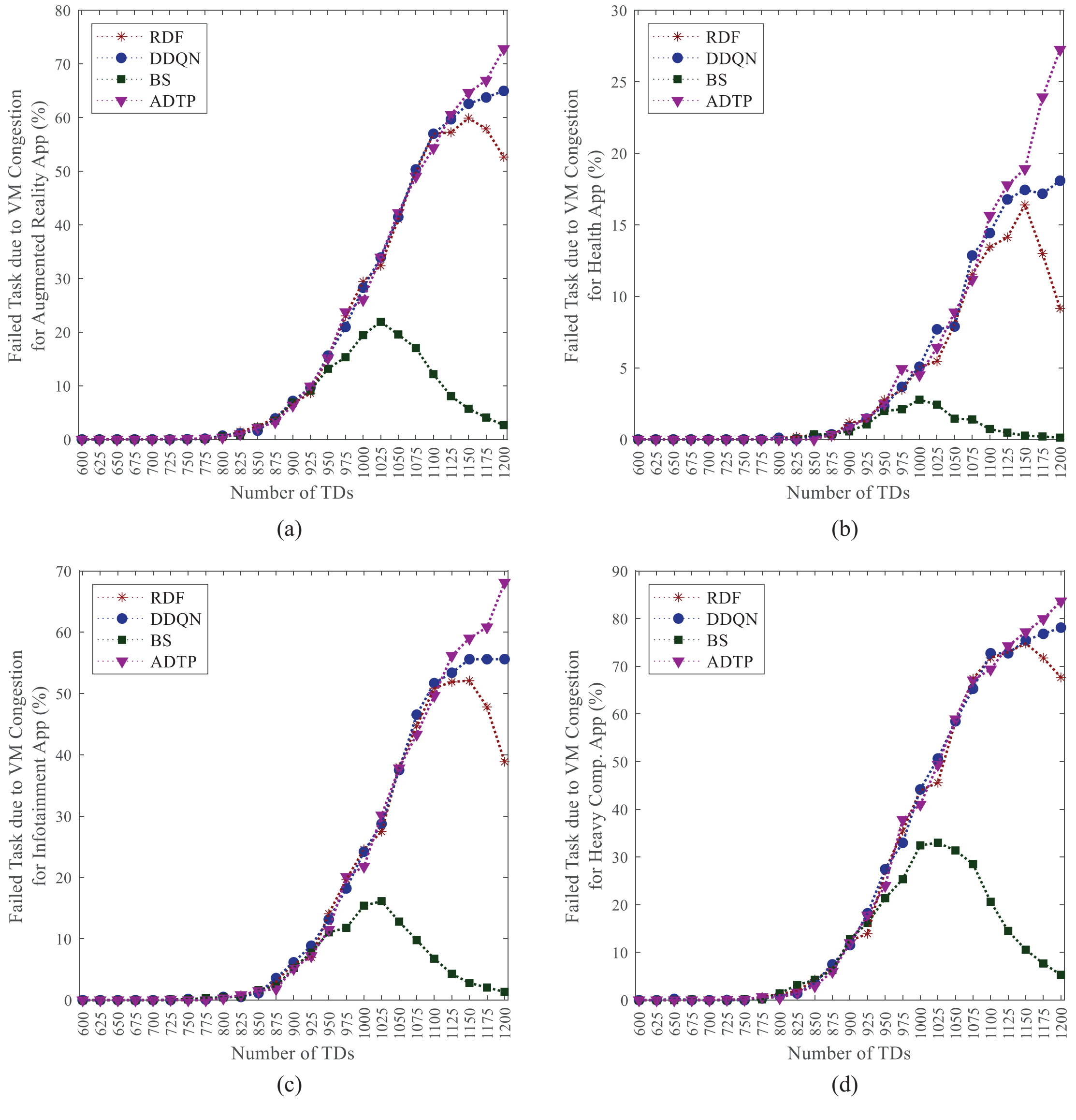

Figure 6: Failed task due to VM congestion for (a) Augmented reality App; (b) Health App; (c) Infotainment App; (d) Heavy Comp. App

Fig. 7 illustrates the relationship between data transmission efficiency and task failure rates. Notably, tasks offloaded to the edge-computing network require data uploads solely through the wireless access network. In contrast, tasks directed to the cloud platform necessitate multi-hop data transmission through both the WAN and the wireless access network. This study concentrates exclusively on optimizing data transmission within the wireless access network, while the WAN is configured in a resource-sharing mode. Analysis of Fig. 7 reveals that when the density of task offloading is low, the influence of various network resource scheduling strategies on task failure rates is minimal. This phenomenon can be attributed to the capability of task data to reach computing nodes within an acceptable time frame. Conversely, as the volume of offloaded tasks escalates, the efficient scheduling facilitated by the ADTP significantly mitigates data transmission delays and failure rates, thereby resulting in a decreased overall task offloading failure rate.

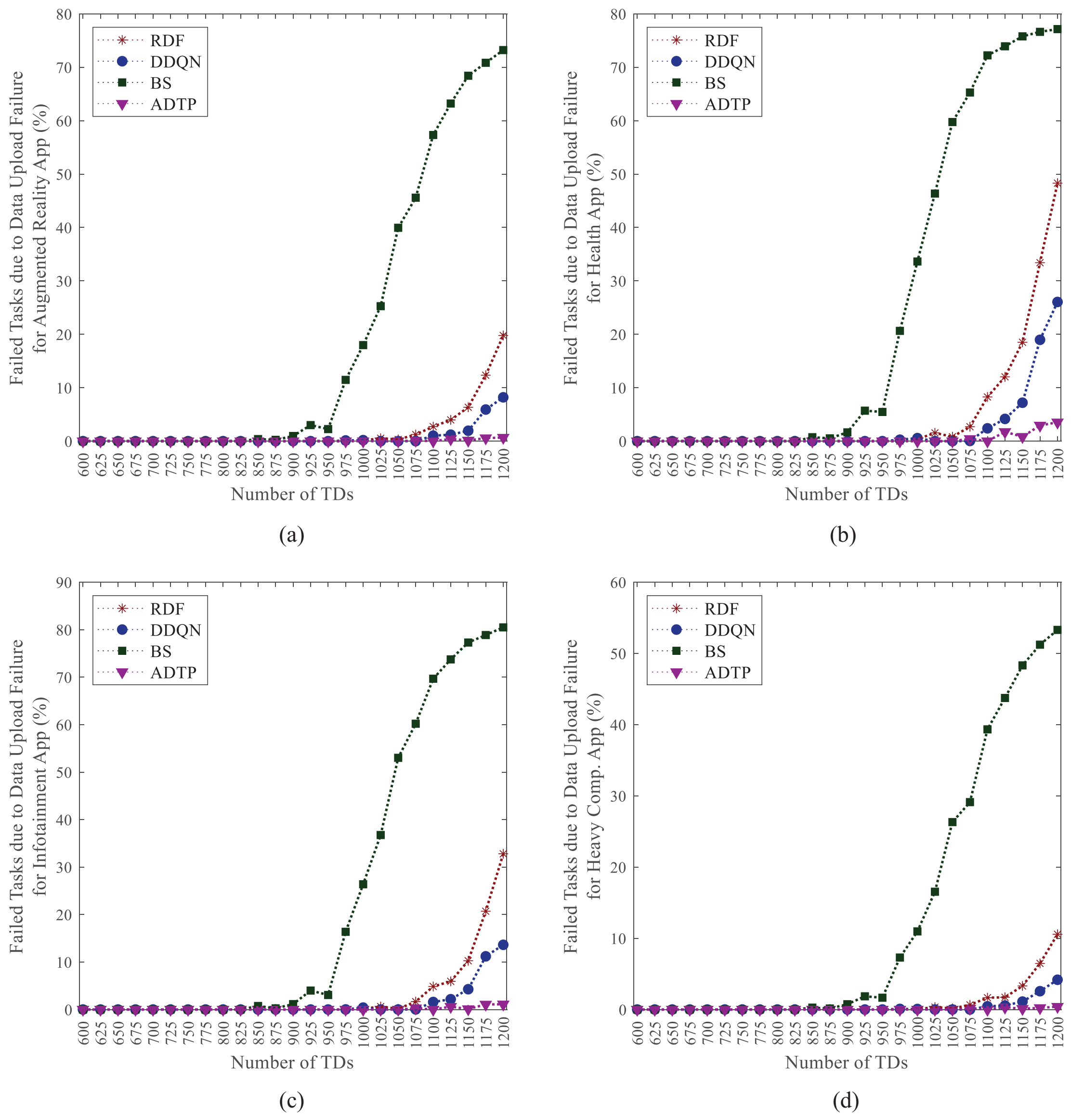

Figure 7: Failed task due to data upload failure for (a) Augmented reality App; (b) Health App; (c) Infotainment App; (d) Heavy Comp. App

In summary, the ADTP effectively monitors the requirements for task data transmission and the fluctuations in network operating conditions to generate optimized bandwidth resource allocation strategies. As a result, task data can be transmitted efficiently to VMs, rationally utilizing the available network bandwidth. Thus, the overall operational performance of the task offloading network is enhanced.

6 Conclusion

Task offloading technology has developed significantly in this information age. In this paper, we propose and focus on the optimization of data transmission within task offloading scheduling in high-density IoT network environments. To address this issue, we consider the dynamic allocation of available bandwidth for each uploaded task data stream, facilitating the efficient execution of tasks on virtual machines based on the timely arrival of task data at the virtual computing nodes. We propose an Asynchronous Data Transmission Policy (ADTP) to jointly consider the matching of tasks with virtual machines and the congestion status of the wireless access network. ADTP comprises two asynchronously functioning intelligent agents, each focusing on the discrete VM allocation decisions and the continuous bandwidth adjustment, aimed at optimizing the overall task offloading data transmission.

Experimental results indicate that ADTP effectively reduces network latency and improves data transmission efficiency in resource-constrained networks. Consequently, more tasks can be timely transferred to VM hosts for execution. This reduces the service time for task offloading and increases the success rate of task offloading. Notably, as the number of TDs increases, task offloading failures tend to be primarily driven by insufficient VM resources rather than transmission barriers. This further demonstrates that ADTP effectively optimizes the transmission process of task data.

The limitation of this study lies in its exclusive focus on optimizing the data uploads of task data from the terminal device, without addressing the optimization of data transmission from the edge network to the cloud. In future work, we aim to jointly optimize the data transmission process of task data within the edge-cloud continuum. Additionally, we will consider incorporating Spiking Neural Network (SNN) technology to reduce the computational costs and energy consumption of decision generation.

Acknowledgement: We thank UTM for supporting us during this work.

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

Author Contributions: The authors confirm their contribution to the paper as follows: Dayong Wang contributed to conceptualization and writing—original draft preparation, Babangida Isyaku and Dayong Wang contributed to methodology, Dayong Wang and Liping Lei contributed to integrate the results from multiple rounds of experiments and analysis, Babangida Isyaku and Kamalrulnizam Bin Abu Bakar contributed to writing reviews and editing, and Kamalrulnizam Bin Abu Bakar supervised the process of the research. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

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

References

[1] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *J. Syst. Archit.*, vol. 118, Sep. 2021, Art. no. 102225. doi: 10.1016/j.sysarc.2021.102225.

[2] B. Isyaku, K. B. A Bakar, N. M. Yusuf, M. Abaker, A. Abdelmaboud and W. Nagmeldin, "Software defined wireless sensor load balancing routing for internet of things applications: Review of approaches," *Heliyon*, vol. 10, May 2024. doi: 10.1016/j.heliyon.2024.e29965.

[3] N. Kumari, A. Yadav, and P. K. Jana, "Task offloading in fog computing: A survey of algorithms and optimization techniques," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109137. doi: 10.1016/j.comnet.2022.109137.

[4] S. Dong, Y. Xia, and J. Kamruzzaman, "Quantum particle swarm optimization for task offloading in mobile edge computing," *IEEE Trans. Ind. Inform.*, vol. 19, no. 8, pp. 9113–9122, Aug. 2023. doi: 10.1109/TII.2022.3225313.

[5] W. Dayong, K. B. A. Bakar, B. Isyaku, T. A. E. Eisa, and A. Abdelmaboud, "A comprehensive review on internet of things task offloading in multi-access edge computing," *Heliyon*, vol. 10, no. 9, May 2024. doi: 10.1016/j.heliyon.2024.e29916.

[6] A. Javadpour, A. Nafei, F. Ja'fari, P. Pinto, W. Zhang and A. K. Sangaiah, "An intelligent energy-efficient approach for managing IoE tasks in cloud platforms," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 4, pp. 3963–3979, Apr. 2023. doi: 10.1007/s12652-022-04464-x.

[7] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, Jun. 2022, Art. no. 103366. doi: 10.1016/j.jnca.2022.103366.

[8] A. Javadpour *et al.*, "An energy-optimized embedded load balancing using DVFS computing in cloud data centers," *Comput. Commun.*, vol. 197, pp. 255–266, Jan. 2023. doi: 10.1016/j.comcom.2022.10.019.

[9] M. Y. Akhlaqi and Z. B. Mohd Hanapi, "Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions," *J. Netw. Comput. Appl.*, vol. 212, Mar. 2023, Art. no. 103568. doi: 10.1016/j.jnca.2022.103568.

[10] J. Zhang *et al.*, "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4283–4294, Jun. 2019. doi: 10.1109/JIOT.2018.2875917.

[11] M. Z. Khan, O. H. Alhazmi, M. A. Javed, H. Ghandorh, and K. S. Aloufi, "Reliable internet of things: Challenges and future trends," *Electronics*, vol. 10, no. 19, Jan. 2021, Art. no. 19. doi: 10.3390/electronics10192377.

[12] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 233:1–233:38, Sep. 2022. doi: 10.1145/3513002.

[13] S. Dong *et al.*, "Task offloading strategies for mobile edge computing: A survey," *Comput. Netw.*, vol. 254, Dec. 2024, Art. no. 110791. doi: 10.1016/j.comnet.2024.110791.

[14] C. Jong, Y. C. Kim, J. H. So, and K. C. Ri, "QoS and energy-efficiency aware scheduling and resource allocation scheme in LTE-A uplink systems," *Telecommun. Syst.*, vol. 82, no. 2, pp. 175–191, Feb. 2023. doi: 10.1007/s11235-022-00980-5.

[15] B. Isyaku, K. A. Bakar, S. Abdulrahman, M. N. Yusuf, F. B. Muchtar and F. A. Ghaleb, "Mobile device influence on SDN controller performance in IoT-managed software-defined wireless networks," in *Advances in Intelligent Computing Techniques and Applications*, F. Saeed, F. Mohammed, Y. Fazea, Eds. Cham: Springer Nature Switzerland, 2024, pp. 62–72. doi: 10.1007/978-3-031-59707-7_6.

[16] B. Isyaku, K. B. A. Bakar, W. Nagmeldin, A. Abdelmaboud, F. Saeed and F. A. Ghaleb, "Reliable failure restoration with bayesian congestion aware for software defined networks," *Comput. Syst. Sci. Eng.*, vol. 46, no. 3, 2023. doi: 10.32604/csse.2023.034509.

[17] X. Dai *et al.*, "Task co-offloading for D2D-assisted mobile edge computing in industrial internet of things," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 480–490, Jan. 2023. doi: 10.1109/TII.2022.3158974.

[18] C. Tang, C. Zhu, N. Zhang, M. Guizani, and J. J. P. C. Rodrigues, "SDN-assisted mobile edge computing for collaborative computation offloading in industrial internet of things," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24253–24263, Dec. 2022. doi: 10.1109/JIOT.2022.3190281.

[19] Y. Li and W. Zhang, "Task-offloading strategy of mobile edge computing for WBANs," *Electronics*, vol. 13, no. 8, Jan. 2024, Art. no. 8. doi: 10.3390/electronics13081422.

[20] Y. Dai, J. Zhao, J. Zhang, Y. Zhang, and T. Jiang, "Federated deep reinforcement learning for task offloading in digital twin edge networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 2849–2863, May 2024. doi: 10.1109/TNSE.2024.3350710.

[21] C. -W. Huang, S. -C. Tseng, P. Lin, and Y. Kawamoto, "Radio resource scheduling for narrowband internet of things systems: A performance study," *IEEE Netw.*, vol. 33, no. 3, pp. 108–115, May 2019. doi: 10.1109/MNET.2018.1700386.

[22] S. Pandiyan, T. S. Lawrence, V. Sathiyamoorthi, M. Ramasamy, Q. Xia and Y. Guo, "A performance-aware dynamic scheduling algorithm for cloud-based IoT applications," *Comput. Commun.*, vol. 160, pp. 512–520, Jul. 2020. doi: 10.1016/j.comcom.2020.06.016.

[23] W. K. G. Seah, C. -H. Lee, Y. -D. Lin, and Y. -C. Lai, "Combined communication and computing resource scheduling in sliced 5G multi-access edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 3, pp. 3144–3154, Mar. 2022. doi: 10.1109/TVT.2021.3139026.

[24] A. Hazra, P. K. Donta, T. Amgoth, and S. Dustdar, "Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial IoT applications," *IEEE Internet Things J.*, 2022. doi: 10.1109/JIOT.2022.3150070.

[25] N. Unnisa and M. Tatineni, "Intelligent allocation strategy of mobile users for multi-access edge computing resources," presented at 2021 Emerging Trends in Industry 4.0 (ETI 4.0), Raigarh, India, May 2021, pp. 1–7. doi: 10.1109/ETI4.051663.2021.9619420

[26] X. Chen, Z. Li, Y. Chen, and X. Wang, "Performance analysis and uplink scheduling for QoS-aware NB-IoT networks in mobile computing," *IEEE Access*, vol. 7, pp. 44404–44415, 2019. doi: 10.1109/ACCESS.2019.2908985.

[27] J. Gao, M. Li, W. Zhuang, X. Shen, and X. Li, "MAC for machine-type communications in industrial IoT—Part II: Scheduling and numerical results," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9958–9969, Jun. 2021. doi: 10.1109/JIOT.2020.3045831.

[28] N. Jiang, Y. Deng, A. Nallanathan, and J. A. Chambers, "Reinforcement learning for real-time optimization in NB-IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1424–1440, Jun. 2019. doi: 10.1109/JSAC.2019.2904366.

[29] J. Wu, G. Zhang, J. Nie, Y. Peng, and Y. Zhang, "Deep reinforcement learning for scheduling in an edge computing-based industrial internet of things," *Wirel Commun. Mob. Comput.*, vol. 2021, p. e8017334, 2021. doi: 10.1155/2021/8017334.

[30] L. P. Qian, H. Zhang, Q. Wang, Y. Wu, and B. Lin, "Joint multi-domain resource allocation and trajectory optimization in UAV-assisted maritime IoT networks," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 539–552, Jan. 2023. doi: 10.1109/JIOT.2022.3201017.

[31] Y. Gong, H. Yao, J. Wang, M. Li, and S. Guo, "Edge intelligence-driven joint offloading and resource allocation for future 6G industrial internet of things," *IEEE Trans. Netw. Sci. Eng.*, 2022. doi: 10.1109/TNSE.2022.3141728.

[32] Y. Hao, Z. Song, Z. Zheng, Q. Zhang, and Z. Miao, "Joint communication, computing, and caching resource allocation in LEO satellite MEC networks," *IEEE Access*, vol. 11, pp. 6708–6716, 2023. doi: 10.1109/ACCESS.2023.3237701.

[33] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Oct. 2020. doi: 10.1109/TNSE.2020.2978856.

[34] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Transacti. Emerging Telecommunicati. Technologi*, vol. 29, no. 11, 2018, Art. no. 3493. doi: 10.1002/ett.3493.

[35] Y. Chiang *et al.*, "Management and orchestration of edge computing for IoT: A comprehensive survey," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14307–14331, Aug. 2023. doi: 10.1109/JIOT.2023.3245611.

[36] K. Sadatdiynov, L. Cui, L. Zhang, J. Z. Huang, N. N. Xiong and C. Luo, "An intelligent hybrid method: Multi-objective optimization for MEC-enabled devices of IoE," *J. Parallel Distrib. Comput.*, vol. 171, pp. 1–13, Jan. 2023. doi: 10.1016/j.jpdc.2022.09.008.

[37] P. Yang, R. Ma, M. Yi, Y. Zhang, B. Li and Z. Bai, "A computation offloading strategy for multi-access edge computing based on DQUIC protocol," *J. Supercomput.*, vol. 80, pp. 18285–18318, May 2024. doi: 10.1007/s11227-024-06176-9.

[38] A. Roy, J. L. Pachuau, and A. K. Saha, "An overview of queuing delay and various delay based algorithms in networks," *Computing*, vol. 103, no. 10, pp. 2361–2399, Oct. 2021. doi: 10.1007/s00607-021-00973-3.

[39] K. Moghaddasi, S. Rajabi, F. S. Gharehchopogh, and A. Ghaffari, "An advanced deep reinforcement learning algorithm for three-layer D2D-edge-cloud computing architecture for efficient task offloading in the Internet of Things," *Sustain. Comput. Inform. Syst.*, vol. 43, Sep. 2024, Art. no. 100992. doi: 10.1016/j.suscom.2024.100992.

[40] I. Khan, X. Tao, G. M. S. Rahman, W. U. Rehman, and T. Salam, "Advanced energy-efficient computation offloading using deep reinforcement learning in MTC edge computing," *IEEE Access*, vol. 8, pp. 82867–82875, 2020. doi: 10.1109/ACCESS.2020.2991057.

[41] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, Sep. 2021. doi: 10.1109/TCCN.2021.3066619.

[42] H. Mai Do, T. P. Tran, and M. Yoo, "Deep reinforcement learning-based task offloading and resource allocation for industrial IoT in MEC federation system," *IEEE Access*, vol. 11, pp. 83150–83170, 2023. doi: 10.1109/ACCESS.2023.3302518.

[43] M. A. Ebrahim, G. A. Ebrahim, H. K. Mohamed, and S. O. Abdellatif, "A deep learning approach for task offloading in multi-UAV aided mobile edge computing," *IEEE Access*, vol. 10, pp. 101716–101731, 2022. doi: 10.1109/ACCESS.2022.3208584.

[44] D. S. Lakew, A. -T. Tran, N. -N. Dao, and S. Cho, "Intelligent offloading and resource allocation in heterogeneous aerial access IoT networks," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 5704–5718, Apr. 2023. doi: 10.1109/JIOT.2022.3161571.

[45] C. Xia, X. Jin, C. Xu, Y. Wang, and P. Zeng, "Real-time scheduling under heterogeneous routing for industrial internet of things," *Comput. Electr. Eng.*, vol. 86, Sep. 2020, Art. no. 106740. doi: 10.1016/j.compeleceng.2020.106740.

[46] H. Zhong, R. Sun, F. Mei, Y. Chen, F. Jin and L. Ning, "Deep grid scheduler for 5G NB-IoT uplink transmission," *Secur. Commun. Netw.*, vol. 2021, pp. 1–10, Aug. 2021. doi: 10.1155/2021/5263726.