Accepted Manuscript

DEED: Dynamic Energy-Efficient Data offloading for IoT applications under unstable channel conditions

Hui Yan, Xiongtao Zhang, Huangke Chen, Yun Zhou, Weidong Bao, Laurence T. Yang

PII: S0167-739X(18)32214-3
DOI: https://doi.org/10.1016/j.future.2019.01.014
Reference: FUTURE 4707

To appear in: *Future Generation Computer Systems*

Received date : 22 September 2018
Revised date : 12 December 2018
Accepted date : 9 January 2019

DEED: Dynamic Energy-Efficient Data Offloading for IoT Applications under Unstable Channel Conditions

Hui Yan[a], Xiongtao Zhang[a], Huangke Chen[a], Yun Zhou[a], and Weidong Bao[a], Laurence T. Yang[b]

[a] *College of Systems Engineering, National University of Defense Technology, Changsha 410073, P. R. China*
{yanhui13, zhangxiongtao14, hkchen, zhouyun007, wdbao}@nudt.edu.cn
[b] *Department of Computer Science, Francis Xavier University Antigonish, NS, Canada*
ltyang@stfx.ca

Abstract

With the widespread use of Internet of Things (IoT) applications, the fast response and efficient data storage have been the main concerns of the service users and providers. Thus, data offloading has become a hotspot in both industry and academia, especially for real-time applications. To achieve efficient data offloading, a great number of in-depth studies have been conducted. Nevertheless, when addressing the issue of data offloading, few studies have taken into account the unstable channel conditions, which is however more practical and really needs more attention. In this paper, we consider the unstable channel state in the communication model. Based on this, we propose the task reliability model, the energy consumption model, and the device reliability model. From the perspective of optimizing energy consumption, we propose an optimal task scheduling model. Moreover, an innovative Dynamic Energy-Efficient Data offloading scheduling algorithm-*DEED* is proposed. The purpose of *DEED* is to as much as possibly reduce the energy consumption while ensuring the task reliability. To verify the effectiveness of the proposed *DEED*, extensive experiments are conducted to compare it with three comparison algorithms: *DRSD*, *DEPD*, and *DRPD*. The experimental results under different channel conditions demonstrate the superiority of the *DEED* in terms of the energy saving, reliability, and robustness.

Keywords: IoT, Data Offloading, Edge Computing, Energy-Efficient

1. Introduction

The development of Internet of Things (IoT) has been hailed as an unprecedented success. In the near future, tens of billions of IoT devices will be applied in homes, schools, companies, hospitals, etc. However, the processing capacity of IoT devices cannot totally guarantee the completion of tasks on time. Consequently, offloading tasks to the network edges and processing them in the edges has become a mainstream paradigm [7]. Therefore, Edge Computing (EC) [18], Mobile Edge Computing (MEC) [14], Mobile Cloud Computing (MCC) [20], Fog Computing (FC) [25] and other similar concepts have been proposed in recent years. To harvest the computing and storage resource of devices at the edge environment, both academia and industry have focused on the collaboration between edge network and IoT devices [27, 28]. Thus, data offloading becomes a critical technology for IoT applications, especially for applications running on mobile devices. Nowadays, data are regarded as one of the most promising resources, and lots of artificial intelligence systems need as many data as possible to improve its performance. Due to the lack of reliability and security of storing data on mobile devices, offloading data to the edge or data center becomes an important way to permanently store data.

In IoT applications, collaborative data offloading still faces many challenges. With the increase of application scenarios, IoT devices are going to be expected to perform more and more sophisticated tasks such as surveillance, crowdsensing, and health monitoring. However, the battery capacity of IoT devices are limited, and recharging or replacing its battery frequently is impractical in most instances. Besides, for mobile IoT devices, they are often used in the network where communication quality dynamically fluctuates, so data loss or data offloading failure is inevitable. As a service pattern, the success rate and response speed of data offloading directly affect the Quality of Service (QoS). To improve the QoS, collaborative data offloading is considered to be an effective method to reduce the communication overhead and energy consumption. Nevertheless, few works to date have studied the problem of collaborative data offloading at edge with efficient energy consumption and high reliability under the unstable channel conditions. For the challenges above, we focus on the collaborative data offloading with high reliability while optimizing the energy consumption under the unstable channel conditions. Based on the optimal scheduling model, we design an online scheduling algorithm-*DEED*, which can reduce the energy consumption as much as possible while ensuring the reliability of data offloading. The main contributions of this work are summarized as follows:

- We propose the overall framework of the mobile device that performs the data offloading. This framework details the inherent constraints and external constraints of mobile devices for the data offloading. Based on these constraints, the mobile device makes data offloading strategies (End-to-Cloud data offloading or End-to-End data offloading).
- This is an innovative work towards efficient energy consumption and high reliability for the collaborative data offloading under unstable channel conditions. We propose a heuristic algorithm that can reduce the energy consumption while ensuring task reliability.
- We propose a method to reduce the algorithm complexity which can improve the algorithm efficiency without impairing its performance almost. Through the algorithm complexity pruning method, the efficiency

of searching optimal strategy can be greatly raised. It is of great significance for applying the algorithm into practical applications.

- We conduct extensive simulation experiments. Compared with three comparison algorithms, the reliability, effectiveness, and robustness of the proposed algorithm are verified.

The rest of this paper is organized as follows. Section 2 gives a brief discussion on the related work, and Section 3 describes the problem formulation and basic models. We introduce the task scheduling model and algorithm in Section 4. Section 5 verifies the proposed algorithm through a series of simulation experiments. The conclusions and future work are given in Section 6.

2. Related Work

Data offloading is an important research topic in the mobile data management, and a large number of studies have focused on reducing energy consumption and task latency for data offloading both in industry and academia. In [13], Li et al. designed a novel offloading strategy to optimize the performance of IoT deep learning applications with edge computing. Moreover, to process the mobile data in real time, Li et al. achieved a fog computing based system in [16]. By offloading the computation instances from the central server to the fog nodes, the system can process more data with low latency. To jointly optimize the computation latency and energy consumption, minimizing the long-term average execution cost was widely studied. Xu et al. applied the in-memory storage and processing in the edge environment to reduce the long-term energy consumption while keeping the latency in an acceptable range [22]. Xie et al. proposed a light-weight and load-aware switch-to-controller selection scheme to cut the long-tail response latency for the edge environment in [21]. In [12], the energy-latency tradeoff in Mobile Edge Computing systems [11] with heterogeneous applications was investigated, including the non-offloadable workload, cloud-offloadable workload, and network traffic. Collectively, these studies have focused on optimizing the power consumption and latency of data offloading, but lacking the consideration of reliability.

The reliability for the mobile data management has soared much attention recently. Specifically, research on the reliability of mobile data management can be divided into two categories: reliability of mobile data processing and reliability of mobile data storage. On the one hand, mobile data are regarded as tasks and many papers have studied the optimization of task reliability for the mobile data processing. In [30], Zhu et al. proposed a fault-tolerant scheduling method for real-time scientific workflows, which ensured the reliability of tasks in case of hardware failures. In [26], Zhang et al. proposed a parallel task scheduling method to maximize reliability with energy conservation. Moreover, Li et al. proposed an algorithm that can improve the task reliability for precedence constrained stochastic tasks in [15]. On the other hand, some papers have studied the reliability of mobile data storage. In [9], Ding et al. presented a collaborative WiFi-based mobile data offloading architecture to enable reliable storage

ACCEPTED MANUSCRIPT

of data on smart phones. Wang et al. studied the impact of self-contention on mobile data storage and they proposed a method to optimize the upper bound of offloadings throughput in [24]. Nevertheless, these methods above were not adaptive for changeable edge environment because they did not take into account the change of channel state. In addition, a lot of studies assumed that the offloading time could be obtained before offloading the data, so as to reduce the model complexity. However, this assumption was too strict for practical applications, especially for those scenes with poor communication conditions.

In the edge environment, multi-device collaborative data offloading is more applicable. Thus, some studies have tried to solve the problem of collaborative data offloading. In [8], Ding et al. discussed the energy-aware collaborative data offloading and introduced the optimization model. Some papers applied the Utility theory to study the collaborative data offloading [10, 19]. They proposed that when and how to offload data depends on the utility. However, these papers did not take into account the inevitable unstable communication conditions in reality. Therefore, our work focuses on energy-efficient and reliability-aware data offloading of mobile devices in the edge environment while considering the unstable channel conditions.

3. Modeling and Problem Formulation

Fig. 1 shows the overview of the target mobile device. The main modules of the mobile device consist of Health Indicator, Communication Manager, Task Scheduler, and Offload Engine. The Health Indicator is responsible for indicating the state of mobile device, and the device state will be regarded as internal constraints for data offloading. Specifically, the Health Indicator can be divided into two parts: Reliability Estimator and Energy Monitor. The Energy Monitor extracts the remaining energy of the mobile device at a regular interval and sends this information to the Reliability Estimator. The Reliability Estimator calculates the device reliability for data offloading based on the energy and task information. Then, the device reliability will be fed back to the Task Scheduler and used as one of the constraints of data offloading. The Communication Manager is responsible for real-time monitoring and managing the dynamic communication network, and the communication network information will be treated as the external constraint for data offloading. The main functions of Communication Manager are topology discovery and channel monitoring. Topology discovery analyzes the topology of communication network among mobile devices based on the Zigbee protocol [3]. Channel monitoring monitors the channel status and communication rates between different mobile devices or between the device and the edge network. Without loss of generality, we regard the edge environment as the cloud. Combined with the external and internal constraints, the Task Scheduler determines the corresponding scheduling strategy of data offloading. The input data of the device includes two parts: one part is the data generated by the device itself, such as image captured by the device, user recording and so on. The other part is user data from other devices, and they request the device to assist them in offloading data to the cloud. For a device, we regard these two kinds of data originated from different sources as the input data. The device types are generally diverse in the practical applications, so we assume that data which will be offloaded do not have special requirements for the device type. For data offloading, we propose two kinds of offloading methods: end-to-cloud (D2C in short) data offloading and end-to-end (D2D in short) collaborative data offloading. The D2C data offloading means that the device directly offloads input data to the cloud, whereas the D2D data offloading means that device sends the input data to other devices and requests other devices to offload the data to the cloud. To facilitate the practical application, we assume that only the one-hop data transmission is considered between devices for collaborative data offloading. The Offload Engine is responsible for offloading data to the cloud or devices according to the scheduling strategy. Meanwhile, it monitors the status of data offloading. Once data are totally offloaded, it feedbacks task status to the Task Scheduler and adjusts resources.

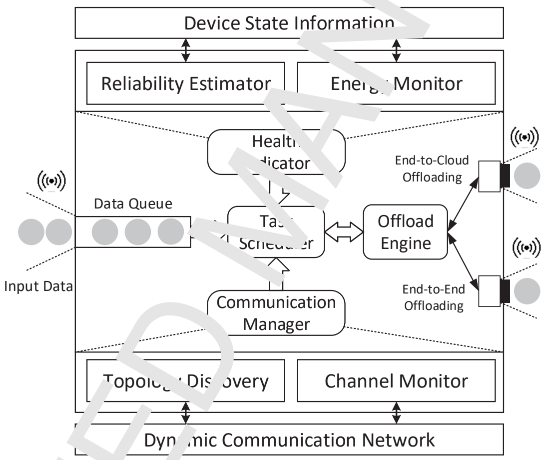


Figure 1: The overview of mobile devices

3.1. Preliminaries

This study focuses on the collaborative data offloading, which can be regarded as the issue of task scheduling. Thus, we define $T = \{t_1, t_2, \cdots, t_n\}$ as the task set that consists of n non-preemptive and independent tasks. With regard to a task $t_i \in T$, it is modeled as $t_i = (A_i, D_i, S_i)$ where A_i, D_i and S_i are t_i's arrival time, deadline, and task size. The task size S_i is measured by Bytes. We assume there is a device set with m mobile devices, $V = \{v_1, v_2, \cdots, v_m\}$, available in the application scene. Generally, mobile devices are in significant heterogeneity. So we define each device as $v_k = (R_k, M_k, W_k)$, where R_k, M_k, and W_k respectively represent the device reliability, the remaining storage capacity, and the channel state of the device v_k. For D2C data offloading, we define x_{ik}^c as the mapping indicator where $x_{ik}^c = 1$ denotes that task t_i is offloaded to cloud on mobile device v_k; otherwise, $x_{ik}^c = 0$. For D2D data offloading, we define x^d_{ikj} as the mapping indicator where $x^d_{ikj} = 1$ denotes that task t_i arriving at device v_k, is offloaded to cloud on mobile device v_j through collaborative data offloading; otherwise, $x^d_{ikj} = 0$.

3.2. Communication Model

The overview of the target communication framework is depicted in Fig. 2. To support the D2C data offloading and D2D collaborative data offloading, we propose two kinds of communication approaches on mobile devices: D2C communication and D2D communication. For D2D communication, we assume the data transmission and reception do not affect each other.

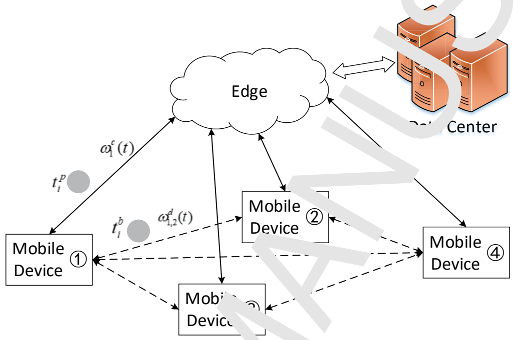


Figure 2: The target communication framework

3.2.1. Communication State Model

We define the channel state of D2C communication and D2D communication of device v_k as w^c_k and w^d_k respectively. Thus, the channel state of the device can be expressed as $W_k = \{w^c_k, w^d_k\}$. Typically, the communication channel state of the device is constantly changing, which leads the communication rate of the device to fluctuate at all time. We assume that there are two channel state sets $\Omega^c = \{\Omega^c_1, \Omega^c_2, \cdots, \Omega^c_N\}$ and $\Omega^d = \{\Omega^d_1, \Omega^d_2, \cdots, \Omega^d_M\}$, which represent all kinds of the channel state of D2C communication and D2D communication, respectively. We assume the channel state sets satisfy the following order $\Omega^c_1 \prec \Omega^c_2 \prec \cdots \prec \Omega^c_N$ $(\Omega^d_1 \prec \Omega^d_2 \prec \cdots \prec \Omega^d_M)$. Different channel state leads to different communication rate r^c_k (r^d_k) and energy consumption p^c_k (p^d_k). The worse the channel quality, the lower the communication rate but the higher the energy consumption [4, 19]. For example, if the channel state of a device is Ω^c_1, then it has the lowest communication rate and the highest energy consumption.

For D2C communication, since end and cloud are in peer-to-peer connection, w^c_k can be expressed as follows:

$$w^c_k = \Omega^c_i, i \in \{1, 2, \cdots, N\}. \tag{1}$$

For D2D communication, however, each mobile device may be connected to one or more mobile devices due to different communication network topologies. We use $e(i,j), (i \neq j)$ to characterize the edge between device v_i and v_j, where $e(i,j) = 1$ denotes that device v_i can communicate with device v_j; otherwise,

$e(i,j) = 0$. Hence, we define the topology related device set of v_k as Vt_k which can be expressed as $Vt_k = \{v_j \mid e(k,j) = 1, v_j \in V\}$. Besides, because of the changeable topology, w_k^d is represented as a dynamic set:

$$w_k^d = \{w_{kj}^d \mid w_{kj}^d = \Omega_i^d, v_j \in Vt_k\}, i \in \{1, 2, \cdots, M\}. \tag{2}$$

3.2.2. Communication Rate Estimation Model

Data offloading usually adopts wireless communication methods such as Wi-Fi, Bluetooth, 4G, or Zigbee. These approaches have a common characteristic that the communication rate exists the upper limit. Besides, the actual communication rate is often far less than the rated speed, but there is a close relationship between the communication rate and the channel state. Therefore, we define $r_k^c = f_c(w_k^c)$ and $r_k^d = f_d(w_k^d)$. For a device, both the external and internal constraints will affect the communication rate; it is hard to estimate the communication rate for a single task. As such, static estimation of the communication rate has mainly been adopted in previous work to simplify the problem complexity [4, 6, 29]. However, the difference between this estimated and the actual value significantly affects the quality of task scheduling. Drawing on the methods of [5, 23], we propose an effective communication rate estimation approach for data offloading by considering the following factors:

Factor 1: We assume that the rated communication rate is the ideal communication rate $\widehat{r_k^c}$ under the ideal channel state $\widehat{w_k^c}$.

Factor 2: Because w_k^c often cannot reach the ideal channel state $\widehat{w_k^c}$, we define the relative ratio of the actual communication rate r_k^c to the rated communication rate $\widehat{r_k^c}$ as $\theta_k^c = \frac{r_k^c}{\widehat{r_k^c}} = \frac{w_k^c}{\widehat{w_k^c}}$. Then, the actual static communication rate can be expressed as $r_k^c = \theta_k^c \widehat{r_k^c}$.

Factor 3: Considering that θ_k^c is closely related to r_k^c, we characterize r_k^c by θ_k^c, which is assumed to follow the Beta distribution $\theta_k^c \sim Beta(\alpha_k^c, \beta_k^c)$. The probability density function (Pdf in short) of θ_k^c can be expressed as follows:

$$f_\Theta(\theta_k^c) = \begin{cases} \frac{\Gamma(\alpha_k^c+\beta_k^c)}{\Gamma(\alpha_k^c)\,\Gamma(\beta_k^c)} (\theta_k^c)^{\alpha_k^c - 1} (1-\theta_k^c)^{\beta_k^c - 1}, & \theta_k^c \in (0,1); \\ 0, & \theta_k^c \notin (0,1). \end{cases} \tag{3}$$

Factor 4: Combined with Eq.(3), it can be inferred that r_k^c also follows the similar Beta distribution with the same α_k^c and β_k^c. It is worth noting that r_k^d can also be modeled as a Beta distribution $Beta(\alpha_k^d, \beta_k^d)$ in the same way.

3.3. Data Offloading Reliability Model

Since the failure of one device will cause all tasks on it to fail, tasks need to be backed up in multiple duplicates to ensure the reliability of the data. However, due to the different reliability requirements of tasks, the number of backups for each task is varied. To schedule more tasks while ensuring the task reliability within the limited battery capacity, we analyze the data offloading reliability in this section. The D2C data offloading and D2D collaborative data offloading are comprehensively adopted in this paper, so we analyze them respectively.

3.3.1. Reliability of End-to-Cloud Data Offloading

The D2C data offloading means mobile devices use the D2C communication approach to offload data on itself to the cloud. To analyze the reliability of D2C data offloading, we introduce following definitions.

Definition 1. Earliest Start Time EST^c_{ik}: *For a task t_i, the earliest start time of itself on device v_k for D2C data offloading is the earliest time when t_i can be offloaded, which is determined by following expression*:

$$EST^c_{ik} = MAX\{AT^c_{ik}, A_i\}, \tag{4}$$

where AT^c_{ik} is the available time of D2C data offloading for task t_i on v_k.

Definition 2. Expected Offloading Time EOT^c_{ik}: *For a task t_i, the expected time usage on device v_k for D2C data offloading is defined as expected offloading time EOT^c_{ik}.*

As analyzed in **Factor 4**, r^c_k can be modeled as a Beta distribution $r^c_k \sim Beta(\alpha^c_k, \beta^c_k)$. The expected communication rate of r^c_k is $E(r^c_k) = \dfrac{\alpha^c_k}{\alpha^c_k + \beta^c_k}\widehat{r^c_k}$. Thus, EOT^c_{ik} can be expressed as follows:

$$EOT^c_{ik} = \frac{S_i}{E(r^c_k)}. \tag{5}$$

Definition 3. Available Offloading Time AOT^c_{ik}: *For a task t_i, the available offloading time on device v_k for D2C data offloading is determined by the following expression*:

$$AOT^c_{ik} = D_i - EST^c_{ik}. \tag{6}$$

The available offloading time AOT^c_{ik} normally can be used to measure the reliability of tasks. We define the baseline offloading time as $\Delta^c_{ik} = \frac{S_i}{\widehat{r^c_k}}$, which represents the shortest time for offloading data t_i. Based on that, we introduce the **Theorem 1**.

Theorem 1. *If the Pdf of θ^c_k is denoted as $f_\Theta(\theta^c_k)$, then the Pdf of task offloading time OT^c_{ik} by the D2C approach can be expressed as following function*:

$$f_{OT}(OT^c_{ik}) = \begin{cases} f_\Theta(\frac{\Delta^c_{ik}}{OT^c_{ik}})\,\frac{\Delta^c_{ik}}{(OT^c_{ik})^2}, & OT^c_{ik} \geq \Delta^c_{ik}; \\ 0, & OT^c_{ik} < \Delta^c_{ik}. \end{cases} \tag{7}$$

Proof. As defined in **Factor 2**, $r^c_k = \theta^c_k \widehat{r^c_k}$ is formed. Combining with $OT^c_{ik} = \frac{S_i}{r^c_k}$, we can drive that $\theta^c_k = \frac{S_i}{\widehat{r^c_k}\,OT^c_{ik}}$, that is $\theta^c_k = \frac{\Delta^c_{ik}}{OT^c_{ik}}$. So the derivative of the θ^c_k versus OT^c_{ik} is $\frac{\mathrm{d}\theta^c_k}{\mathrm{d}OT^c_{ik}} = -\frac{\Delta^c_{ik}}{(OT^c_{ik})^2}$. As mentioned in [17], if the Pdf of a random variable X is given as $f_X(x)$ and given a new variable $Y = g(X)$ while its function g is monotonic, then the compound Pdf is $f_Y(y) = |\frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y}|\, f_X(g^{-1}(y))$, where $g^{-1}(y)$ denotes the inverse function. So applying θ^c_k and $\frac{\mathrm{d}\theta^c_k}{\mathrm{d}OT^c_{ik}}$ to the function, the theorem is approved. □

The probability of which task t_i is completed on device v_k before its deadline D_i is the reliability of data offloading in essence, and we define it as R^c_{ik}. We denote the cumulative distribution function (Cdf in short) of OT^c_{ik} as $F_T(OT^c_{ik})$. Based on that, the **Theorem 2** is proposed as follows.

Theorem 2. *If the Cdf of θ^c_k is denoted as $F_\Theta(\theta^c_k)$, then the reliability R^c_{ik} for D2C data offloading can be expressed as follows:*

$$R^c_{ik} = 1 - F_\Theta(\frac{\Delta^c_{ik}}{AOT^c_{ik}}). \tag{8}$$

Proof. The detailed inducement is as follows:

$$\begin{aligned}
R^c_{ik} &= F_T(AOT^c_{ik}) \\
&= \int_{\Delta^c_{ik}}^{AOT^c_{ik}} f_T(t)\,dt \\
&= \int_{\Delta^c_{ik}}^{AOT^c_{ik}} f_\Theta(\frac{\Delta^c_{ik}}{t})\frac{\Delta^c_{ik}}{t^2}\,dt \\
&= -F_\Theta(\frac{\Delta^c_{ik}}{AOT^c_{ik}}) + F_\Theta(1) \\
&= 1 - F_\Theta(\frac{\Delta^c_{ik}}{AOT^c_{ik}}).
\end{aligned}$$

□

3.3.2. Reliability of End-to-End Collaborative Data Offloading

The D2D collaborative data offloading means that the mobile device offloads its data to the nearby device, and requests it to eventually offload data to the cloud. This method can effectively improve the robustness of data offloading. To facilitate the analysis, we assume that device v_k can communicate with device v_j, and v_k requests v_j to assist it in collaboratively offloading data t_i. In essence, this approach can be broken down into two phases: D2D data transmission and D2C data offloading.

For the first phase, the main factors affecting data transmission include the data size and the D2D communication channel quality. To analyze the reliability of D2D data offloading, we introduce following definitions.

Definition 4. Earliest Start Time EST^t_{ik}: *For a task t_i, the earliest start time of itself on device v_k for D2D data transmission is defined as the earliest time when t_i can be transmitted, which is determined by following expression:*

$$EST^t_{ik} = MAX\{AT^t_{ik}, A_i\}, \tag{9}$$

where AT^t_{ik} is the available time of D2D data transmission for task t_i on v_k.

Definition 5. Expected Transmission Time ETT^t_{ikj}: *For a task t_i, the expected time usage of D2D data transmission from device v_k to device v_j is defined as the expected transmission time ETT^t_{ikj}.*

As analyzed in **Factor 4**, r^d_{kj} can be modeled as a Beta distribution $r^d_{kj} \sim Beta(\alpha^d_{kj}, \beta^d_{kj})$, so ETT^t_{ikj} can be expressed as following equation:

$$ETT_{ikj}^{t} = \frac{S_i}{E(r_{kj}^{d})}. \tag{10}$$

Definition 6. Expected Finish Time EFT_{ikj}^{t}: *For a task t_i, the expected finish time of D2D data transmission from device v_k to device v_j is defined as EFT_{ikj}^{t}, which can be expressed as follows*:

$$EFT_{ikj}^{t} = EST_{ik}^{t} + ETT_{ikj}^{t}. \tag{11}$$

For the second stage, the method of reliability assessment for D2C offloading is the same as the **Section 3.3.1**, except for the task arrival time changes from A_i to EFT_{ikj}^{t}. For example, the initial arrival time and deadline of task t_i on device v_k are A_i and D_i, respectively. If t_i is transmitted from device v_k to v_j, the arrival time of t_i for v_j can be regarded as EFT_{ikj}^{t}, but its deadline is unchanged. We denotes it as $t_i^b = (EFT_{ikj}^{t}, D_i, C_i)$. For v_j, it can be regarded as a new arrival task. Then, we can get the task reliability R_{ikj}^{d} of task t_i^b according to **Theorem 2**.

To ensure the reliability of data offloading, task t_i may be offloaded on multiple devices. We denote the union reliability of multiple data offloading as $\bigcup(R_{ik}^{*})$, and it can be expressed as follows:

$$\bigcup(R_{ik}^{*}) = 1 - (1 - R_{ik}^{c}\, x_{ik}^{c}) \prod_{\substack{j=1 \\ j \neq k}}^{m} (1 - R_{ikj}^{d}\, x_{ikj}^{d}). \tag{12}$$

3.4. Energy Consumption Model

Different channel state leads to different energy consumption power p_k^c (p_k^d). The worse the channel quality, the higher the energy consumption power. We assume that there is an ideal energy consumption power $\widehat{p_k^c}$ under the ideal channel state $\widehat{w_k^c}$. The relative ratio of the actual energy consumption power p_k^c to $\widehat{p_k^c}$ is regarded as $\varepsilon_k^c = \frac{p_k^c}{\widehat{p_k^c}}$. So the actual energy consumption power can be expressed as $p_k^c = \varepsilon_k^c\, \widehat{p_k^c}$. Similarly, $p_k^d = \varepsilon_k^d\, \widehat{p_k^d}$ can be derived. We introduce the energy consumption model from two aspects: energy consumption of D2C data offloading and energy consumption of D2D collaborative data offloading.

For the D2C data offloading, we define the energy consumption as E_{ik}^{c}, and it is closely related to the expected offloading time EOT_{ik}^{c} and the energy consumption power p_k^c. We represent it as following equation:

$$E_{ik}^{c} = p_k^c\, EOT_{ik}^{c}. \tag{13}$$

We define the energy consumption of D2D collaborative data offloading as E_{ikj}^{d}. As mentioned above, the D2D collaborative data offloading can be divided into two phases, so E_{ikj}^{d} is composed of two parts: the energy consumption of D2D data transmission E_{ikj}^{t} and the energy consumption of D2C data offloading E_{ij}^{c}. Specifically, E_{ij}^{c} can be calculated by Eq.13 and E_{ikj}^{t} can be expressed as following equation:

$$E_{ikj}^{t} = p_{kj}^{d}\, ETT_{ikj}^{t}. \tag{14}$$

Thus, $E^d_{ikj} = E^t_{ikj} + E^c_{ij}$ is formed.

3.5. Device Reliability Model

Data offloading is not only related to the task itself, but also to the reliability of device. The failure of device during the data offloading will also cause the failure of tasks. We denote the device reliability for task t_i on device v_k as R_{ik}. To reduce the overhead of device reliability estimation, we assume that the device battery life follows the Normal distribution. We indicate the random variable of device power as E_k. Thus, the Pdf of E_k can expressed as follows:

$$f(E_k) = \frac{1}{\sqrt{2\pi}\,\sigma_k}\, exp(-\frac{(E_k - \mu_k)^2}{2\sigma_k^2}), \tag{15}$$

where μ_k and σ_k are the mean and variance of E_k, respectively.

We indicate the amount of power has been consumed as Ec_k and the amount of power needed for data offloading as Ed_k, so the device reliability R_{ik} can be deduced as follows:

$$R_{ik} = 1 - F(E_k - Ec_k < Ed_k). \tag{16}$$

4. Optimal Task Scheduling Model and Algorithm

The key issue of data offloading is to allocate tasks to mobile devices so that the task deadline can be reached. However, due to the mobility of devices and the unstable channel conditions, it is a great challenge to solve the problem precisely. Therefore, we propose a task scheduling optimization model based on device state and channel state.

4.1. Optimal Task Scheduling Model

For mobile devices, subject to the limitation of battery capacity, energy consumption has an essential effect for scheduling task. Thus, we take the energy consumption as the main objective of the collaborative data offloading. We denote the energy-consumption minimization problem as an Integer Linear Programming problem (ILP in short):

$$min : E^c_{ik}\, x^c_{ik} + \sum_{\substack{j=1 \\ j \neq k}}^{m} E^d_{ikj}\, x^d_{ikj} \tag{17}$$

$$s.t. \bigcup (R^*_{ik}) \geq \widehat{T_i}, \tag{18}$$

$$R^*_{ik} \geq \widehat{R_i}, \tag{19}$$

$$R_{ik}\, x_{ik} \geq \widehat{R_k}\, x_{ik}, \forall v_k \in V, \tag{20}$$

$$M_{ik} - S_i\, x_{ik} \geq \widehat{M_k}\, x_{ik}, \forall v_k \in V, \tag{21}$$

$$x^c_{ik}, x^d_{ikj} \in \{0, 1\}, \tag{22}$$

where $x_{ik} = MAX\{x^c_{ik}, x^d_{ikj}\}$ represents that task t_i is ultimately offloaded on device v_k.

The optimal objective of the ILP problem is to minimum the total energy consumption for D2C and D2D data offloading as shown in function 17. The first constraint (Eq.18) indicates that the union reliability of collaborative data offloading must be greater than the lowest union reliability threshold $\acute{T}_i$. Since the failure of data offloading not only wastes resources but also affects other tasks, so we set the minimum task reliability threshold $\widehat{R_i}$ as shown in the second constraint (Eq.19). The third constraint (Eq.20) represents the reliability of device v_k on which data are offloaded must be greater than the device reliability threshold $\widehat{R_k}$. The fourth constraint (Eq.21) indicates that the remain storage of the device M_{ik} for offloading task t_i must be greater than the storage threshold $\widehat{M_k}$. The last constraint (Eq.22) specifies the range of x^c_{ik} and x^d_{ikj}. Obviously, the optimal scheduling strategy under current device and channel state can be obtained by solving the ILP problem.

4.2. Algorithm Complexity Pruning

For the scenario with few devices, the ILP problem can be solved directly by the Enumeration or Implicit Enumeration within acceptable overhead. However, as the device number increases, the complexity of searching the optimal solution will increase exponentially. To reduce the overhead, we propose the following complexity pruning method. Firstly, we classify all constraints into three categories: device constraints (Eq.20, Eq.21), task constraints (Eq.18, Eq.19) and range of solutions (Eq.22). We define device constraints as hard constraints, because the scheduling method will be the feasible solution only if all device constraints are satisfied at the same time. However, task constraints are usually affected by multiple devices, so we define the task constraints as soft constraints. Based on the definitions above, we propose to reduce the complexity through two stages. For the first stage, we apply the hard constraints to narrow the feasible solution range. For the second stage, we use the relative utility to accelerate problem solving under the soft constraints.

For the hard constraints, the Branch and Bound method [2] is used to eliminate inferior solutions. For $\forall v_k \in V$, we assume that $x_{ik} = 1$, then we investigate whether device v_k satisfies the hard constraints. Specifically, we first check whether Eq.20 and Eq.21 are both satisfied. If they are satisfied, we regard v_k as the pending solution and let it enter the pending solution set Vp_k; otherwise, it will be eliminated.

For the soft constraints, Eq.19 has an important impact on Eq.18, so we first verify whether Eq.19 is satisfied. If the constraint is satisfied, then we calculate the union reliability $\bigcup(R^*_{ik})$ according to Eq.12. As the union reliability increases, the total energy consumption will inevitably increase. However, our objective is to reduce energy consumption as much as possible while ensuring the task reliability. Thus, we propose a novel approach to search the optimal solution. Concretely, we define the ratio of task reliability to energy consumption for the device as I^*_{ik}, which is expressed as following function:

$$I^*_{ik} = \begin{cases} \frac{R^c_{ik}}{E^c_{ik}}, & x^c_{ik} = 1; \\ \frac{R^d_{ikj}}{E^d_{ikj}}, & x^d_{ikj} = 1. \end{cases} \tag{23}$$

Obviously, I^*_{ik} represents the reliability utility of the scheduling strategy under unit energy consumption. The greater the I^*_{ik}, the higher the reliability utility under unit energy consumption. Hence, we calculate the reliability utility I^*_{ik} for all pending solutions in Vp_k and sort them in descending order by the reliability utility. Then, we progressively traverse the device in Vp_k for data offloading. Once the Eq.18 and Eq.19 are satisfied at the same time, the solution is the optimal solution for the problem. Based on the two stages above, searching the optimal solution does not need to traverse the whole pending solutions, so the algorithm complexity can be significantly reduced.

4.3. Dynamic Energy-Efficient Data Offloading Algorithm-DEED

Based on the data offloading mechanisms discussed above, we design an innovative Dynamic Energy-Efficient Data offloading scheduling algorithm-*DEED* which takes into account the unstable communication state. *DEED* uses heuristic approaches to optimize the energy consumption while ensuring the task reliability. For data offloading, the First Come First Service principle is adopted.

Algorithm 1 specifies how the optimal scheduling strategy is selected. It firstly updates the topology and the channel state of v_k. Then it determines the topology related device set Vt_k (see line 3). To reduce the complexity, it selects devices that satisfy hard constraints (see lines 4-6) and determines the pending device set Vp_k. In the pending device set Vp_k, the algorithm respectively calculates the parameters of tasks for D2C data offloading (see lines 8-14) and D2D data offloading (see lines 15-23). The algorithm sorts the Vp_k according to I^*_{ik} (see line 24). Finally, the algorithm successively verifies whether the Eq.18 and Eq.19 are all satisfied in the Vp_k. Once they are satisfied, the optimal solution Sv^*_i is obtained (see lines 25-33). According to Sv^*_i, the Offload Engine, as shown in the Fig. 1, allocates the task t_i to the corresponding device so that the optimal data offloading is achieved.

4.4. Algorithm Complexity Analysis

We analyze the time complexity of *DEED* in this section. Totally, the time complexity of *DEED* is determined by the device constraints and the task constraints, respectively. To facilitate the analysis, we first make following assumptions: (1) There are M elements in the topology related device set Vt_k; (2) There are N elements in the pending device set Vp_k, where $0 \leq N \leq M$. Based on the assumptions above, we introduce the **Theorem 3**.

Theorem 3. *The time complexity of DEED is* $\mathrm{O}(M + N^2)$.

Proof. To determine the pending device set Vp_k, all devices belonging to the topology related device set Vt_k need to be checked according to the device constraints, so the time complexity is $\mathrm{O}(M)$. Once the related device set Vt_k is determined, it needs to use the complexity of $\mathrm{O}(N)$ to calculate the I^*_{ik} of all device, and the time complexity becomes $\mathrm{O}(M + N)$. Then the algorithm takes the Bubble Sort method [1] to sort the Vp_k in the descending order of I^*_{ik}, which will lead to the time complexity of $\mathrm{O}(N^2)$. Thus, the overall time complexity is $\mathrm{O}(M + N^2)$. To obtain the optimal strategy, the task constraints should be tested, which will cause the time complexity of $\mathrm{O}(N)$. However, $\mathrm{O}(N)$ is an

Algorithm 1: Optimal scheduling strategy selection

$Vt_k \leftarrow null$;$Vp_k \leftarrow null$;$Sv_i^* \leftarrow null$;$\cup(R_{ik}^*) \leftarrow 0$;$I_{ik}^* \leftarrow null$;
while *task* t_i *arrives at device* v_k **do**
    Update the topology Vt_k *and channel state* W_k *of* v_k;
    foreach $v_j \in Vt_k \cup v_k$ **do**
        if $R_{ij} \geq \widehat{R_j}$ && $M_{ij} - S_i \geq \widehat{M_k}$ **then**
            $Vp_k \leftarrow Vp_k \cup v_j$;
    foreach $v_j \in Vp_k$ **do**
        if $v_j == v_k$ **then**
            Update the available time AT_{ik}^c;
            Estimate the D2C communication rate r_k^c;
            Calculate EST_{ik}^c, EOT_{ik}^c, AOT_{ik}^c *based on Eq.4, 5, 6*;
            Calculate R_{ik}^c *according to* ***Theorem** 2*;
            $E_{ik}^c \leftarrow p_k^c\, EOT_{ik}^c$;
            $I_{ik}^* \leftarrow I_{ik}^* \cup \frac{R_{ik}^c}{E_{ik}^c}$;
        else
            Update the available time AT_{ik}^t;
            Estimate the D2D communication rate r_{kj}^d;
            Calculate EST_{ik}^t, ETT_{ikj}^t, EFT_{ikj}^t *based on Eq.9, 10, 11*;
            $t_i^b \leftarrow (EFT_{ikj}^t, D_i, S_i)$;
            Calculate EST_{ij}^c, EOT_{ij}^c, AOT_{ij}^c *based on Eq.4, 5, 6*;
            Calculate R_{ikj}^d *according to* ***Theorem** 2*;
            $E_{ikj}^d \leftarrow p_{kj}^d\, ETT_{ikj}^t + p_j^c\, EOT_{ij}^c$;
            $I_{ik}^* \leftarrow I_{ik}^* \cup \frac{R_{ikj}^d}{E_{ikj}^d}$;
    Sort Vp_k *in descending order according to* I_{ik}^*;
    foreach $v_j \in Vp_k$ **do**
        if $v_j == v_k$ **then**
            $\cup(R_{ik}^*) \leftarrow 1 - (1 - \cup(R_{ik}^*))(1 - R_{ik}^c))$;
        else
            $\cup(R_{ik}^*) \leftarrow 1 - (1 - \cup(R_{ik}^*))(1 - R_{ikj}^d))$;
        $Sv_i^* \leftarrow Sv_i^* \cup v_j$;
        if *Eq.18 and Eq.19 are all satisfied* **then**
            return Sv_i^*;
            break;

order of magnitude lower than $O(N^2)$, so the overall time complexity is not affected. Totally, the time complexity of *DEED* is $O(M + N^2)$. $\square$

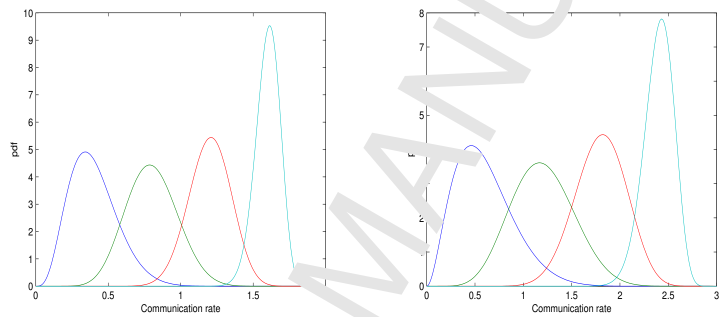
5. Performance Evaluation

In order to verify the performance of *DEED*, we quantitatively compare *DEED* with three comparison algorithms: Dynamic Random Selection Data offloading scheduling algorithm-*DRSD*, Dynamic Energy Priority Data offloading scheduling algorithm-*DEPD*, and Dynamic Reliability Priority Data offloading scheduling algorithm-*DRPD*. The three comparison algorithms are briefly described as follows:

- *DRSD* adopts the random scheduling strategy. The algorithm randomly selects a device as the target device, and then verifies whether the device constraints (Eq.20, Eq.21) are satisfied. If the device constraints are not satisfied, the device can not be used for task scheduling; otherwise, the algorithm calculates the union reliability and task reliability according to the idea of *DEED*. Once the task constraints (Eq.18, Eq.19) are satisfied, the optimal solution is found.
- *DEPD* is a variant of *DEED*, but it adopts the greedy scheduling strategy. The difference between *DEPD* and *DEED* is that *DEPD* considers energy consumption priority in task scheduling. The algorithm first finds the pending device set in the same way as *DEED*, but it sorts the pending device set in ascending order of energy consumption. Then the algorithm calculates the union reliability and task reliability in the same way with *DEED*. Once the task constraints (Eq.18, Eq.19) are satisfied, the optimal solution is found.
- *DRPD* is also a variant of *DEED*, but the greedy scheduling strategy is adopted. The difference between *DRPD* and *DEED* is that *DRPD* considers reliability priority in task scheduling. The algorithm first finds the pending device set in the same way as *DEED*, but it sorts the pending device set in descending order of task reliability. Then the algorithm calculates the union reliability and task reliability in the same way with *DEED*. Once the task constraints (Eq.18, Eq.19) are satisfied, the optimal solution is found.

5.1. Experimental Setup

Through extensive experiments, we find that these algorithms are not sensitive to the union reliability threshold $\widehat{T_i}$, task reliability threshold $\widehat{R_i}$ and host reliability threshold $\widehat{R_k}$ which are mentioned in Section 4.1. Therefore, we set $\widehat{T_i} = 0.75$, $\widehat{R_i} = 0.7$, and $\widehat{R_k} = 0.7$. To reflect the heterogeneity of mobile devices, we divide mobile devices into four types. The storage of each device type is set to 500MB, 1000MB, 1500MB, 2000MB. As mentioned in Section 3.5, the device power follows the Normal distribution with different μ_k and σ_k. Thus, we set μ_k as 1000mAh, 1500mAh, 2000mAh and 2500mAh, and we respectively set σ_k as 5mAh, 7.5mAh, 10mAh and 12.5mAh. To reflect the diversity of channel states, we assume that there are four channel states for D2C and D2D communication, that is, $\Omega^c = \{2,4,6,8\}$ and $\Omega^d = \{2,4,6,8\}$, respectively. We assume that the ideal channel state $\widehat{w_k^c} = 10$, and the corresponding ideal communication rate $\widehat{r_k^c} = 2MB/s$ and corresponding communication power $\widehat{p_k^c} = 0.1mAh/s$, respectively. Similarly, we set the ideal channel state $\widehat{w_k^d} = 10$, and the corresponding ideal communication rate $\widehat{r_k^d} = 3MB/s$ and corresponding communication power $\widehat{p_k^d} = 0.12mAh/s$, respectively. As analyzed in Section 3.2.2, r_k^c and r_k^d are characterized by Beta distribution. For D2C communication, we set β_k^c as 18, and we set α_k^c as 4.5, 12, 27, and 72 for different channel states respectively. For D2D communication, we set β_k^d as 12, and we set α_k^d as 3, 8, 18, and 48 for different channel states respectively. The Pdfs of the D2C and D2D communication rate are shown in Fig. 3.



(a) D2C

(b) D2D

Figure 3: The Pdf of the communication rate

As shown in Fig. 3, the Beta distribution has the similar shape like the Normal distribution, but it has the communication rate upper and lower bound, which is more realistic. Besides, the given parameters above can ensure different channel states correspond to different communication rates. The worse the channel state, the larger the variance of the distribution, indicating the lower the stability of the communication rate.

Based on the experimental setup, we compare the performance of these algorithms on following metrics:

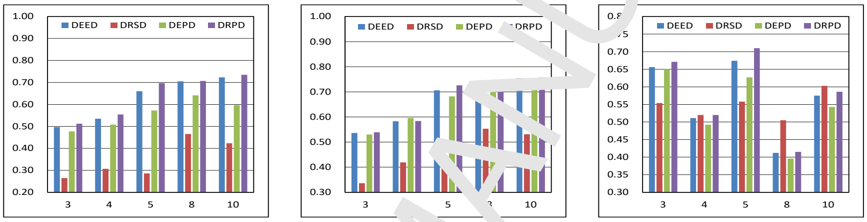
- Task Completion Ratio (TCR): TCR is defined as the percentage of tasks that are finished before deadlines among all tasks, which represents the reliability of algorithm.
- Task Acceptance Ratio (TAR): TAR represents the percentage of tasks that are accepted among all tasks, which represents the robustness of algorithm.
- Ratio of Task time over Hosts time (RTH): RTH is defined as the ratio of total task execution time over the total active time of devices, which represents the resource utilization of algorithm.

5.2. *Experiments and Analysis*

On the whole, the topology and communication quality are constantly changing, but they can be regarded as the static state for a short period of time. Therefore, to demonstrate the performance of these algorithms, we divide experiments into two parts: experiments under static communication conditions and experiments under dynamic communication conditions.

5.2.1. *Experiments under Static Communication Conditions*

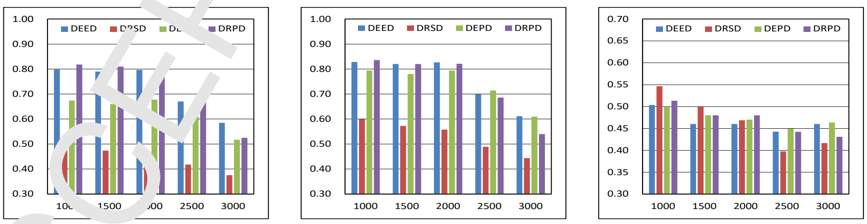
For static communication conditions, many factors affect the performance of the algorithms, for instance, device number, task number, task size, task arrival rate, task deadline urgency, and the probability of device communication. By analyzing the impact of these factors, the algorithm performance under the static communication conditions can be effectively verified.



(a) TCR (b) TAR (c) RTH

Figure 4: The impact of the device number

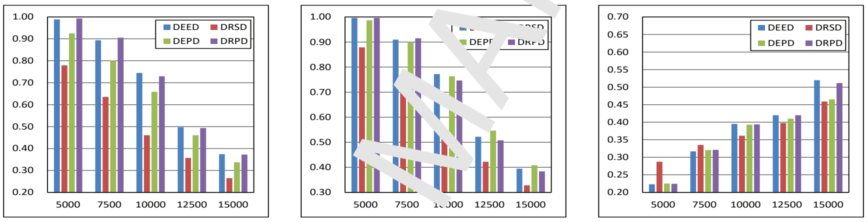
The experimental results in Fig. 4 specify that the device number is closely related to algorithm performance. As the device number increases, TCR and TAR increase therewith, but there is no obvious increasing trend of RTH. When the device number increases, the number of devices which are available for offloading data increase correspondingly, so TCR and TAR increase. However, the increase of device number will also cause the device to be idle, so RTH does not increase significantly. It is worth noting that when the device number is greater than 8, the increment of device number is not meaningful for improving TCR and TAR. Totally, *DEED* and *DRPD* have better performance.



(a) TCR (b) TAR (c) RTH

Figure 5: The impact of the task number

Fig. 5 shows the relationship between the task number and the algorithm performance. As the task number increases, TCR and TAR remain unchanged firstly, but when the task number is greater than 2000, TCR and TAR decrease rapidly. Nevertheless, RTH shows the opposite trend that it firstly drops and then stays steady when the task number is greater than 2000. By analyzing the log files, we find that when the number of tasks is about 2500, the battery capacity of device will be insufficient for data offloading. Many tasks are rejected because the device reliability is not satisfied, which results in the rapid decline of TCR and TAR. However, the increase of task number will result in devices with poor channel state being idle for a long time, which causes the RTH to drop. When task number is greater than 2,500, since all devices have insufficient power for data offloading, almost all devices are in idle state, so RTH remains stable. With the increase of task number, the performance of *DEED* gradually exceeds *DRPD*. As the task number increases, the device worktime increases accordingly, so the scarcity of battery capacity becomes the main limitation of data offloading. Because *DEED* takes into account energy optimization, it shows the best performance.

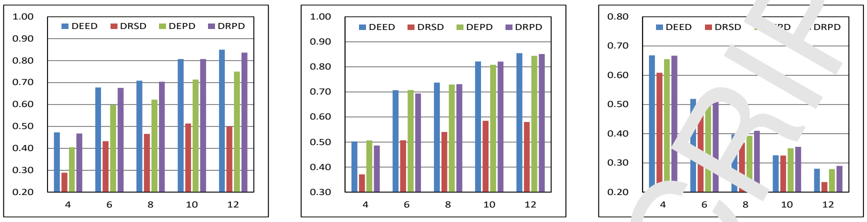


(a) TCR (b) TAR (c) RTH

Figure 6: The impact of the task size

As shown in Fig. 6, task size has a significant impact on algorithm performance. We set the lower limit of the task size to 1000KB. When the upper limit of the task size is 5000KB, all algorithms show good performance on TCR and TAR. As the task size increases, TCR and TAR decrease rapidly, but RTH shows the increasing trend. When the task size becomes larger, the offloading time of the task is increasing, which results in the decrease of TAR and TCR, and leads to the increase of RTH under the same task arrival rate. In general, *DEED* has the best performance with the increase of task size. It is worth noting that when the upper limit of the task size is 5000KB, the RTH of *DRSD* is significantly higher than other algorithms. This phenomenon does not mean that the resource utilization of *DRSD* is higher. On the contrary, it indicates that the random selection strategy leads to excessive resource waste.

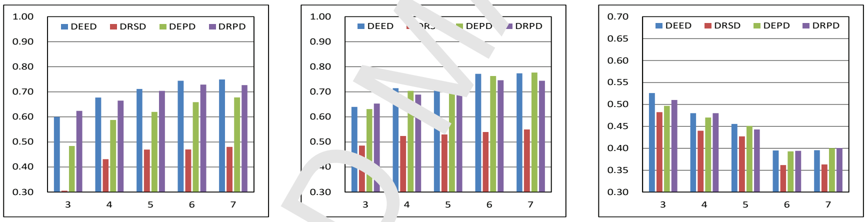
The impact of the task arrival rate for each algorithm is shown in Fig. 7. We assume that the task arrival follows the Poisson distribution and the X-axis represents the average task arrival interval. The smaller the task arrival interval, the higher the task arrival rate. As the task arrival interval grows, the TCR and TAR become larger, but RTH becomes smaller. This is because when the task



(a) TCR (b) TAR (c) RTH

Figure 7: The impact of the task arrival rate

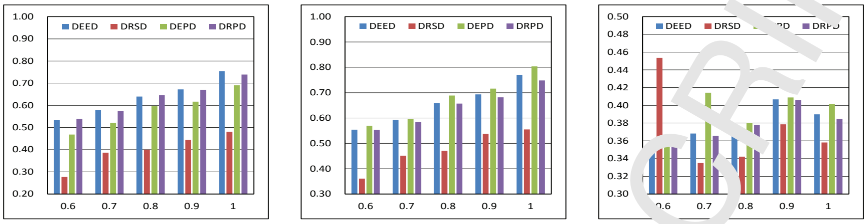
arrival interval becomes larger, the device has enough time to offload the data, making TCR and TAR larger, but RTH smaller. When task arrival interval is small, *DRPD* shows the best performance; *DRPD* adopts the reliability priority strategy, which can guarantee higher reliability for data offloading when task arrival rate is high. When task arrival interval is large, *DEED* shows the best performance; *DEED* comprehensively considers the energy consumption and reliability, which can extend the worktime of mobile devices when task arrival rate is low.



(a) TCR (b) TAR (c) RTH

Figure 8: The impact of the task deadline urgency

Fig. 8 depicts the relationship between deadline urgency and algorithm performance. The X-axis represents the deadline urgency of the task. The higher the deadline urgency, the smaller the TCR and TAR, but the larger the RTH. As the deadline urgency decreases, TCR and TAR gradually increase, but RTH gradually decreases. This is because when the deadline is tight, the device will be very busy. It causes many tasks to dissatisfy the optimization model constraints, so the TCR and TAR are smaller, but the RTH is larger. However, as the urgency decreases, there will be more idle devices to offload data, so TCR and TAR increase, but RTH decreases. Deadline urgency has a great influence on *DRSD*. When deadline urgency is high, *DRSD*'s TAR is relatively high, but *DRSD*'s TCR is low. Although *DRSD* accepts a lot of tasks, the number of tasks completed on time is small. It indicates that a large amount of resources are wasted by *DRSD*. Besides, *DRPD* has the best performance when the deadline urgency is higher, but *DEED* shows the best performance when the urgency is lower. These all above are determined by the scheduling strategy of each algorithm.



(a) TCR (b) TAR (c) RTH

Figure 9: The impact of the probability of device communication

Fig. 9 shows the relationship between the probability of device communication and algorithm performance. The X-axis represents the probability of device communication, and the smaller the value, the lower the the probability that mobile device can communicate with others by the D2D communication. As shown in Fig. 9, TCR and TAR are closely related to the probability of device communication. As the probability of device communication increases, TCR and TAR increase accordingly. The reason is that with the increase of the probability of device communication, the number of devices which are capable of collaborative data offloading increases, so more tasks can be accepted and completed before its deadline. As a result, TCR and TAR relatively increase. On the whole, *DRPD* has the best performance when the probability of device communication is lower, but *DEED* is the best when the probability of device communication is higher. This is because *DRPD* adopts the reliability-priority strategy and it is more suitable for small-scale device set; *DEED* considers energy consumption and reliability comprehensively, so it has better adaptability for the higher probability of device communication. It is worth noting that when the probability of device communication is 0.6, *DRSD*'s RTH obviously exceeds other algorithms. However, when the probability of device communication is 0.7, its RTH declines rapidly, and then it increases with the increase of the probability of device communication. The reasons behind this phenomenon are complicated. When the probability of device communication is 0.6, the ability of mobile device communicating with others is weak, which means the device set is divided into multiple small subsets that are isolated from each other. *DRSD* adopts the random selection method, which leads to the increase in the probability of selecting a device with poor D2C communication in these small subsets. To satisfy the task constraints, *DRSD* has to increase the number of backups, so its RTH is significantly greater than other algorithms. However, in the same situation, the TAR and TCR of the *DRSD* are the smallest, which indicates that *DRSD* not only wastes resources, but also has poor adaptability to the smaller probability of device communication. When the probability of device communication is 0.7, the ability of mobile device communicating with others is slightly enhanced, and the number of backups is reduced. Thus, the RTH of *DRSD* declines. When the probability of device communication is greater than

0.7, with the increase of TAR and TCR, the resource utilization of each device increases, so the RTH of *DRSD* increases accordingly.

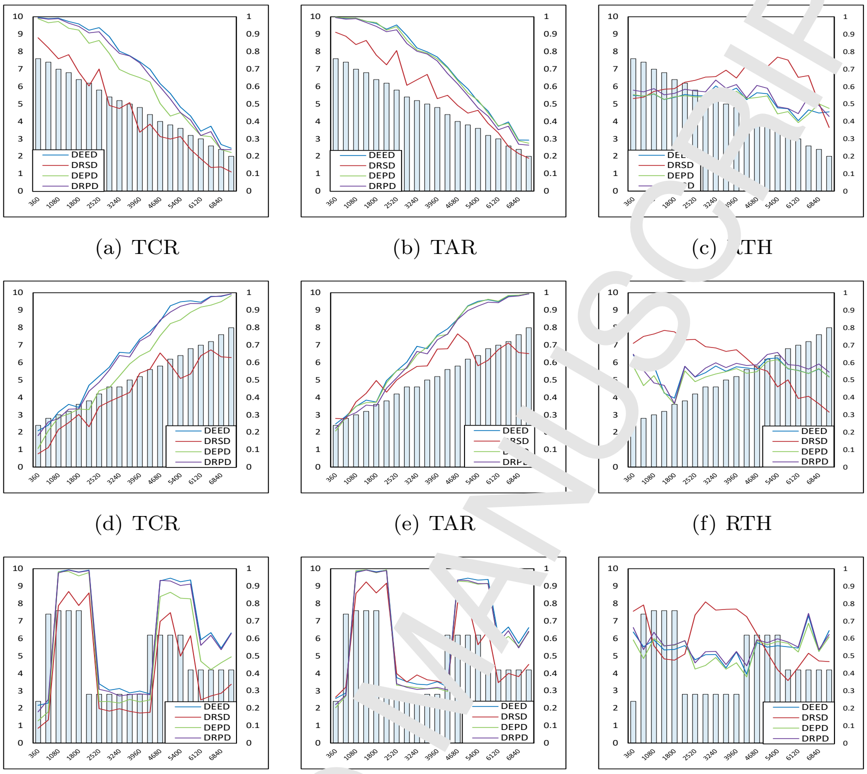
In general, under the static communication conditions, *DEED* performs better reliability and adaptability than other algorithms, and *DRPD* has a good performance while the task arrival rate and task deadline urgency are high. *DEPD* has a good performance in saving energy, whereas *DRSD* not only wastes resources but also has low reliability and adaptability.

5.2.2. Experiments under Dynamic Communication Conditions

For dynamic communication conditions, we study the influence of channel state on the performance of all algorithms. The change of channel state involves D2C state and D2D state. We study the performance of all algorithms in 7200 seconds where the channel state is updated every 360 seconds. The adaptability and elasticity of each algorithm are verified under the continuous deterioration, continuous improvement, and severe fluctuation communication conditions. To characterize the channel conditions, we define $\overline{S_c}$ ($\overline{S_d}$) as the average of the D2C (D2D) channel states of all devices. Based on the experimental setup, we compare these algorithms with respect to the TCR, TAR and RTH. It should be mentioned that the bar represents the mean channel state measured by the main Y-axis, the curve represents the metrics of each algorithm measured by the secondary Y-axis, and the X-axis represents the update time in the figures below.

The impacts of the degrading D2C channel state on each algorithm are shown in Fig. 10 (a)(b)(c). With the deterioration of $\overline{S_c}$, the TCR and TAR of all algorithms show the significant downward trend, which means the D2C channel state has an important impact on the TCR and TAR. However, the RTH firstly rises slowly, then falls quickly as shown in Fig. 10 (c). This is because when the channel quality begins to decline, to ensure the reliability, algorithms need to increase the backups of task, which leads to the increase of RTH. However, when the channel quality is too low, many tasks are rejected because they cannot be completed, so the RTH drops. Fig. 10 (d)(e)(f) show the performance of algorithms when the channel quality is gradually improved. With the improvement of D2C channel quality, the TCR and TAR of all algorithms increase rapidly and marginally close to 1. On the contrary, RTH shows the tendency to rise at first, then slowly decline. This is because when $\overline{S_c}$ is small, the TAR is low, so many devices are idle. Nevertheless, when $\overline{S_c}$ is large, the number of task backups is reduced, so many devices are idle. When the update time is more than 4680 seconds, the TAR and TCR of *DRSD* drop significantly, which means the random selection strategy shows poor adaptability. As shown in Fig. 10 (g)(h)(i), we evaluate the performance of each algorithm when the D2C channel quality is in violent fluctuations. When $\overline{S_c}$ soars quickly, TAR and TCR increase accordingly. Conversely, when $\overline{S_c}$ rapidly deteriorates, TAR and TCR also decrease quickly. Before 4320 seconds, the RTH of *DEED*, *DEPD* and *DRPD* are all in fluctuation. This phenomenon further proves the poor or good channel quality will lead to the decrease of RTH. Compared with other algorithms, *DEED* is more suitable for fast changing channel conditions.

ACCEPTED MANUSCRIPT



(a) TCR (b) TAR (c) RTH

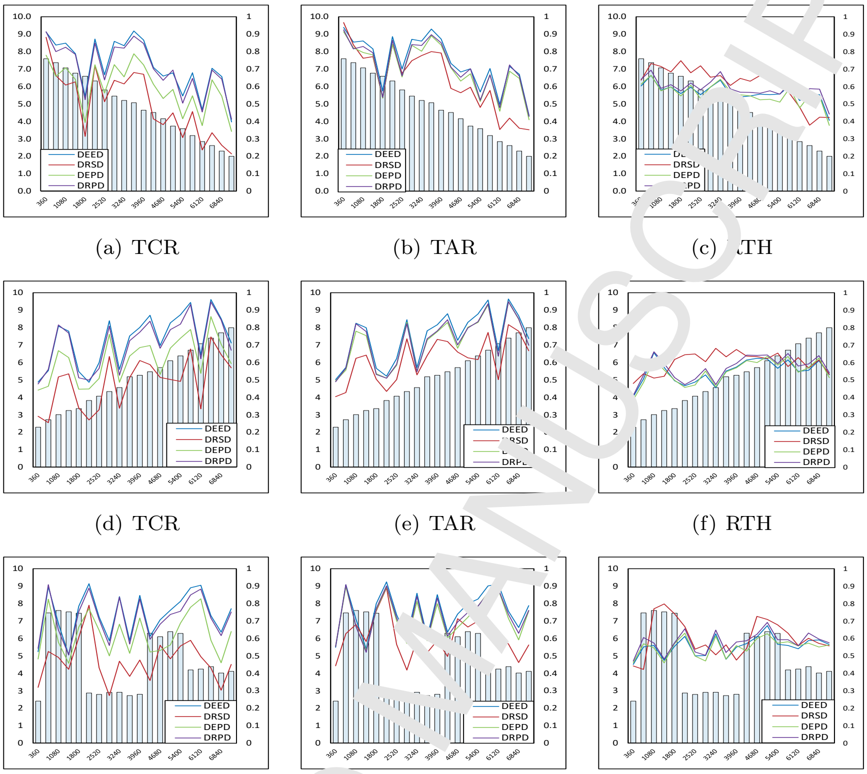
(d) TCR (e) TAR (f) RTH

(g) TCR (h) TAR (i) RTH

Figure 10: The impact of the D2C channel state

Totally, each algorithm is closely related to the D2C channel state. The performance of *DEED*, *DEPD* and *DRPD* is generally the same when the channel quality is good, but *DEED* has the better adaptability for the poor channel quality.

Compared with the D2C channel state, the D2D channel state has smaller impact on the performance of the algorithms. The impacts of deterioration D2D channel state on all algorithms are shown in Fig. 11 (a)(b)(c). With the deterioration of $\overline{S_d}$, the TCRs and TARs of all algorithms decrease in fluctuations, and the RTHs show the tendency to fluctuate first and then decrease. This indicates the deterioration D2D channel state has little impact on the performance of algorithms, but when $\overline{S_d}$ is less than a certain threshold, it will lead to the rapid performance degradation. It is worth noting that the performance of all algorithms drops abnormally at the 1800 second. This is because the D2D channel quality of devices with good D2C channel quality is too poor, so it results in the rapid decline of the performance. Fig. 11 (d)(e)(f) show the performance of algorithms when the channel quality is gradually improved. With the increase



(a) TCR (b) TAR (c) RTH

(d) TCR (e) TAR (f) RTH

(g) TCR (h) TAR (i) RTH

Figure 11: The impact of the D2D channel state

of the D2D channel quality, TCR and TAR generally show the growing trend, and RTH increases first and then fluctuates. As shown in Fig. 11 (g)(h)(i), we evaluate the performance of each algorithm when the D2D channel quality is in violent fluctuations. There is no obvious correlation between $\overline{S_d}$ and TCR, TAR and RTH, especially when $\overline{S_d}$ is large than 3. This phenomenon indicates that $\overline{S_d}$ will have a significant impact on the performance of algorithms only when it is below a certain threshold.

In general, the D2D channel state has little effect on the performance of the algorithms. However, once the D2D channel state of a device with a good D2C channel state deteriorates, the performance of the algorithm will be greatly degraded.

6. Conclusions and Future Work

In this paper, we focus on the data offloading for the IoT applications under unstable channel conditions. We propose a dynamic energy-efficient data offloading scheduling algorithm *DEED*, which can effectively deal with the problem of collaborative data offloading under unstable channel conditions. We propose a novel method to model the unstable channel quality. Meanwhile, we propose an optimal task scheduling model and a method to reduce the algorithm complexity, which can improve the algorithm efficiency without impairing its performance almost. Through a large number of simulation experiments, we evaluate the performance of *DEED*. We set up three performance metrics to measure the reliability, robustness, and resource utilization of *DEED*. Compared with three comparison algorithms *DRSD*, *DEPD*, and *DRPD*, the proposed *DEED* shows better performance under both the static and dynamic communication conditions.

The following issues will be studied in our future work. Firstly, we will study the fault tolerance model for data offloading to further enhance the data reliability from the perspective of hardware. Secondly, we will study the topology discovery and channel quality-awareness model based on the Zigbee protocol to facilitate the implementation of distributed scheduling algorithms. Finally, we plan to apply *DEED* to the IoT application that we are researching, such as Cooperative Reconnaissance of Drones, to test its performance.

7. Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 6187[illegible]78, 6[illegible]572511, and 91648204, in part by Science Fund for Distinguished Young Scholars in Hunan Province under grant 2018JJ1032, in part by the China Postdoctoral Science Foundation under Grant 2016M602960 and Grant 2017T100[illegible].

Reference

[1] Astrachan, O., 200[illegible]. Bubble sort: An Archaeological Algorithmic Analysis. ACM Sigcse Bulletin [illegible] (1), 1–5.

[2] Balas, E., Toth, P., 19[illegible]. Branch and bound methods. Ioe.engin.umich.edu 149 (1), 361–[illegible].

[3] Baronti, P., Pillai, P., Chook, V. W. C., Chessa, S., Gotta, A., Hu, Y. F., 2007. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. Computer Communications 30 (7), 1655–1695.

[4] Chen, C. A., Won, M., Stoleru, R., Xie, G. G., 2015. Energy-Efficient Fault-Tolerant Data Storage and Processing in Mobile Cloud. IEEE Transactions on Cloud Computing 3 (1), 28–41.

[5] Chen, H., Zhu, X., Liu, G., Pedrycz, W., 2018. Uncertainty-Aware Online Scheduling for Real-Time Workflows in Cloud Service Environment. IEEE Transactions on Services Computing, DOI:10.1109/TSC.2018.2866421.

[6] Chen, H., Zhu, X., Qiu, D., Liu, L., Du, Z., 2017. Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. IEEE Transactions on Parallel and Distributed Systems 28 (9), 2674–2688.

[7] Chiang, M., Zhang, T., 2017. Fog and IoT: An Overview of Research Opportunities. IEEE Internet of Things Journal 3 (6), 854–864.

[8] Ding, A. Y., Han, B., Xiao, Y., Hui, P., 2013. Enabling energy-aware collaborative mobile data offloading for smartphones. In: Sensor, Mesh and Ad Hoc Communications and Networks. pp. 487–495.

[9] Ding, A. Y., Hui, P., Kojo, M., Tarkoma, S., 2012. Enabling energy-aware mobile data offloading for smartphones through vertical collaboration. In: ACM Conference on CONEXT Student Workshop. pp. 27–28.

[10] Han, Y., Zhu, Y., Yu, J., 2015. Utility-maximizing data collection in crowd sensing: An optimal scheduling approach. In: IEEE International Conference on Sensing, Communication, and NETWORKING. pp. 345–353.

[11] Jararweh, Y., Doulat, A., Darabseh, A., Alsmirat, M., Al-Ayyoub, M., Benkhelifa, E., 2016. SDMEC: Software Defined System for Mobile Edge Computing. In: IEEE International Conference on Cloud Engineering Workshop. pp. 88–93.

[12] Kwak, J., Kim, Y., Lee, J., Chong, S., 2015. DREAM: Dynamic Resource and Task Allocation for Energy Minimization in Mobile Cloud Systems. IEEE Journal on Selected Areas in Communications 33 (12), 2510–2523.

[13] Li, H., Ota, K., Dong, M., 2018. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. IEEE Network 32 (1), 96–101.

[14] Li, H., Shou, G., Hu, Y., Guo, Z., 2016. Mobile edge computing: Progress and challenges. In: IEEE International Conference on Mobile Cloud Computing.

[15] Li, K., Tang, X., Veeravalli, B., Li, K., 2014. Scheduling Precedence Constrained Stochastic Tasks on Heterogeneous Cluster Systems. IEEE Transactions on Computers 64 (1), 191–204.

[16] Li, L., Ota, K., Dong, M., 2018. Deep Learning for Smart Industry: Efficient Manufacture Inspection System with Fog Computing. IEEE Transactions on Industrial Informatics PP (99), 4665–4673.

[17] Olkin, I., Gleser, L. J., Derman, C., 1980. Probability models and applications. Free Press.

[18] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge Computing: Vision and Challenges. IEEE Internet of Things Journal 3 (5), 637–646.

[19] Wang, J., Zhu, X., Bao, W., Liu, L., 2017. A Utility-Aware Approach to Redundant Data Upload in Cooperative Mobile Cloud. In: IEEE International Conference on Cloud Computing. pp. 384–391.

[20] Xiao, W., Bao, W., Zhu, X., Zhou, W., Peizhong, L., 2016. Improving the Performance of Data Sharing in Dynamic Peer-to-Peer Mobile Cloud. In: IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS). pp. 743–752.

[21] Xie, J., Guo, D., Li, X., Shen, Y., Jiang, X., 2018. Cutting Long-tail Latency of Routing Response in Software Defined Networks. IEEE Journal on Selected Areas in Communications PP (99), 1–1.

[22] Xu, J., Ota, K., Dong, M., 2018. Saving Energy on the Edge: In-Memory Caching for Multi-Tier Heterogeneous Networks. IEEE Communications Magazine 56 (5), 102–107.

[23] Yan, H., Zhu, X., Chen, H., Guo, H., Zhou, W., Bao, W., 2019. DEFT: Dynamic Fault-Tolerant Elastic scheduling for tasks with uncertain runtime in cloud. Information Sciences 477, 30–46.

[24] Yang, W., Li, H., Wu, J., 2011. ACK Offloading for Reliable Multipath Transfer over Self-Contention Wireless Network. In: Third International Conference on Communications and Mobile Computing. pp. 165–169.

[25] Yi, S., Hao, Z., Qin, Z., Li, Q., 2015. Fog computing: Platform and applications. In: Third IEEE Workshop on Hot Topics in Web Systems and Technologies.

[26] Zhang, L., Li, K., Xu, Y., Mei, J., Zhang, F., Li, K., 2015. Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. Information Sciences 319 (C), 113–131.

[27] Zhang, Q., Lin, M., Yang, L. T., Chen, Z., Khan, S. U., Li, P., 2018. A Double Deep Q-learning Model for Energy-efficient Edge Scheduling. IEEE Transactions on Services Computing, DOI:10.1109/TSC.2018.2867482.

[28] Zhang, Q., Yang, L. T., Li, P., Bu, F., 2018. An Adaptive Dropout Deep Computation Model for Industrial IoT Big Data Learning with Crowdsourcing to Cloud Computing. IEEE Transactions on Industrial Informatics, DOI:10.1109/TII.2018.2791424.

[29] Zhu, X., Sim, K. M., Jiang, J., Wang, J., Chen, C., Liu, Z., 2017. Agent-based Dynamic Scheduling for Earth-Observing Tasks on Multiple Airships in Emergency. IEEE Systems Journal 10 (2), 661–672.

[30] Zhu, X., Wang, J., Guo, H., Zhu, D., Yang, L. T., Liu, L., 2016. Fault-Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds. IEEE Transactions on Parallel and Distributed Systems 27 (12), 3501–3517.

Hui Yan received the B.S. degree from the College of Systems Engineering at National University of Defense Technology, China, in 2017. He is currently working toward his M.S. degree at the College of Systems Engineering, National University of Defense Technology. His research interests include cloud computing, edge computing, system reliability, and multi-objective optimization.

Xiongtao Zhang received his B.S. degree from the College of Systems Engineering at National University of Defense Technology, China, in 2018. Currently, he is working toward his M.S. degree at the College of Systems Engineering, National University of Defense Technology. His research interests include cloud computing, edge computing, swarm intelligence and evolutionary algorithms.

Huangke Chen received his B.S. and M.S. degree from the College of Information and System Management at National University of Defense Technology, China, in 2012 and 2014, respectively. Currently, he is working toward his Ph.D. degree at the College of Systems Engineering, National University of Defense Technology. He was a visiting Ph.D. student at University of Alberta, Edmonton, AB, Canada, from Mar. 2017 to Mar. 2018. His research interests include computational intelligence, multi-objective evolutionary algorithms, large-scale optimization, task and workflow scheduling.

Yun Zhou received her Ph.D degree in Mechatronics Engineering and Automation from National University of Defense Technology in 2010. She is currently an associate professor in the College of Systems Engineering at National University of Defense Technology, Changsha, China. Her recent research interests include modeling and simulation and cloud computing. She has published more 30 research articles in referred journals and conference proceedings.

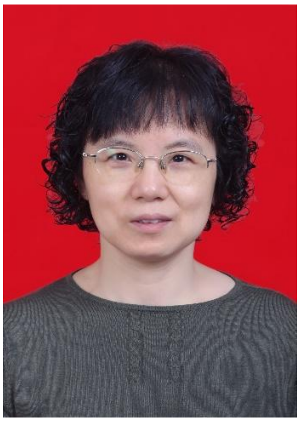
Weidong Bao received the Ph.D. degree in information system from the National University of Defense Technology in 1999. He is currently a professor in the College of Systems Engineering at National University of Defense Technology, Changsha, China. His recent research interests include cloud computing, information system, and complex networks. He has published more than 100 research articles in refereed journals and conference proceedings such as IEEE TC, IEEE TPDS, IEEE CLOUD and so on. He serves on the editorial board of AIMS Big Data and Information Analytics.

Laurence T. Yang has published around 300 papers in refereed journals, conference proceedings and book chapters. His research fields include networking, high performance computing, embedded systems, ubiquitous computing and intelligence. He has been involved in more than 100 conferences and workshops as a program/general/steering conference chair and more than 300 conference and workshops as a program committee member. Currently is the chair of IEEE Technical Committee of Scalable Computing (TCSC), the chair of IEEE Task force on Ubiquitous Computing and Intelligence, the co-chair of IEEE Task force on Autonomic and Trusted Computing. He is also in the executive committee of IEEE Technical Committee of Self-Organization and Cybernetics for Informatics, and of IFIP Working Group 10.2 on Embedded Systems.

Hui Yan

Xiongtao Zhang

Huangke Chen



Yun Zhou

Weidong Bao

Laurence T. Yang

Highlights

1. We proposed an intricate device framework to facilitate the decision making of data offloading for mobile Internet of Things applications.

2. We proposed a novel method to model the unstable channel quality that makes it more realistic. Based on that, we proposed an optimal task scheduling model.

3. We proposed an innovative dynamic energy-efficient data offloading scheduling algorithm, *DEED*, to as much as possibly reduce the energy consumption while ensuring task reliability.

4. We proposed a method to reduce the algorithm complexity which could improve the algorithm efficiency without impairing its performance almost.

5. Extensive experiments were conducted to verify the performance of data offloading among *DEED* and other algorithms both under the static and dynamic communication conditions.