Energy Conserved Computation Offloading for O-RAN based IoT systems

Liping Wang*, Jianhong Zhou*†, IEEE member, Yunxiang Wang*, Boyi Lei*

*School of Computer and Software Engineering,Xihua University
†National Key Laboratory of Science and Technology on Communications,
University of Electronic Science and Technology of China
E-mail:zhoujh@uestc.edu.cn

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838678

Abstract—Executing computation offloading under the mobile edge computing (MEC) network architecture is a widely recognized way to reduce latency and energy consumption for IoT systems. However, under legacy MEC network architecture, it is hard to jointly utilize global and local information to derive the optimal offloading strategy at the distributed edge servers, which may cause the latency and energy consumption reduction to the bottleneck. To meet the ultra-low latency and ultra-low energy consumption requirements of future IoT system, researchers start to focus on involving the recent emerging open radio access network (O-RAN) architecture into IoT systems. In this paper, we design a computation offloading strategy for an O-RAN based IoT systems to reduce the energy consumption of IoT devices (IoTDs) under the premise of latency requirement. Specifically, we first adopt O-RAN architecture to deploy the IoT system. Under the O-RAN based IoT system, we formulate computation offloading problem with the aim of minimizing energy consumption by jointly deciding offloading ratio, transmit power, and local processing speed. To make the optimization problem tractable, we decouple the original problem into two sub-problems, and the original problem is solved by iteratively solving these two sub-problems. Experimental results show that under our proposed O-RAN based IoT system, the strategy we obtained can reduce the average energy consumption of all IoTDs by more than 20%.

Index Terms—energy conserved; computation offloading; offloading strategy; O-RAN; IoT system.

I. Introduction

With the development of next-generation mobile communication network, some scenarios may have extremely strict demands on IoT systems, such as massive connections and ultra-low latency. It requires the network deployment and management more flexible and intelligent. Unfortunately, it is difficult to meet these extreme demands by using existing network architecture [1]. The introduction of new network architecture into IoT systems is imminent. The O-RAN came into being under these demands. In February 2018, five service operators, namely China Mobile, AT&T, Deutsche Telekom, NTT DOCOMO and Orange, formed the O-RAN Alliance by merging the original C-RAN Alliance and xRAN Forum [2], which aims to clarify requirements and help the supply chain ecosystem to deploy innovative products and services to strengthen the industry in 21st century [3].

As more and more IoT devices' tasks are computationally intensive (*e.g.*, smart transportation, and smart healthcare) [4], the limited computing power and battery power of most IoTDs may become a bottleneck of the system. Researchers often design computation offloading based on traditional network architecture to increase computing speed, save energy, and reduce latency [5], [6], [7].

Compared with the traditional network architecture, the O-RAN provides an intelligent, flexible and interoperable standard network architecture [8]. Therefore, performing computation offloading under O-RAN architecture can solve some problems encountered under traditional network architecture, thereby effectively improving the IoT system performance. Specifically, under the traditional network architecture, it is impossible to simultaneously collect local and global real-time information when performing computation offloading (*e.g*, the number of real-time tasks that each IoTD needs to process and the currently available resources of all DUs cannot be obtained simultaneously). In this case, we cannot obtain optimal offloading and resource allocation strategies, which will result in suboptimal reduction of latency and energy consumption. However, the involvement and division of the near-real-time and the non-real-time intelligent controllers (RICs) in O-RAN architecture can solve this problem. Both RICs can adjust and optimize the strategy in real time by collecting the RAN's global and local information according to different time granularities, which can better reduce congestion and avoid waste of resources. Based on the above benefits, we try to deploy the IoT system under O-RAN architecture, and improve the IoTDs' energy efficiency by executing computation offloading for the O-RAN based IoT system.

In [9], the authors propose a partial offloading strategy to minimize the energy consumption of IoTDs with considering the delay limitation, transmission power limitation, the maximum number of CPU cycles and memory of the MEC network. Authors of [10] propose a joint optimization scheme of offloading and resource allocation based on spectrum efficiency in the task-aware C-RAN, and solve a profit maximization problem under the constraints of load delay, front haul capacity, limited bandwidth

and computing resources. All the works mentioned above only consider to apply computation offloading in legacy closed and rigid MEC network architecture, which leads the latency and energy consumption reduction to the bottleneck.

In order to take the advantages of O-RAN architecture and enable the IoT systems intelligent and flexible, we consider energy conserved computation offloading for an O-RAN based IoT (ORAN-IoT) system in this paper. Under the O-RAN based IoT system, we optimize the computation offloading strategy to minimize the energy consumption of IoTDs while meeting the latency requirement. The main contributions of this work are as follows:

1) We consider the IoTDs, DUs in O-RAN (O-DU), and cloud in O-RAN (O-cloud) for processing computationally intensive tasks collaboratively. Meanwhile, we use the plane that separated the control plane from the data plane by deploying CUs in O-RAN (O-CU) to realize the hierarchical management of the system through standardized communication interface.

2) We propose a computation offloading model with considering two cost indicators, which are delay and energy consumption. Both indicators are important to decide where some local tasks should be offloaded, the O-DU or the O-cloud.

3) We formulate a joint non-convex optimization problem to obtain the optimal offloading strategy based on local processing speed, transmit power, and local offloading ratio.

The rest of the paper is organized as follows: the system model is described in section II. The optimization problem is defined and analyzed in Section III. The simulation results are presented in section IV. The paper is concluded in section V.

II. System model

We design an O-RAN based IoT system as Fig 1. All IoTDs may send requests to the DUs through the connected RUs. All RUs contain the transceiver antenna and dedicate to performing physical layer operations. Meanwhile, the operations of RUs and DUs are controlled by the connected CUs, which are responsible for handling higher layer protocols. The connections between RUs and DUs are high-capacity front haul links. The DUs are connected with the central unit control plane (CU-UP) and central unit user data plane (CU-UP) of the CUs through F1-C and F1-U, respectively. The near-real-time RICs are deployed in the CUs and connected to non-real-time RICs, which are deployed in the service management and orchestration (SMO). It is worth noting that under the O-RAN architecture, the SMO is mainly responsible for managing network functions and coordinated control. The non-real-time RICs mainly perform data analysis by collecting non-real-time information of RAN, such as the number of terminal devices served, uplink channels, quality of service requirements of devices, and server resources at DUs and CUs, *etc.* The inference and policies are downlinked to the near-real-time RICs through A1 interfaces. The near-real-time RICs are responsible for collecting and analyzing the immediate information of RAN. Thus, the near-real-time RICs could monitor the behavioral changes of IoT devices, and make adjustments in real time. Only the execution results are transmitted to the IoT devices. We also deploy A1, O1, and O2 interfaces in the IoT system. The A1 interface enables the non-real-time RICs to provide policy-based guidance for near-real-time RICs. The O1 interfaces provide open interfaces for the SMO to manage the CUs, DUs and RUs in the system. The O2 interfaces are between the SMO and the O-cloud, which provide intelligent management and operation for the O-cloud.

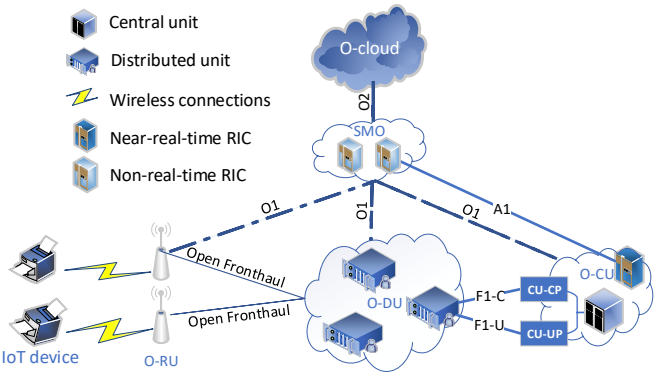

Fig. 1. O-RAN based IoT system model

A. Channel model

Assume that there are $m, m \in \{1, 2, \ldots, M\}$ IoTDs in the system. We consider the case of full frequency multiplexing where the spectrum used by the RUs is overlapped and hence the inter-RUs interference should be taken into consideration. However, the spectrum is assigned orthogonally to the terminal devices when accessing the same RU. Thus the intra-RU interference could be ignored. We assume that every RU has a full CSI for all downlinks between RUs and IoTDs. The signal interference noise ratio (SINR) for RU r to transmit to IoTD m could be obtained as:

$$SINR_{m,r} = \frac{p_m g_{m,r}}{\sigma^2 + \sum_{j=1, j \neq r}^{R} p_m g_{m,j}}, \tag{1}$$

where p_m is the transmit power of IoTD m, $g_{m,r}$ denotes the channel gain from RU r to IoTD m. σ^2 is the power of additive Gaussian white noise (AWGN), whose distribution is assumed to obey the Gaussian distribution. According to Shannon's formula, so the propagation rate of the signal over the wireless link channel for RU r to transmit to IoTD m $V_{m,r}$ is:

$$V_{m,r} = Blog_2\left(\frac{p_m g_{m,r}}{\sigma^2 + \sum_{j=1, j \neq r}^{R} p_m g_{m,j}}\right), \tag{2}$$

where B is the total available bandwidth.

B. Delay model

In the ORAN-IoT system, the delay is mainly caused by three aspects: transmission, processing and queuing. Thus, the task latency in our system includes the transmission latency $t^D_{m,off}$ and $t^C_{m,off}$, which represent the transmission delay to the DUs and to the O-cloud, respectively. The processing latency are $t^L_{m,exe}$, $t^D_{m,exe}$ and $t^C_{m,exe}$, which represent the processing delay on-premises,at the DUs and at the O-cloud, respectively. The queuing latency is Q^D_m, which represent the queuing delay at the DUs. As the O-cloud has abundant computing resources, the queueing delay at the O-cloud will not be considered. Next, we model the three latencies in detain.

(1) The queuing latency

The queuing delay of each DU is modeled as having an Q/Q/1 queue with one DU, where the first Q means that the traffic arrival rate follows a Poisson distribution and the second Q means that the service rate is exponentially negatively distributed. The queuing delay is calculated based on the traffic and node capacity, which is calculated as:

$$Q^D_m = \frac{1}{Y_d - X_d}, \tag{3}$$

where Y_d is the traffic arrival rate of the DUs and X_d is the service rate of the DUs.

(2) The transmission latency

We assume that the input data size of the m_{th} IoTD is D_m , and the offloading ratio is $1-\lambda_m, (0<\lambda_m<1)$, which means that the amount of data executed locally is $\lambda_m D_m$, and the amount of data offloaded is $(1-\lambda_m)\,D_m$. The front haul transmission for RU r to transmit to DU d, which assume that the front haul capacity is $C_{r,d}$. Thus, the transmission latency could be obtained as:

$$t^D_{m,off} = \frac{(1-\lambda_m)\,D_m}{\mathrm{V}_{m,r}} + \frac{(1-\lambda_m)\,D_m}{\mathrm{C}_{r,d}}, \tag{4}$$

$$t^C_{m,off} = \frac{(1-\lambda_m)\,D_m}{\mathrm{V}_{m,r}} + d_m, \tag{5}$$

where d_m indicates the latency caused by the wired transmission to the O-cloud.

(3) The processing latency

We define f^L_m,f^D_m,f^C_m as the computation capability assigned by the IoTD, the O-DU and the O-cloud to the task, respectively. We assume that $f^D_m < f^C_m$, thus the processing latency could be modeled as follows:

$$t^L_{m,exe} = \frac{\lambda_m D_m}{f^L_m}, \tag{6}$$

$$t^D_{m,exe} = \frac{(1-\lambda_m)\,D_m}{f^D_m}, \tag{7}$$

$$t^C_{m,exe} = \frac{(1-\lambda_m)\,D_m}{f^C_m}, \tag{8}$$

C. Energy consumption model

In our problem, the total energy consumption e_m includes transmission energy consumption e^D_m and e^C_m, which means the transmission energy consumption to the DUs and the O-cloud, respectively. Meanwhile, we also need to consider the local processing energy consumption e^L_m. The three kinds of energy consumption mentioned above could be modeled as follows:

$$e^D_m = p_m t^D_{m,off} + r_m, \tag{9}$$

$$e^C_m = p_m t^C_{m,off} + r_m, \tag{10}$$

$$e^L_m = \varepsilon \lambda_m D_m {f^L_m}^2, \tag{11}$$

where r_m is the energy consumption after the data transmission is completed. ε is the chip factor.

D. Computation offloading model

In order to make our system perform better, we define two cost indicators to decide whether computationally intensive tasks should be offloaded to O-DU or O-cloud. We define the offloading option for each device as a_m, where $a_m = 1$ indicates offloading the task to the O-cloud, and $a_m = 0$ indicates offloading the task to the O-DU. Thus, the delay cost in the O-DU and the O-cloud are computed as follows:

$$t^D_m = t^D_{m,off} + t^D_{m,exe} + Q^D_m, \tag{12}$$

$$t^C_m = t^C_{m,off} + t^C_{m,exe}, \tag{13}$$

Therefore, the total cost L^D_m and L^C_m, which means that the task of IoTD m is offloaded to the O-DU and the O-cloud, respectively.

$$L^D_m = \gamma^e_m e^D_m + \gamma^t_m t^D_m, \tag{14}$$

$$L^C_m = \gamma^e_m e^C_m + \gamma^t_m t^C_m, \tag{15}$$

where,γ^e_m,$\gamma^t_m \in [0,1]$ can be tuned according to the requirement of the task.

We first define the size of data to be offloaded as D^*_m and the power of completing the emission as p^*_m. For simplicity, we first ignore the latency of the front haul link, so we can compute a threshold d^{thr} associated with d_m as:

$$d^{thr} = \frac{D^*_m\left(f^L_m - f^D_m\right)}{\left(1 + p^*_m \gamma^e_m/\gamma^t_m\right) f^C_m f^D_m}, \tag{16}$$

If $d_m < d^{thr}$, then $a_m = 1$, otherwise, $a_m = 0$.

III. Energy conserved strategy for computation offloading strategy

In this section, we formulate the computation offloading problem with considering the energy consumption and latency requirement as a non-convex optimization problem and obtain the energy efficient computation offloading strategy by resolving this problem.

A. Problem formulation

According to the model mentioned above, the total energy consumption e_m and the total delay t_m for the m_{th} device could be calculated as:

$$e_m = e_m^L + a_m e_m^C + (1 - a_m)\, e_m^D, \tag{17}$$

$$t_m = \max\left\{t_{m,exe}^L, a_m t_m^C + (1 - a_m)\, t_m^D\right\}, \tag{18}$$

By jointly considering the local computational speed $F^L = \left(f_m^L\right)_{\forall m}$, the offloading strategy $A = (a_m)_{\forall m}$, the transmitting power $P = (p_m)_{\forall m}$, and the offloading ratio $\lambda = (\lambda_m)_{\forall m}$, we obtain the weighted sum of energy consumption of all IoTDs $\sum_{m=1}^{M} \omega e_m$. Here, the weighted sum can be considered as an energy consumption tradeoff among all IoTDs. Our goal is to obtain a combination strategy to minimize energy comsumption with consideration of f^L, A, P, and λ. The problem could be formulated as follows:

$$\begin{aligned}
\mathbf{P}_1: \quad & \min_{\{f^L, P, A, \lambda\}} \sum_{m=1}^{M} \omega e_m \\
\text{s.t.} \quad & \text{C1}: 0 \le f_m^L \le F^L, \forall m \\
& \text{C2}: 0 \le p_m \le P_{MAX}, \forall m \\
& \text{C3}: 0 \le \lambda_m \le 1, \forall m \\
& \text{C4}: 0 \le t_m \le T_{MAX}, \forall m \\
& \text{C5}: a_m = 0, \forall m \\
& \text{C6}: a_m = 1, \forall m
\end{aligned}$$

where C_1 is the maximum local processing speed constraint, C_2 is the non-negative transmit power constraint, C_3 is the offloading ratio constraint, C_4 is the maximum latency constraint, C_5 and C_6 are the offloading strategy constraints. As the problem P1 is non-convex, we divide it into two sub-problems, which are to determine the offloading point among IoTD, O-DU, O-cloud and to allocate the offloading ratio, transmission power and local processing speed according to the determined offloading point. We adopt a continuous convex optimization framework for convex approximation of the two sub-problems and solve the non-convex optimization problem by iteratively solving the two sub-problems.

B. Problem Analysis

For problem P_1, we first focus on the local processing speed adjustment to obtain the optimal computational speed for each IoTD in a close-form. Since e_m increases monotonically with f_m^L, P_1 could be solved by minimizing f_m^L and reducing the dimensionality of the original problem, which can be expressed as follows:

$$\frac{\lambda_m D_m}{T_{MAX}} \le f_m^L, \tag{19}$$

Thus:

$$0 \le \lambda_m \le \min\left\{1, \frac{F^L T_{MAX}}{D_m}\right\}, \tag{20}$$

We can calculate the value of d^{thr} by using (14), (15), (16) according to the f_m^L, f_m^D, f_m^C,D^* and p^*, and compare the value of d^{thr} and d_m to search for the value of A, which can be defined as a_m^*, the P_1 could be transformed to P_2 as follows:

$$\begin{aligned}
\mathbf{P}_2: \quad & \min_{\{P, \lambda\}} \sum_{m=1}^{M} \omega e_m \\
\text{s.t.} \quad & \text{C2} \\
& C_7: 0 \le \lambda_m \le \min\left\{1, \frac{F^L T_{MAX}}{D_m}\right\}, \forall m \\
& C_8: 0 \le \max\left\{t_{m,exe}^L, a_m^* t_m^C + (1 - a_m^*)\, t_m^D\right\} \\
& \qquad \le T_{MAX}, \forall m
\end{aligned}$$

C_7 and C_8 are converted from (20) and (18). According to the a_m^*, we denote t_m^* as the latency in the DUs or the O-cloud after executing the policy, which includs the queuing delay and the processing latency. Therefore, C_8 can be expressed as:

$$h_m(p_m, \lambda_m) = \frac{(1 - \lambda_m)\, D_m}{B log_2(\frac{p_m g_{m,r}}{\sigma^2 + \sum_{j=1, j \ne r}^{R} p_m g_{m,j}})} - T_{MAX} + t_m^* \le 0, \tag{21}$$

This problem is still non-convex due to (21), which can be divided into (22) and (23) according to the non-convexity and expressed as:
non-convex:

$$h'_m(p_m) = -log_2(\frac{p_m g_{m,r}}{\sigma^2 + \sum_{j=1, j \ne r}^{R} p_m g_{m,j}}), \tag{22}$$

convex:

$$h'_m(\lambda_m) = \frac{B(t_m^* - T_{MAX})}{(1 - \lambda_m) D_m}, \tag{23}$$

We exploit the framework of successive inner convexification to optimize a sequence of approximate convex problems [11], which allows the development of a computationally-efficient algorithm. By solving the convex approximate solution of the original problem, the non-convex objective function and constraints are replaced with a suitable convex approximation. Let $p_m = 2^{q_m}$, we have:

$$h'_m(q_m) = -\log_2\left(1 + \frac{2^{q_m} g_{m,r}}{\sigma^2 + \sum_{j=1, j \ne r}^{R} 2^{q_m} g_{m,j}}\right), \tag{24}$$

where $q_m \triangleq [q_1, q_2, \ldots, q_m]^T$. At the k_{th} sequence of convexification is denoted as $\tilde{h}'_m\left(q_m^{(k)}; q_m^{(k-1)}\right)$, for a convex approximation of $h'_m\left(q_m^{(k)}\right)$, we require the following three properties to be satisfied:

$$h'_m\left(q_m^{(k)}\right) \le \tilde{h}'_m\left(q_m^{(k)}; q_m^{(k-1)}\right), \quad \forall m, k, \tag{25}$$

$$h'_m\left(q_m^{(k-1)}\right) = \tilde{h}'_m\left(q_m^{(k-1)}; q_m^{(k-1)}\right), \quad \forall m, \tag{26}$$

$$\nabla h'_m\left(q_m^{(k)}\right) = \nabla \tilde{h}'_m\left(q_m^{(k-1)}; q_m^{(k-1)}\right), \quad \forall m, \tag{27}$$

The limit of any convergent sequence $q_m^{(k)}$ is a Karush-Kuhn-Tucker (KKT) point. Next, we use the Lagrangian multipliers ϑ_m and μ to solve the convex problem P_2, which can be defined as $L\left(q_m, \lambda_m, \vartheta_m, \mu\right)$. The closed form expression is found with the help of the KKT condition, the KKT conditions are given by:

$$\frac{\partial L}{q_m} = (\log 2) 2^{q_m} - \vartheta_m \varphi_m = 0, \tag{28}$$

$$\frac{\partial L}{\lambda_m} = \frac{-\left(t_m^* - T_{MAX}\right)}{\left(1-\lambda_m\right)^2 D_m} + \mu = 0, \tag{29}$$

where φ_m is proportion of $h_m'\left(q_m\right)$ in the iteration. The process of updating the resource allocation as well as the Lagrange multiplier is repeated until convergence. Therefore, we obtain the expression for the optimal offloading ratio and transmission power of the IoTD m, respectively.

Algorithm 1 Joint offloading algorithm to solve P_2

Input: B,γ_m^e, γ_m^t, $\lambda_m^{(0)}$, $p_m^{(0)}$,T_{MAX}
Output: λ_m and p_m

```
Minimize f_m^L from (19)
Search A according to (14) and (15)
Initialize k ← 1
Repeat
Update p_m^(k) using continuous internal convexizations
k ← k + 1
until (21) find a suitable approximation
Update A
Using Lagrangian multipliers to solve convex problems in KKT conditions
Until convergence
```

IV. Simulation Results

In this section, we realize the O-RAN based IoT system and simulate the computation offloading procedure to show the effectiveness of our proposed energy conserved strategy. We consider that the coverage radius of the O-RAN based IoT system is 1000m. Three RUs, two DUs and one CU are deplyed, and 30 IoTDs are randomly distributed in this area. For the wireless channel, we set the channel loss model as $g_{m,r} = 37.6 \times log(dist) + 127$, where $dist$ is the propagation distance. The other simulation parameters are summarized in Table I.

TABLE I
Partial Parameters

Parameter	Typical Value	Parameter	Typical Value
B	10MHz	T_{MAX}	1s
P_{MAX}	0.1w	$C_{r,d}$	100MHz
σ^2	2×10^{-13}	γ_m^e	0.5
γ_m^t	0.5	F^L	0.5GHz

We compare the average energy consumption of the entire system based on different strategies. For the energy consumption, two parts are considered, which are energy consumption for local task execution and energy consumption for offloading. The four strategies for comparison are energy conserved offloading (ECO) strategy, the strategy without offloading(WOO), the strategy without leaving-offloading (WLO) and the strategy with legacy-offloading (LO).

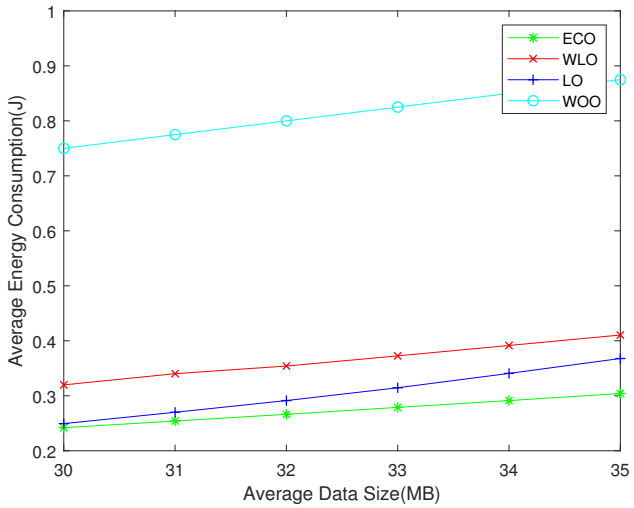

Fig. 2. The average energy consumption comparison with the number of tasks under four strategies

The strategy without offloading (WOO) means that all tasks are computed at IoT devices.

The strategy with legacy-offloading (LO) means that the task is partially offloaded to the O-DU with considering the amount of data [12].

The strategy without leaving-offloading (WLO) means that the task is completely offloaded to O-DU or O-cloud with considering the minimum energy and delay [13].

As shown in Fig 2, in terms of the average energy consumption, the offloading-based strategies, which are LO, WLO, and ECO strategies, are significantly better than the local execution-based strategy. Compared with local execution-based strategy (WOO strategy), the average energy consumption of ECO strategy is reduced by nearly 60% with the number of tasks increasing. As all tasks are executed at the IoTDs for the WOO strategy, the local processing energy consumption is extremely high while the emission energy consumption is extremely low. However, for the offloading-based strategies, only parts of the tasks are executed at IoTDs, which could better balance energy consumption of local processing and transmission and thus reduce total energy consumption. Obviously, the energy consumption for WLO strategy is higher and increases faster than both LO and ECO strategy. It is because that the WLO strategy considers offloading all tasks to O-DU or O-cloud. All offloading minimizes the energy consumption of local processing while increases the latency. Thus, more transmission power is needed to reduce the delay, which induces the transmission energy consumption increasing. In addition, the total energy consumtion for the LO strategy is also

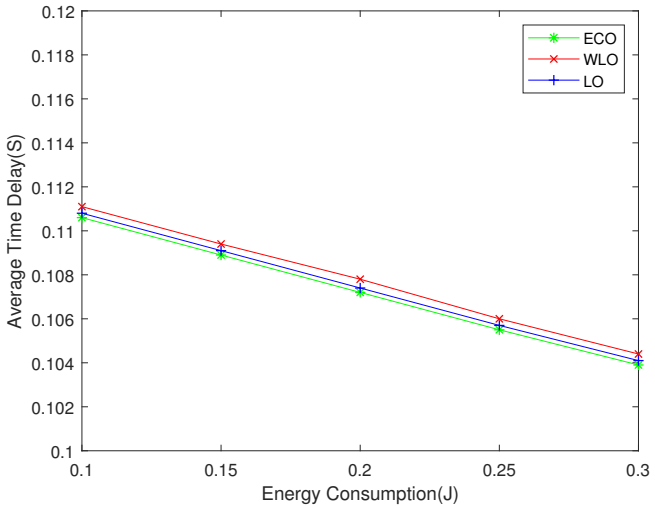

Fig. 3. Delay comparsion with the same energy consumption under three strategies

higher than that for the proposed ECO strategy. It is precisely because that the LO strategy is under the condition that the local processing speed is the same when executing offloading. if the number of tasks increases or the latency requirement becomes higher, more energy consumption may be required for local processing, which will result in an increase in total energy consumption. As the proposed ECO strategy could flexibly change the local processing speed and offloading rate as well as limit the offloading transmission power by considering delay constraints, it could better optimize the total energy consumption of local processing and transmission. Thus, the energy consumption for our proposed ECO strategy is the lowest among the four strategies and has been reduced by 26% and 12% when compared with WLO strategy and LO strategy, respectively.

Fig 3 shows the transmission delay comparison for the three strategies which are executed with the condition of same energy consumption. As no offloading happens for WOO stategy, we do not need to consider the transmission delay for the WOO strategy. As shown in Fig 3, if the number of arriving tasks is the same, the delay for the three offloading strategies will gradually decrease with the increasing of energy consumption. It is because that the increasing energy consumption could increase the local processing speed to allow more tasks to be processed locally or increase the transmission power to allow more tasks to be offloaded, which could reduce processing delay or transmission delay, respectively. Under the condition of the same energy consumption, the delay required for the ECO strategy is the lowest among the three strategies, which is lower than the delay for the LO strategy and the WLO strategy, respectively.

V. Conclusion

Making full use of the flexible and intelligent management of O-RAN architecture to reduce latency and energy consumption in IoT system has become a research hotpot in the IoT fields. In this work, we propose an energy consumption minimization problem for O-RAN based IoT system by optimizing offloading strategies. In order to solve this problem effectively, we decoupled the original problem into two sub-problems and solved it iteratively, determine the offloaded allocation among IoTD, O-DU, and O-cloud, and allocate the offload ratio, transmission power and local processing speed. Simulation results demonstrate the effectiveness of the proposed ECO strategy. In the future, we will better utilize the O-RAN to predict the arriving traffic and manage intelligently, so that the energy conserved offloading strategy can better adopt to the dynamic environment.

Acknowledgments

The work of this paper was supported by the National Science Foundation of China (No. 62171387), the China Postdoctoral Science Foundation (No. 2019M663475)

References

[1] S. K. Singh, R. Singh, and B. Kumbhani, "The evolution of radio access network towards open-ran: challenges and opportunities," in 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). IEEE, 2020, pp. 1–6.

[2] N. Kazemifard and V. Shah-Mansouri, "Minimum delay function placement and resource allocation for open ran (o-ran) 5g networks," Computer Networks, vol. 188, p. 107809, 2021.

[3] "O-ran: Towards an open and smart ran," https://www.o-ran.org/resources, 2018.

[4] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense iot networks," IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4977–4988, 2018.

[5] B. He and T. Li, "An offloading scheduling strategy with minimized power overhead for internet of vehicles based on mobile edge computing," Journal of Information Processing Systems, vol. 17, no. 3, pp. 489–504, 2021.

[6] C. Pradhan, A. Li, C. She, Y. Li, and B. Vucetic, "Computation offloading for iot in c-ran: Optimization and deep learning," IEEE Transactions on Communications, vol. 68, no. 7, pp. 4565–4579, 2020.

[7] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," IEEE Transactions on Communications, vol. 68, no. 3, pp. 1524–1537, 2019.

[8] S. Niknam, A. Roy, H. S. Dhillon, S. Singh, R. Banerji, J. H. Reed, N. Saxena, and S. Yoon, "Intelligent o-ran for beyond 5g and 6g wireless networks," arXiv preprint arXiv:2005.08374, 2020.

[9] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," IEEE Internet of Things Journal, vol. 8, no. 5, pp. 3774–3785, 2020.

[10] Z. Jian, W. Muqing, and Z. Min, "Joint computation offloading and resource allocation in c-ran with mec based on spectrum efficiency," IEEE Access, vol. 7, pp. 79 056–79 068, 2019.

[11] B. R. Marks and G. P. Wright, "A general inner approximation algorithm for nonconvex mathematical programs," Operations research, vol. 26, no. 4, pp. 681–683, 1978.

[12] F. Z. Sun, Haijian and R. Q. Hu, "Joint offloading and computation energy efficiency maximization in a mobile edge computing system," vol. 68, no. 3. IEEE, 2019, pp. 3052–3056.

[13] A. A. Alahmadi, T. E. El-Gorashi, and J. M. Elmirghani, "Energy efficient and delay aware vehicular edge cloud," in 2020 22nd International Conference on Transparent Optical Networks (ICTON). IEEE, 2020, pp. 1–4.