Delay optimization in MEC federation environment based on QoS level and resource allocation using Deep Learning method

Tran Ngoc Hung Anh, Myungsik Yoo
Soongsil Univ.

tranngochunganh@soongsil.ac.kr, myoo@ssu.ac.kr

QoS 및 자원 할당 기반 MEC 연합환경에서의 딥러닝 기법을 이용한 지연 최적화

트란녹흥안, 유명식

Abstract

The expansion of 5G technology recently leads to the develop of Multi-Access Edge Network (MEC) as a potential solution to achieve the quality of service (QoS) of applications inside user devices (UDs). This article proposes a method in which MEC servers (MECSs) can communicate together, all receive the decision from MEC controller about how to sharing the resource to achieve the optimal processing delay of all QoS classes. This problem can be solved by proposed Deep Deterministic Policy Gradient (DDPG) framework.

Index Terms – MEC federation, task priority, resource allocation, Markov decision process, deep reinforcement learning (DRL).

Ⅰ. Introduction

With the increase of the devices number in the era of next-generation mobile networks. MEC appears as a solution for responding with the tremendous data gathered by the devices while meet the demand of different QoS requirements. Many works focus on the optimization issue in MEC network such as minimize the delay [1], energy [2], or system cost [3]. However, these studies treat the tasks gathered by users equally, without considering the different of QoS requirements.

Our work develops a MEC network model by cooperating of task classification and resource allocation. We first classify tasks into different classes based on the QoS delay tolerance, then the resource allocation algorithm based deep reinforcement learning can intelligently allocate the resource to process each class to minimize the total delay.

Ⅱ. System model

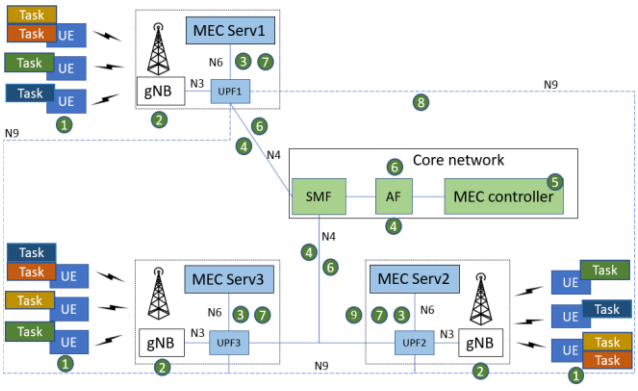

Figure 1. System architecture

The model of MEC federation system is described in Fig. 1. In our system, there are UDs, assumed IoT equipment, generating tasks and sending to corresponding UPFs, these UPFs connect to MECSs in their serving region to periodically receive MECSs' resource status, we consider the set of UPF and MECS are the same and expressed by $\{1, 2, ..., M\}$. Denote

that tasks can be migrated and processed at other MECS when a single MECS lacks resource.

A. Task classification model

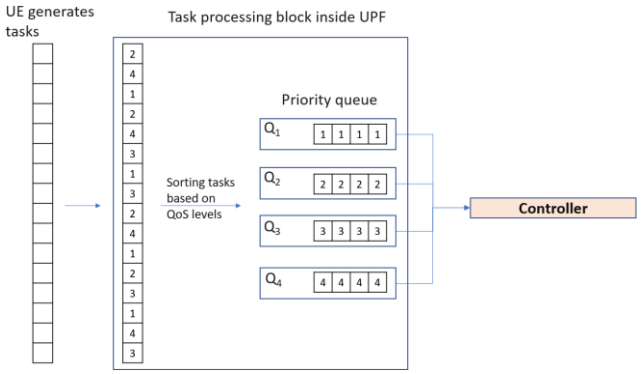

Figure 2. Classification model

The model of classifying tasks is shown in Fig. 2. At time slot t, multiple tasks generated by UDs will be sent to UPF. Inside each UPF, the task processing block will label and sort them into four different queueing classes based on the corresponding delay tolerances [4], denoted class $i \in \{1,2,3,4\}$. After that, the controller receives size and location of each class sent by UPF, then decide where the classes are executed and how much computing resource of server can supply to processing those. In this article, we assume all the classes are executed at the same time.

B. Computation model

At first, the routing delay of class i from UE to local UPF m at time slot t is expressed by:

$$\sum_{n=1}^{N^i} T^{i,m}_{UD^i_n,UPF^m}(t) = \frac{\sum_{n=1}^{N^i} D^{i,m}_n(t)}{R_{UD^i_n,UPF^m}(t)} \quad (1)$$

Which N^i is the total tasks' number of class i, $D^{i,m}_n(t)$ is the data size generated by task $n^{th} \in N^i$, sent to local UPF m, and $R_{UD^i_n,UPF^m}$ is the routing capacity from UD sending task n^{th} of class i to UPF m. We consider the links from UD to UPF are wireless, the link among UPFs and UPF to MECS are wireline. Hence, according to Shannon formular, the data transfer rate is obtained below:

$$R_{UD^i_n,UPF^m}(t) = Blog_2(1 + \frac{P_n(t)g_n(t)}{BN_o}) \quad (2)$$

With B is the channel bandwidth between UD and UPF, P_n is the transmission power to send task n^{th}, g_n represents the channel power gain from the UD to UPF, and $N_o = -174dBm$ [5] denotes the Gaussian noise power spectrum density.

After coming to UPF, all tasks of the class i will be locally executed or migrate to another optimal MECS according to controller's decision. These two modes are described below:

- Executing at the local MECS m $(A^{m,v}_i = 0)$

  When the class i is processed locally, the total delay $\sum_{n=1}^{N^i} T^{n,m}_i(t)$ consists of routing delay from UPF m to MECS m, $T^{i,m}_{UPF^m,MECS^m}$, and processing delay at the MECS m, $T^{i,m}_{MECS^m}$ as follows:

$$T^{i,m}_{UPF^m,MECS^m}(t) = \frac{\sum^{i,m}_n D^{i,m}_n(t)}{R_{UPF^m,MECS^m}(t)} \quad (3)$$

$$T^{i,m}_{MECS^m}(t) = \frac{\sum^{i,m}_n D^{i,m}_n(t).L^m_i(t)}{w^m_i(t).c^m(t)} \quad (4)$$

  With $R_{UPF^m,MECS^m}$ is the routing capacity from UPF m to MECS m, L^m_i is the number of CPU cycles of MECS m for processing 1 bit task of class i, w^m_i is the weight of computing resource c^m of MECS m that decided by MEC controller to process class i at time t.

- Migrating to the target MECS v $(A^{m,v}_i = 1)$

  The total delay $\sum_{n=1}^{N^i} T^{n,m,v}_i(t)$ in this mode include routing delay from UPF m to target UPF v, $T^{i,m}_{UPF^m,UPF^v}$, from UPF v to MECS v, $T^{i,m}_{UPF^v,MECS^v}$, and the processing delay at the MECS v, $T^{i,m}_{MECS^v}$ as follows:

$$T^{i,m}_{UPF^m,UPF^v}(t) = \frac{\sum^{i,m}_n D^{i,m}_n(t)}{R_{UPF^m,UPF^v}(t)} \quad (5)$$

$$T^{i,m}_{UPF^v,MECS^v}(t) = \frac{\sum^{i,m}_n D^{i,m}_n(t)}{R_{UPF^v,MECS^v}(t)} \quad (6)$$

$$T^{i,m}_{MECS^v}(t) = \frac{\sum^{i,m}_n D^{i,m}_n(t).L^v_i(t)}{w^v_i(t).c^v(t)} \quad (7)$$

  Which R_{UPF^m,UPF^v} is the routing capacity between UPF m to UPF v, $R_{UPF^v,MECS^v}$ is the routing

capacity from UPF v to MECS v, L_i^v is the number of CPU cycles of MECS v for processing 1 bit task of class i, w_i^v is the weight of computing resource c^v of MECS v that decided by MEC controller to process class i at time t.

C. Problem formulation

The objective of this work is to minimize the total delay (routing delay and processing delay) for local or remote processing of all classes made by tasks sending from UDs. The objective can be formulated below:

$$\begin{aligned}
&\min T \\
&= min \sum_{t=1}^{T} [\sum_{m=1}^{M} \sum_{\substack{v=1 \\ (v \neq m)}}^{M} \sum_{n=1}^{N^i} \sum_{i=1}^{4} (T_{UD_n^i, UPF^m}^{i,m}(t) \\
&+ (1 - A_i^{m,v}(t)).T_i^{n,m}(t) \\
&+ A_i^{m,v}(t).T_i^{n,m,v}(t)] \\
&= min \sum_{t=1}^{T} [\sum_{m=1}^{M} \sum_{\substack{v=1 \\ (v \neq m)}}^{M} \sum_{i=1}^{4} (\frac{\sum_{n=1}^{N^i} D_n^{i,m}(t)}{R_{UE_n^i, UPF^m}(t)} \\
&+ (1 - A_i^{m,v}(t)).(\frac{\sum_{n=1}^{N^i} D_n^{i,m}(t)}{R_{UPF^m, MECS^m}(t)} \\
&+ \frac{\sum_{n=1}^{N^i} D_n^{i,m}(t).L_i^m(t)}{w_i^m(t).c^m(t)}) \\
&+ A_i^{m,v}(t)(\frac{\sum_{n=1}^{N^i} D_n^{i,m}(t)}{R_{UPF^m, UPF^v}(t)} + \frac{\sum_{n=1}^{N^i} D_n^{i,m}(t)}{R_{UPF^v, MECS^v}(t)} \\
&+ \frac{\sum_{n=1}^{N^i} D_n^{i,m}(t).L_i^v(t)}{w_i^v(t).c^v(t)})] \qquad (8)
\end{aligned}$$

subject to:

$$T_{UD_n^i, UPF^m}^{i,m}(t) + (1 - A_i^{m,v}(t)).T_i^{n,m}(t) + A_i^{m,v}(t).T_i^{n,m,v}(t)) \leq T_{thres}^{i,n}(t) \qquad (9)$$

$$\sum_{i=1}^{4} w_i^m = 1; \sum_{i=1}^{4} w_i^v = 1 \qquad (10)$$

$$c^v \leq c_{max}^v; c^m \leq c_{max}^m \qquad (11)$$

$$c^v \geq 0;\ c^m \geq 0 \qquad (12)$$

$$A_i^{m,v} \in \{0, 1\} \qquad (13)$$

$$\sum_{m,v} A_i^{m,v} = 1 \qquad (14)$$

Constraint (9) ensures that total delay of task n^{th} of class i should meet its corresponding delay threshold. The constraint (10) means total weight at each MECS for four classes is equal 1. Constraint (11) and (12) state that computing resource allocated at each MECS at time t must be smaller than its maximum computing resource, and allocated computing resource for each MECS is positive, respectively. Constraint (13) represents that the processing decision is a binary objective. Constraint (14) means each class must be chosen to be processed on only one server at time t.

III. Methodology

This section transforms the formulated problem into Markov decision process and then solving it by applying Deep Reinforcement Learning.

A. Markov decision modelling

The Markov decision process can be defined by state space, action space, policy, and reward function as presented below:

- State space:

$$S(t) = \{i, D(t), c(t)\} \qquad (15)$$

- Action space:

$$A(t) = \{w(t)\} \qquad (16)$$

- Policy: The state transition probability $P\left(s(t+1)\middle|\left(s(t), a(t)\right)\right)$ represents the probability of $s(t+1)$ given $s(t)$ and selected action $a(t)$.

- Reward function: The reward value is the feedback from the environment after making an action decision. The optimization problem is to minimize the total service delay. However, the reward should be maximized. Hence, we define the reward function $R(t)$ as the reverse of objective function:

$$R(t) = \frac{1}{T(t)} \qquad (17)$$

B. DRL-based resource allocation

This work utilizes DDPG framework which includes 3 phases:

- Update critic network then get Q-value: minimize loss function between target and predicted Q-value.
- Update actor network then optimize the policy μ.

- Update target network.

IV. Experiments and results

In this section, we run simulation to evaluate the performance of the DDPG framework-based resource allocation.

A. Simulation setup

For the simulations, we take 30 user devices to generate tasks in the time slot t, the size gathered by these UDs has range of $[0.25, 5]$ Mb. Besides, the delay tolerance of each class is based on [4]. Detail parameters are shown in Table I.

Table I. Parameters setting

Parameters	Values
Number of UDs	30
Number of UPF/MECSs	5
CPU cycles per bit of MECS	[750,800]
Task size	[0.25, 5] Mb
Bandwidth between UD-UPF	120 MHz
Power transmission	5 dBm
Computing capacity of MECS	60 GHz

B. Results

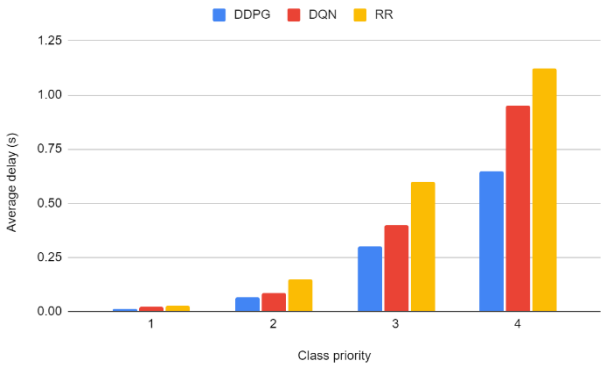

Figure 3. The different methods' average delay

Fig. 3 describes the average delays of the proposed architecture, DQN and RR methods. When using DDPG, the delay of each class not only satisfy the QoS delay threshold, but also smaller than the same class simulated by other methods.

V. Conclusion

In this article, we propose a MEC federation network based deep reinforcement learning to classify tasks come according to QoS levels and minimize the total delay of all QoS classes. In the future, we want to consider minimizing delay of classes separately and examine how minimize one class will affect to the others.

ACKNOWLEDGMENT

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0- 01015, Development of Candidate Element Technology for Intelligent 6G Mobile Core Network).

REFERENCES

[1] Yang, S., Lee, G., & Huang, L. (2022). Deep Learning-Based Dynamic Computation Task Offloading for Mobile Edge Computing Networks. Sensors, 22(11), 4088.

[2] Wang, Z., Li, P., Shen, S., & Yang, K. (2021). Task Offloading Scheduling in Mobile Edge Computing Networks. Procedia Computer Science, 184, 322-329.

[3] Wu, Y., Xia, J., Gao, C., Ou, J., Fan, C., Ou, J., & Fan, D. (2022). Task offloading for vehicular edge computing with imperfect CSI: A deep reinforcement approach. Physical Communication, 55, 101867.

[4] Wang, G., Xu, F., & Zhao, C. (2020). QoS-enabled resource allocation algorithm in internet of vehicles with mobile edge computing. IET Communications.

[5] Gupta, S., & Singh, N. (2022). Toward intelligent resource management in dynamic Fog Computing-based Internet of Things environment with Deep Reinforcement Learning: A survey. International Journal of Communication Systems.

.