Geographic Clustering Based Mobile Edge Computing Resource Allocation Optimization Mechanism

Song Kang, Linna Ruan, Shaoyong Guo*
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
{kangsong, linnaruan, syguo}@bupt.edu.cn

Wencui Li
China Electric Power Equipment and Technology Co. Ltd
Zhengzhou Electric Power Design Institute
Zhengzhou, China
elf8650@163.com

Xuesong Qiu
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
xsqiu@bupt.edu.com

***Abstract*—With the development of Internet of Things (IoT), a large number of terminals and devices are connected to the network. Mobile edge computing (MEC) is proposed to assist cloud computing, to relieve the pressure of network and satisfy the requirements of delay-sensitive applications. Considering reasonable allocation of computing resources is the most important aspect corresponding to delay, this paper designs geographic clustering and collaborative scheduling (GC-CS) mechanism. This mechanism can be divided into two parts, which are the decentralized deployment of MEC servers and the resource allocation optimization in MEC. For the first part, this paper designs the load balancing based geographic clustering (LBGC) algorithm which combines the idea of greedy algorithm to realize the initial allocation of computing resources. For the second part, delay minimization oriented collaborative scheduling (DMCS) algorithm is designed to decrease the response delay without increasing system overhead. Finally, the effectiveness of the mechanism is verified by simulation in the IoT scene.**

Keywords—mobile edge computing, Internet of Things, resource allocation

I. INTRODUCTION

Nowadays, IoT is widely used in many scenes which contain delay-sensitive applications, but the high response delay caused by long transmission link of traditional cloud computing can't meet the demand of them [1]. By deploying MEC servers, a portion of computing and storage capacity of data center is dropped to the edge of network to shorten the transmission link so as to reduce the delay [2]. However, MEC servers are limited by their performance, it is necessary to allocate the resources of servers to get high efficiency.

This paper mainly studies the geographic clustering and computing resource allocation of MEC. Among the existing strategies, [3] and [4] proposed two resource allocation strategies based on latency and density which neglect the load relationship between servers. [5] designed the AREA algorithm which compares predicted delay with actual, but its complexity was too high. To this end, this paper first divides server's responsibility scope by geographic clustering to allocate computing resource initially, which can reduce the complexity of mechanism and suits for IoT. Among existing algorithms, the HEED algorithm in [6] only considered energy consumption. In [7], a clustering method based on communication flow is proposed. And most clustering algorithms need to select the sink node [8], which weren't applicable to IoT scene.

978-3-903176-24-9©2019IFIP

Therefore, this paper proposes a geographic clustering algorithm to balance the load of servers and also designs a cooperation scheduling algorithm to minimize response delay. The rest of this paper is organized as follows: In Section II, MEC network structure and models used in the algorithms is proposed. In Section III, we propose the LBGC algorithm, and we design the DMCS algorithm in Section IV. In Section V, the performance of the proposed mechanism is verified by simulation. Finally, we conclude the paper in Section VI.

II. SYSTEM MODE

A. Scene Description

The IoT scene with MEC technology is shown in figure 1. Each MEC server covers such terminals. Although terminals have some computing capacity, since this paper focuses on response delay, we consider only the case that terminals offloading requests to servers. In order to achieve the dynamics of algorithm, this paper adopted dynamic time slot method, which means the length of each slot is different and determined by the results produced in this time slot.

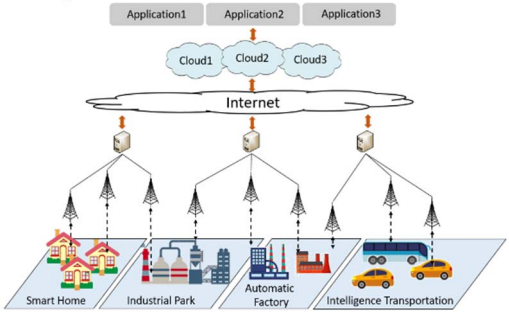

Fig.1. MEC network structure in IoT

B. Response Delay

Response delay usually consists of four parts: computing delay, transmission delay, propagation delay and queuing delay. However, the quality of resource allocation won't affect transmission delay [9], so it is ignored.

1) Propagation delay

The formula for propagation delay is:

$$t_{pro} = \frac{d}{v_{pro}} \quad , \tag{1}$$

where d represents the propagation distance, and v_{pro} is the network propagation speed between network nodes.

2) Computation delay

The computation formula of computation delay is:

$$t_{com} = \frac{F}{p \times v_{com}} \quad , \tag{2}$$

$$p = f_i / W_j \quad , \tag{3}$$

where W_j is the computation demand of terminal j, v_{com} is the computing processing speed, and p is the proportion of the capacity allocated by the server in computation demand.

3) Queuing delay

Queuing delay cannot be calculated easily and need the simulation according to the actual situation in the server' s requests queue to infer the queuing delay.

C. Service Consumption

It refers to the consumption generated by servers during configuration and computing, including energy and hardware loss, consisting of computation consumption, transmission consumption and configuration consumption [10].

1) Computation consumption

$$U_i^{op,t} = \sum_j x_{i,j}^t f_{i,j}^t a_{i,j} \quad . \tag{4}$$

In t slot, if terminal j is attributed to the MEC server i, $x_{i,j}^t$ is 1, $f_{i,j}^t$ represents the resource allocated to terminal j, and $a_{i,j}$ represents the computation consumption of unit computing capacity generated by server i to request j.

2) Transmission consumption

$$U_i^{tran,t} = (\sum_j x_{i,j}^t W_j^t p_j + \sum_j y_{i,j}^t W_j^t q_i) + (\sum_j x_{i,j}^t d_{i,j} b_j + \sum_j y_{i,j}^t D_{i,n} h_i) \tag{5}$$

where $y_{i,j}^t$ represents whether server i transfers the request of terminal j to others. W_j^t is the size of data of terminal j, p_j and q_i are transmission consumption of unit data from terminal j or server i. d_{ij} is distance between server and terminal, D_{in} is distance between servers, b_j and h_i are the propagation consumption of unit distance from terminal j and server i, respectively.

3) Configuration consumption

$$U_i^{con,t} = \sum_j l_i |x_{i,j}^t - x_{i,j}^{t-1}| \; x_{i,j}^t + \sum_j x_{i,j}^t f_{i,j}^t c_i \quad . \tag{6}$$

The two terms are reconfiguration and configuration consumption, l_i is the reconfiguration consumption of server i, and c_i is the configuration consumption of unit capacity.

D. System Overhead and Load

If we only consider the minimization of response delay, it may lead to the excessive increase of service consumption, so this paper integrates delay and consumption with different weight parameters, and then we get the system overhead:

$$E = \beta \times t_d + \gamma \times U \quad , \tag{7}$$

$$\beta + \gamma = 1, \quad 0 \le \beta, \gamma \le 1 \quad , \tag{8}$$

where β and γ are two weight parameters of overhead. The units of most above parameters are unified, which are defined according to their importance.

Balancing load can also promote system on overhead. In this paper, the load of servers is balanced by geographic clustering, which can provide a basis for the optimization of resources allocation. We define load as the product of the distance and computation demanded [11].The formula of load from terminal and load condition of server are:

$$L_{i,j} = d_{i,j} W_j \quad , \tag{9}$$

$$L_i = C_i R_{i,k} - \sum_j x_{i,j} L_{i,j} \quad . \tag{10}$$

In the above two equations, C_i and $R_{i,k}$ represent the capacity and current cluster radius of MEC server i.

III. Load Balancing Based Geographic Clustering Algorithm

As mentioned above, the main purpose of geographic clustering is to divide the responsible area of each MEC server [7]. Resource allocation is a NP hard problem, since this paper focuses on delay, we choose greedy algorithm which can get a satisfactory solution with low time complexity [12].

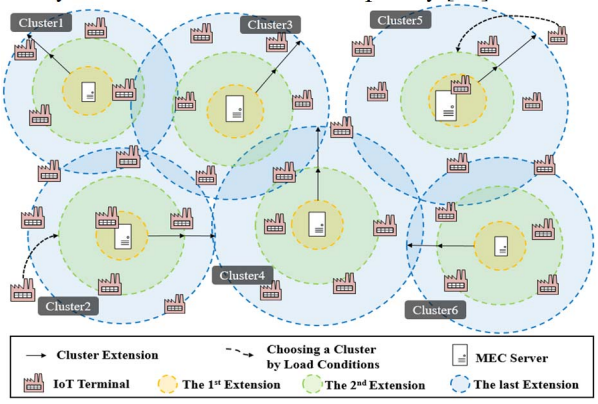

Fig.2. Schematic diagram of LBGC algorithm process

In consideration of the requirements of geographic clustering, the greedy algorithm is planned, with the shorter distance as its selection criteria, and MEC servers as the centers of the gradual expansion. Besides, it is specified that the reselection and missed terminals selects servers based on their load condition. The process is shown in figure 2, and the specific LBGC algorithm is shown in algorithm 1.

Algorithm 1 LBGC Algorithm

Initialization: Set the MEC server set S, the terminal set L_N of each MEC server. Set terminal set P which contains all terminals. Set the maximum expansion times of each cluster to K.
for $k<K$ **do**
  for Each server $i \in S$ **do**
    if its remaining resources > *0* **and** cluster i can expand **then**
      Expand the scale of **c**luster i ;
    end if
    for Each terminal $j \in P$ **do**
      if terminal j is in this scale **and** belongs to no cluster **then**
        Add terminal j to a temporary set *Q1* ;
      else if terminal j is in this scale **and** belongs to a cluster **then**
        Add terminal j to a temporary set *Q2* ;
      end if
    end for
    Sort the terminals in the set *Q1* **and** *Q2* by distance ;
    for Each terminal $q1 \in Q1$ **do**
      if its demand < remaining resource of server i **then**
        Add terminal $q1$ to L_i ;
      end if
    end for

for Each terminal $q2 \in Q2$ **do**
  if the load of server i is better than terminal $q2$'s server **then**
   Add terminal $q2$ to L_i ;
  end if
 end for
end for
end for
for Each terminal $j \in P$ **do**
 if terminal j does not belong to any cluster **then**
  Select the server i which has the best Load condition;
 end if
end for

In algorithm 1, the expansion of cluster radius is:

$$R_{i,0} = \sqrt{\alpha \frac{C_{i,0}}{C_i}} R_i' \quad , \tag{11}$$

$$R_{i,k} = \frac{C_{i,k}}{C_i} R_{i,(k-1)} + R_{i,(k-1)} \quad (1 \le k \le K) \quad . \tag{12}$$

Among these, $i \in S$, R_{i0} refers to the initial radius of cluster i, C_{i0} is the remaining computing capacity of server i after the previous slot, C_i is its total capacity, R_i' is the final radius in previous slot, and R_{ik} is the radius after the kth expansion. In the process of clustering, there are two constraints:

$$C_i \ge \sum_{j=1}^{M} x_{i,j} z_{i,j} W_j \quad , \tag{13}$$

$$R_i \le \min\{d_{i,j} \mid j \in S, j \ne i\} \quad . \tag{14}$$

In formula (13), if terminal j is included in cluster i during expansion, it is considered that the request offloaded by terminal j has priority to be processed as the positive request, then $z_{ij} = 1$. And alternative requests are requests that assigned to clusters in make-up selection stage. Formula (14) means that the radius of cluster i less than the distance of its server with other servers.

IV. Delay Minimization Oriented Collaborative Scheduling Algorithm

In the case where computation demand is unevenly distributed, several servers may cover too many terminals, so further optimization is required. Previously, requests are divided into positive and alternative ones. This paper stipulates that all alternative requests can only be allowed to enter the request queue only if they won't affect unreached positive requests and the remaining capacity are sufficient to support their computing processing.

The schematic diagram of MEC business process is shown in figure 3. The idea of DMCS algorithm is to get requests with insufficient resources, then transfer them to idle servers, so as to reduce the maximum delay of system. If propagation speed between servers is fast enough, the propagation delay can be neglected. The objective function and constraints of DMCS algorithm are shown in formula (15) and (16).

$$\min_{t_i} \max\{t_i \mid i \in S\} \quad , \tag{15}$$

$$\text{s.t. } E_{before} \le E_{after} \quad . \tag{16}$$

t_i is the expected delay of the last completed request of server i, E_{before} is the overhead after geographic clustering, and E_{after} refers to overhead after collaborative scheduling. The specific process of DMCS algorithm is as follows:

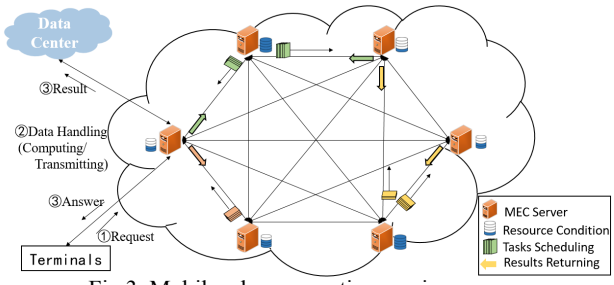

Fig.3. Mobile edge computing service process

Algorithm 2 DMCS Algorithm

Initialization: Set the MEC server set S, estimate the response delay of each server by simple simulation, and set *flag=1*.
while *flag* **do**
  Set *flag=0* ;
  Sort the MEC servers in the set S by response delay ;
  Get the server n which has the longest response delay Tm ;
  Get the terminal p which offloads the last request finished in server n, and get request p, set the length of the slot $T=Tm$;
  if the request p has been scheduled **then** break;
  end if
  if at least one server can reduce the delay of request p by way of transition and the overhead will not increase **then**
   Find server i which can reduce the most delay ;
   Pass the request p to server i ;
   Estimate the delay of each server by simple simulation ;
   Set *flag=1* ;
  end if
end while

The response delay is estimated by the simulation according to the processing order set above, which is not special. In addition, when scheduling, make sure:

$$\min\{t_v' \mid v \in S|i\} < t_i \quad , \tag{17}$$

$$E'_{\arg\min\{t_v' \mid v \in S|i\}} + E_i' \le E_{\arg\min\{t_v' \mid v \in S|i\}} + E_i \quad . \tag{18}$$

Formula (17) represents that the lowest value of each severs' maximum response delay after scheduling must be less than the maximum delay before scheduling. Formula (18) indicates that the total cost of edge computing system is not allowed to increase after the request be forwarded.

V. Simulation and Results

A. Description of the simulation scene

In this section, the feasibility and effectiveness of the proposed mechanism are verified by simulating the whole process with C++ language. The data sources in the simulation are real data from IoT, as the data is too complex for scene simulation, only three kinds of terminals with distinct characteristics are selected, which are life terminals, traffic terminals and production & energy terminals.

The scene consists of 3000 terminals and 9 MEC servers, the number of 3 types of terminals is 1400, 1000 and 600. In this paper, CPU cycles/s is used as the unit of resource. The ratio of computing demand of 3 terminals types is 2:3:6, and computing capacity of this 9 MEC servers are set as {20, 20, 20, 35, 20, 25, 25, 35, 25}×10^9 CPU cycles/s. The average propagation speed between servers is set as 6×10^4m/s, and 3×10^3m/s between servers and terminals. The other parameters are set as α =0.8, β =0.2, a =0.05/(CPU cycles/s), p=q=0.001/GB, b=h=0.002/m, l=0.02, c=0.03/(CPU cycles/s). The geographic distribution is shown in figure 4, these red, blue, and green points represent 3 types of terminals, and diamonds are MEC servers.

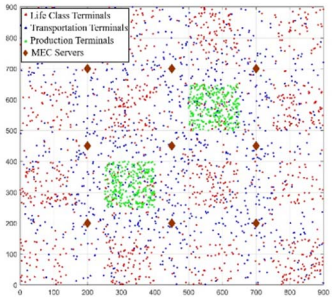

Fig.4. Geographic distribution of terminals (uniform)

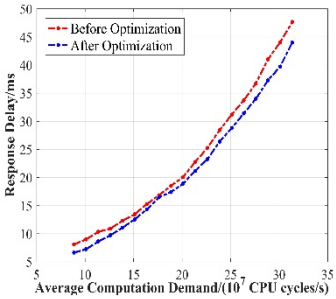

Fig.5. Comparison of response delay (uniform)

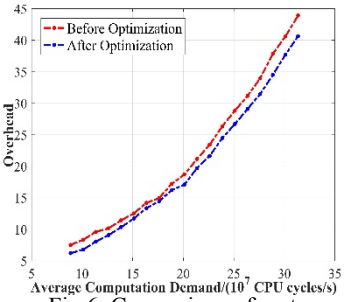

Fig.6. Comparison of system overhead (uniform)

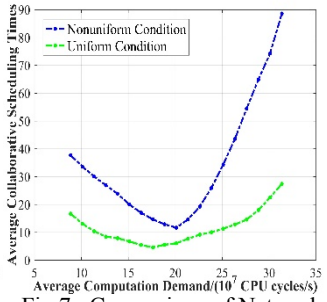

Fig.7. Comparison of Network collaborative scheduling times

B. *Simulation of uniform distribution*

The process within a time slot is divided into before and after optimization for comparison. The results are shown in figure 5 and 6. It can be seen that the maximum response delay increases exponentially, and the tend of overhead is similar to it. We can see that the effect of DMCS algorithm decreases first and then increases. When the demand is about 17.6, the optimization effect is the least obvious, and the average times of scheduling achieves the minimum value as shown in the green curve of figure 7, DMCS algorithm plays the least. This is because the response delay of each servers is basically similar, the load balancing effect of the LBGC algorithm is maximized.

C. *Simulation of non-uniform distribution*

It is assumed that the surge of computation demand will make the distribution chaotic, leading to unreasonable clustering and increasing the demand for network scheduling. In order to verify it, this paper sets the non-uniform distribution condition, it is shown in figure 8.

The result is shown in figure 9, 10 and blue curve in figure 7. It can be seen that the optimization effect is more obvious after network scheduling in this case, and the number of scheduling times is higher than uniform case. So GC-CS mechanism has a good emergency effect for emergencies.

D. *Effect analysis*

In order to further prove the effectiveness of the GC-CS mechanism, Latency based strategy and density based clustering strategy (DBC) are selected for comparison. Figure 11 to 13 show the changes of delay with request arrival rate, the number of terminals and computing capacity of MEC servers in the particular scene. The response delay of GC-CS mechanism is generally smaller and has better effectiveness. Besides, we can see that it changes more gently and has better stability,

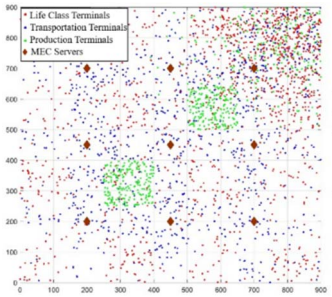

Fig.8. Geographic distribution of terminals (non-uniform)

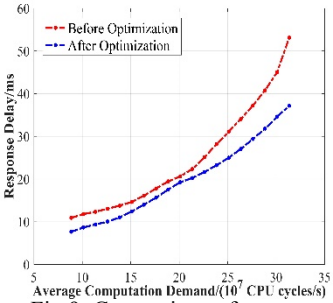

Fig.9. Comparison of response delay (non-uniform)

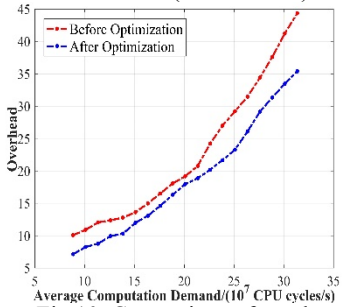

Fig.10. Comparison of overhead (non-uniform)

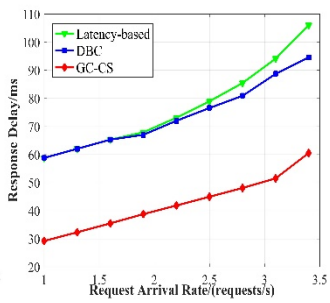

Fig.11.Comparisons of delay with arrival rate between 3 strategies

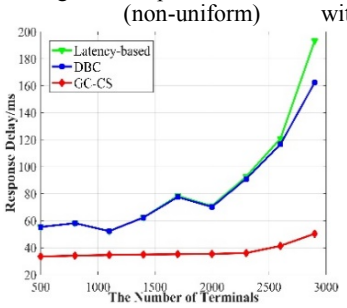

Fig.12. Comparisons of delay with the number of terminals between 3 strategies

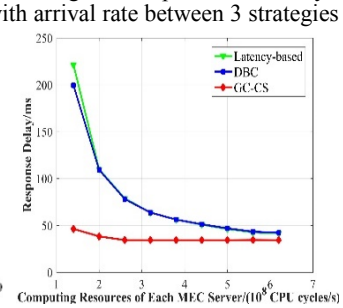

Fig.13. Comparisons of delay with the resource of servers between 3 strategies

which further proves that GC-CS mechanism has high effectiveness in performance optimization.

VI. CONCLUSION

In order to realize the optimization of computing resource allocation of MEC servers in the IoT scene, this paper designs the GC-CS mechanism which contains LBGC algorithm and DMCS algorithm with lower complexity. The improvement of load balancing and delay minimization by our mechanism are verified by simulation. We find that our goals can be realized in the case of both uniform and non-uniform distribution of terminals, especially for the second case, the delay can be reduced more effectively. In the future work, we will take application types into consideration, to improve the establishment of IoT model, making the design of resource allocation mechanism more suitable for the practical scene.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant: 61702048) and Construction of Public Support Platform for Standard Management Service of Industrial Internet Platform.

REFERENCES

[1] Y. Mao, C. You and Z. Jun, etc., "A Survey on Mobile Edge Computing: The Communication Perspective," IEEE Communications Surveys & Tutorials. Vol.19, pp. 2322–2358, 2017.

[2] A. C. Baktir, A. Ozgovde, C. Ersoy, "How Can Edge Computing Benefit from Software-Defined Networking: A Survey, Use Cases & Future Directions," IEEE Communications Surveys & Tutorials, Vol.10, pp. 1-34, 2017.

[3] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware service placement and load dispatching in mobile cloud systems," IEEE Trans. Comput., vol. 65, no. 5, pp. 1440–1452, May 2016.

[4] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," IEEE Trans. Cloud Comput., vol. 5, no. 4, pp. 725–737, Oct./Dec. 2017.

[5] Q. Fan, N. Ansari, "Application Aware Workload Allocation for Edge Computing-Based IoT," vol.5, no.3, 2146-2153, Jun.2018.

[6] J. Qu, H. Lv, J. Zhao, "HEED N0n-uniform Clustering Alogrithm Based on Geographic Location," Journal of Shanghai Dianji University, vol.19, no.13, pp.170-175, 2016.

[7] M. Bouet, V. Conan, "Mobile Edge Computing Resources Optimization: A Geo-Clustering Approach," IEEE Transactions on Network and Service Management. vol.15, no.2, pp.787-796, Jun.2018.

[8] M. Zhang, C. Gong, Y. Lu, "An Novel Dynamic Clustering Algorithm based on Geographical Location for Wireless Sensor Networks," 2008 International Symposium on Information Science and Engieering, pp.565-568, 2008.

[9] C. Meng, "Research on the Principle of Data Center Service Resource Scheduling in Edge Computing," Beijing University of Posts and Telecommunications, Jan. 2018.

[10] L. Wang, L, Jiao, J. Li, etc, "Online Resource Allocation for Arbitrary User Mobility in Distributed Edge Clouds," IEEE International Conference on Distributed Computing Systems, vol.37, pp.1281-1290, 2017.

[11] Q. Dong, Y. Ma, J. Li, "Load balancing oriented scheduling scheme in edge computing network," Application Reasearch of Computers, vol.37, no.3, pp.1-6, 2018.

[12] L. He, P. Le, "Design and implementation of greedy optimization algorithm in resource allocation," Journal of Geomatics, vol.35, no.2, pp.3-7, Apr.2010.