A Computation Offloading Mechanism Based on Sharable Cache in Smart Community

1st Yixin Zhang
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
zhangyixin_1030@163.com

2nd Yong Yan
State Grid Zhejiang Electric Power Co., LTD
Zhejiang, China
55682381@qq.com

3rd Yang Yang*
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
yyang@bupt.edu.cn

4th Zhipeng Gao
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
gaozhipeng@bupt.edu.cn

5th Xuesong Qiu
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
xsqiu@bupt.edu.cn

***Abstract*—In the smart community, many smart devices often offload computation tasks that use related data to edge servers. In this case, unnecessary data transmissions will occur, resulting in waste of energy resources and increased latency. Therefore, we consider adding a shareable cache to the edge server, allowing different computing tasks to share the original data file of computation task, and further reducing the energy consumption and latency. In this paper, we propose a computation offloading mechanism based on edge server shareable cache. We first propose a four-tier architecture. Subsequently, a resource constraint optimization model combining communication, computing, and caching is established. Then, we design a multi-factor-based edge server sharable cache management and update mechanism (MFBM), and propose a two-scenario computation offloading mechanism (TSCOM). Finally, simulation results show that the mechanism proposed in this paper is effective.**

Keywords—edge computing, computation offloading, smart community, IoT service, cache

I. INTRODUCTION

Edge computing has become a key technology to promote the development of smart communities. In smart community, many smart devices often offload computation tasks that use related data to edge servers. These tasks may be generated from various IoT applications in the smart community, including smart vehicles, electronic health services, interactive games and virtual reality applications. When the data used by the computation tasks of multiple smart devices are related, repeated data transmissions will occur, resulting in waste of energy resources and increased latency.

Therefore, we consider adding a shareable cache to the edge server, allowing different computation tasks to share the original data file, and further reducing the energy consumption and latency of data transmission. We first propose a four-tier architecture. Subsequently, an optimization model combining communication, computing, and caching is established. Then, we design a multi-factor-based edge server sharable cache management and update mechanism, which solves the issues raised above, and propose a two-scenario computation offloading mechanism. Finally, we prove the effectiveness of the proposed mechanism through simulation.

The remainder of this paper is organized as follows. Section II sorts out some related works, Section III presents the system model and the problem formulation. Section IV proposes the mechanism. Simulation results are discussed in Section V. Finally, Section VI concludes the paper.

II. RELATED WORKS

There have been some studies on computation offloading [1-3]. In terms of edge caching, existing studies focus more on the strategy of caching popular cloud content in edge servers [4-5], and the goal is to reduce the load of cloud servers, reduce user access delays, and avoid network congestion, but it ignores that computation task data caching actually has a wider and common application scenario. A few studies have considered caching computation task data in edge servers [6-7], but few consider the collaboration between edge servers. At the same time, the caching strategy will affect the offloading decision and resource allocation, and vice versa. In order to improve the efficiency of the system, the combination of caching and computation task offloading is essential.

However, the management and update strategy of the cache is a complex problem, and there are many challenges to be solved [8]. First of all, although the computing and caching capabilities of edge servers are better than local smart devices, they are still limited. Therefore, we need to consider efficient use of limited resources. Secondly, deciding whether to cache a data file depends not only on the frequency of the data file being accessed, but also on the size and complexity of the task, etc. It is difficult to find a general caching method. Finally, whether a data file is worth caching is likely to be affected by time, but many papers ignored this point.

III. SYSTEM MODEL AND THE PROBLEM FORMULATION

A. System overview

Fig. 1. shows the four-layer smart community edge computing architecture proposed in this article. The set of edge servers is denoted as $\mathrm{N} = \{1,2, \dots, n, \dots, N\}$, and the set of smart devices (SDs) that connected to edge server n is denoted as $\mathrm{M}_n = \{1,2, \dots, m_n, \dots, M_n\}$. The set of computation tasks in this model is $\mathcal{A} = \{A_1, A_2, \dots, A_i, \dots, A_I\}$. A computation task is characterized by three parameters, the data size of the computation task D_i, the computational density required for the task X_i (cycles/bit), and the maximum tolerance latency T_i^{max} , and the computation task is denoted as $A_i = \{D_i, X_i, T_i^{max}\}$. We assume that there is a certain probability of each task requested by SD m_n, and the probability is denoted as $\lambda_{i,m,n}$, where $\sum_{i=1}^{I} \lambda_{i,m,n} = 1$.

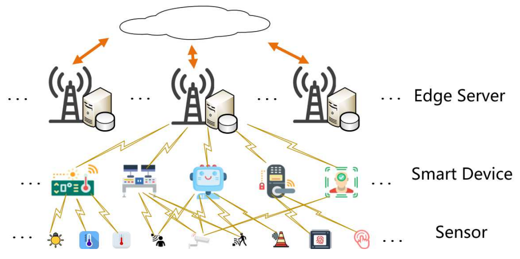


Fig. 1. System model

B. Communication model

In this paper, the uplink transmission rate of m_n is $r_{m,n}$ [8], and it can be obtained as

$$r_{m,n} = B\log_2\left(1 + \frac{g_{m,n}p_{m,n}}{\sigma^2}\right) \quad (1)$$

where B is the bandwidth, $p_{m,n}$ is the transmission power of m_n, $g_{m,n}$ is the channel gain between m_n and n, and σ^2 is the noise power. To simplify the model, we assume that the bandwidth is identical for all the channels.

C. Computation model

(1) Local computing

The local energy consumed by local computing is

$$e_{m,n,i}^{l} = \kappa\left(f_{m,n}^{l}\right)^2 D_i X_i \quad (2)$$

where $f_{m,n}^{l}$ is the computation capability of SD m_n, κ is a coefficient of capacity related to the chip architecture, and its value is usually taken as 10^{-25} [9].

The execution time of A_i can be calculated as

$$t_{m,n,i}^{l} = \frac{D_i X_i}{f_{m,n}^{l}} \quad (3)$$

(2) Edge computing

The process of computation offloading may be consist of three parts: task offloading from SD to edge server, execution on edge server, and collaboration between edge servers. The latency of task offloading from SD m_n to edge server n is

$$t_{m,n,i}^{c-off} = \frac{D_i}{r_{m,n}} \quad (4)$$

At this time, the energy consumption of task offloading from SD n can be defined as

$$e_{m,n,i}^{c-off} = p_{m,n}\frac{D_i}{r_{m,n}} \quad (5)$$

where $p_{m,n}$ is the transmission power of m_n. We will ignore the latency and energy consumption of downloading [10].

In order to improve the efficiency of computing offloading, edge servers can collaborate with each other. For a computation task A_i of SD m_n, it can be executed at the edge server n, which is connected by SD m_n, and it can also be executed by adjacent edge servers that cooperate with n. We denote the edge server executing this task as n', then the execution time can be given as

$$t_{m,n,i,n'}^{c-exe} = \frac{D_i X_i}{f^c} \quad (6)$$

Where f^c is the computation capability allocated by edge server for computation task.

The energy consumption of task execution can be represented as

$$e_{m,n,i,n'}^{c-exe} = \kappa(f^c)^2 D_i X_i \quad (7)$$

If the task offloaded to the edge server n needs to be executed on the edge server n', then the transmission delay from n to n' is

$$t_{n,n',i}^{c-tr} = D_i \zeta_{n,n'} \quad (8)$$

where $\zeta_{n,n'} = \tau_0 d_{n,n'} + \tau_1$. $\zeta_{n,n'}$ is the transmission latency per unit of computation task, $d_{n,n'}$ is the distance between edge server n and n', τ_0 and τ_1 are linear coefficients. Similar to many existing studies, we assume that as the transmission distance increases, the latency increases linearly.

The energy consumption caused by the transmission between cooperative edge servers can be expressed as

$$e_{n,n',i}^{c-tr} = D_i \varepsilon_{n,n'} \quad (9)$$

where $\varepsilon_{n,n'}$ is the energy coefficient between edge servers.

D. Problem formulation

This paper aims to minimize the total energy consumption of the system, and under the constraints of delay, limited computation and cache capacity, etc., we intend to study a collaborative computation offloading strategy combined with cache resource allocation.

First, the offloading decision variable and cache state variable are defined as follows. We define cache state variable as $\alpha_{i,n} \in \{0,1\}$, it indicates whether the data file of task A_i is cached at edge server n ($\alpha_{i,n} = 1$) or not ($\alpha_{i,n} = 0$). We also define offloading decision variable as $\beta_{i,m}^{n,n'} \in \{0,1\}$, it indicates weather edge server n' is chosen by SD m_n to execute computation task A_i ($\beta_{i,m}^{n,n'} = 1$) or not ($\beta_{i,m}^{n,n'} = 0$).

If a computation task A_i of SD m_n is executed locally, then the latency and energy consumption can be expressed as

$$t_{m,n,i}^{l} = (1 - \sum_{n'=1}^{N} \beta_{i,m}^{n,n'}) t_{m,n,i}^{l} \quad (10)$$

$$e_{m,n,i}^{l} = (1 - \sum_{n'=1}^{N} \beta_{i,m}^{n,n'}) e_{m,n,i}^{l} \quad (11)$$

If a computation task A_i of SD m_n is executed at an edge server, then the latency and energy consumption can be expressed as

$$t_{m,n,i}^{c} = \sum_{n'=1}^{N} \beta_{i,m}^{n,n'} \{(1 - \alpha_{i,n})(t_{m,n,i}^{c-off} + t_{n,n',i}^{c-tr}) + t_{m,n,i,n'}^{c-exe}\} \quad (12)$$

$$e_{m,n,i}^{c} = \sum_{n'=1}^{N} \beta_{i,m}^{n,n'} \{(1 - \alpha_{i,n})(e_{m,n,i}^{c-off} + e_{n,n',i}^{c-tr}) + e_{m,n,i,n'}^{c-exe}\} \quad (13)$$

In summary, the overall energy consumption of the system is

$$e = \sum_{i=1}^{I} \sum_{n=1}^{N} (e_{m,n,i}^{l} + e_{m,n,i}^{c}) \quad (14)$$

To formulate the problem, we can build a model as follows

$$\underset{m,n,i}{\text{minimize}}\ e \quad (15)$$

s.t. $C1$: $t_{m,n,i}^{l} + t_{m,n,i}^{c} \le T_i^{max},\ i \in I, n \in N, m_n \in M_n$

$C2$: $\sum_{i=1}^{I} \sum_{n=1}^{N} \sum_{m=1}^{M} \beta_{i,m}^{n,n'} \lambda_{i,m,n} f^c \le F_{n'}, i \in I, n \in N, m_n \in M_n$

$C3$: $\sum_{i=1}^{I} \alpha_{i,n} D_i \le C_n,\ i \in I, n \in N, m_n \in M_n$

$C4$: $\sum_{n'=1}^{N} \beta_{i,m}^{n,n'} \le 1,\ i \in I, n \in N, m_n \in M_n$

$C5$: $\sum_{i=1}^{I} \lambda_{i,m,n} = 1,\ i \in I, n \in N, m_n \in M_n$

Constraint *C1* indicates the execution time of the task should less than its maximum tolerance latency. Constraint *C2* guarantees that the allocated computational capability cannot exceed the upper limit of the edge server. In constraint *C3*, C_n is the maximum cache size of edge server *n*, and *C3* is to ensure that the size of the cache space occupied by data files cannot exceed the upper limit of the edge server capacity. *C4* implies that each task can only choose one offloading mode. *C5* is the limit on the probability of each task.

IV. Collaborative Computation Offloading Strategy Combined With Cache Resource Allocation

A. *Multi-factor-based cache management and update mechanism (MFBM)*

In order to reduce unnecessary data transmission and improve utilization efficiency of limited cache space, this paper designs a multi-factor-based edge server sharable cache management and update mechanism (MFBM).

Step 1. In the initial stage, when the sharable cache space of the edge server is not full, data files can be stored until the limit of sharable cache space is reached. Step 2. At the same time, record the access frequency (AF) of each time slot for each data file. Step 3. Use the idea of exponentially weighted moving average (EWMA) to obtain the EWMA value of AF for each data file. The weighted influence of each value decreases exponentially with the passage of time. The closer the time is to the current moment, the greater the weighted influence of the data. Since the weight of the data is smaller as time goes forward, we only consider the data of the latest $\frac{1}{1-\xi}$ time slots to calculate the $EWMA_t$ of the current time slot. At the same time, define a deviation to correct the cold start problem. When *t* is very small, the estimation at the initial stage can be more accurate. When *t* is very large, the deviation correction is almost ineffective. We define the corrected EWMA value as $EWMA_t{}'$.

$$EWMA_t = \xi \cdot EWMA_{t-1} + (1-\xi) \cdot AF_t = (1-\xi) \cdot [AF_t + \xi \cdot AF_{t-1} + \xi^2 \cdot AF_{t-2} + \ \dots + \xi^{t-1} \cdot AF_1] \qquad (16)$$

$$EWMA_t{}' = \frac{EWMA_t}{1-\xi^t} \qquad (17)$$

Step 4. Calculate the Popularity value of the current moment for each cached data file. Popularity is an indicator defined in this paper to evaluate whether a file should be cached, defined as *Pop*. It consists of the $EWMA_t{}'$, the size of the data file, and the computation complexity.

$$Pop_t = \omega_1 Nor(EWMA_t{}') + \omega_2 Nor(D_i) + \omega_3 Nor(X_i) \qquad (18)$$

where ω_1, ω_2, ω_3 are weights of $EWMA_t{}'$, the size of the data file, and the computational density, and $\omega_1 + \omega_2 + \omega_3 = 1$. The weights are given by [11].

Due to the large difference between the various parameters, we need to normalize them. The min-max normalization function is denoted as Nor(), and is defined as

$$\mathrm{Nor}(x) = \frac{x - min(X)}{max(X) - min(X)} \qquad (19)$$

where x is the value to be normalization, $min(X)$ and $max(X)$ mean the minimum and maximum value of set X.

Step 5. For data files that are not cached, create a table to record their AF, and calculate their Popularity. Step 6. Data file with a larger Popularity is cached, and data file with a smaller Popularity is replaced.

B. *Two-scenario computation offloading mechanism (TSCOM)*

In the previous section, we studied the management and update mechanism of the sharable cache on the edge server, and the computation offloading strategy will be affected by this mechanism. If the data file required by a computation task has been cached in the sharable cache of the edge server, then we believe that the computation task will be executed on the edge server if the edge server's computation capacity allows. Because this will effectively reduce local energy consumption and shorten response time, and avoid the waste of data files cached at the edge but no SD access. It is worth noting that this setting will in turn affect the results of the caching strategy. Therefore, we propose a two-scenario computation offloading mechanism (TSCOM).

The first scenario is, when the data file required by the computation task A_i of a SD m_n has been cached on the edge server, the problem can be transformed into

$$e^{c}_{m,n,i} = e^{c-exe}_{m,n,i} \qquad (20)$$

$$t^{c}_{m,n,i} = t^{c-exe}_{m,n,i} \qquad (21)$$

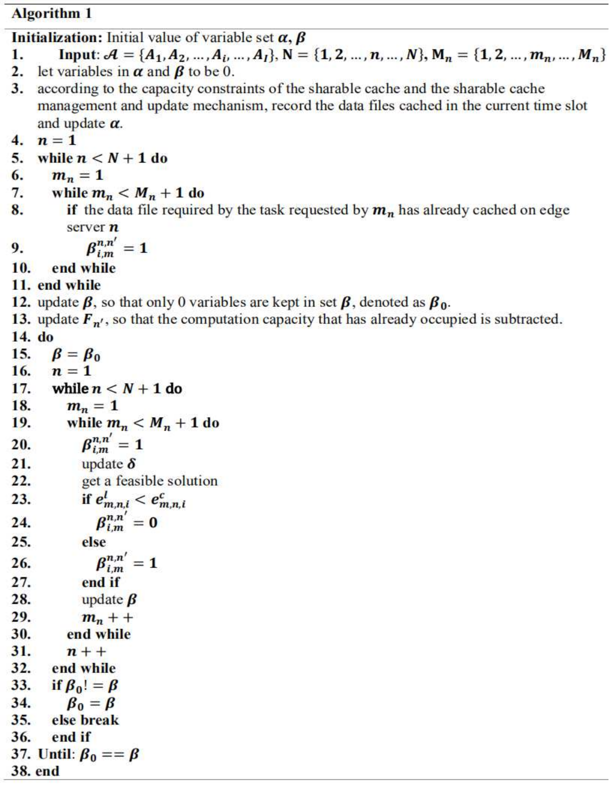
Since this part of the current time slot is fixed and the most efficient, we only need to focus on other computing tasks that data files are not cached by the edge server. The second scenario can be transformed into a problem with only one decision variable.

$$e^{c}_{m,n,i} = \sum_{n'=1}^{N} \beta^{n,n'}_{i,m} \{e^{c-off}_{m,n,i} + e^{c-tr}_{n,n',i} + e^{c-exe}_{m,n,i,n'}\} \qquad (22)$$

$$\text{minimize } e = \sum_{i=1}^{I} \sum_{n=1}^{N} (e^{l}_{m,n,i} + e^{c}_{m,n,i}) \qquad (23)$$

s.t. *C1, C2, C4, C5.*

where C1 should be updated, so that the computation capacity of F_n that has already been occupied is subtracted.



Algorithm 1

```
Initialization: Initial value of variable set $\alpha, \beta$
1.      Input: $\mathcal{A} = \{A_1, A_2, \dots, A_i, \dots, A_I\}$, $N = \{1, 2, \dots, n, \dots, N\}$, $M_n = \{1, 2, \dots, m_n, \dots, M_n\}$
2.  let variables in $\alpha$ and $\beta$ to be 0.
3.  according to the capacity constraints of the sharable cache and the sharable cache
    management and update mechanism, record the data files cached in the current time slot
    and update $\alpha$.
4.  $n = 1$
5.  while $n < N + 1$ do
6.    $m_n = 1$
7.    while $m_n < M_n + 1$ do
8.      if the data file required by the task requested by $m_n$ has already cached on edge
        server $n$
9.        $\beta^{n,n'}_{i,m} = 1$
10.   end while
11. end while
12. update $\beta$, so that only 0 variables are kept in set $\beta$, denoted as $\beta_0$.
13. update $F_{n'}$, so that the computation capacity that has already occupied is subtracted.
14. do
15.   $\beta = \beta_0$
16.   $n = 1$
17.   while $n < N + 1$ do
18.     $m_n = 1$
19.     while $m_n < M_n + 1$ do
20.       $\beta^{n,n'}_{i,m} = 1$
21.       update $\delta$
22.       get a feasible solution
23.       if $e^{l}_{m,n,i} < e^{c}_{m,n,i}$
24.         $\beta^{n,n'}_{i,m} = 0$
25.       else
26.         $\beta^{n,n'}_{i,m} = 1$
27.       end if
28.       update $\beta$
29.       $m_n{+}{+}$
30.     end while
31.     $n{+}{+}$
32.   end while
33.   if $\beta_0 != \beta$
34.     $\beta_0 = \beta$
35.   else break
36.   end if
37. Until: $\beta_0 == \beta$
38. end
```

V. Simulation and Analysis

A. Simulation Scenario

We consider the scenario has 3 edge servers, 10 task types, and 50 SDs connected to each edge server. We set the linear coefficients for transmission delay τ_0 as $5ms/km$ and $\tau_1 = 19.7ms$ [12]. Remaining parameters are shown in Table 1.

TABLE I. Parameters

parameter	reference value	parameter	reference value
background noise	-174dBm/Hz	B	20MHz
p	[0,100]mw	f^l	[0.5,0.9]GHz
F	10GHz	D_i	[500k,1M] bits
X_i	[800,1000] cycles/bit	$\varepsilon_{n,n'}$	2×10^{-8}J/bit

B. Result Analysis

This section evaluates the proposed mechanism (MFBM+TSCOM) from the perspective of energy consumption. We compare mechanism with four techniques: 1) no cache and TSCOM, 2) least recently used caching algorithm (LRU) and TSCOM [13], 3) MFBM and all executed at the edge (all edge) [8], 4) MFBM and all executed locally (all local) [8].

Fig.2 shows that with the increasing number of smart devices, the energy consumption of the proposed mechanism is always lower than other techniques. No cache and LRU can be regarded as the comparison mechanism of MFBM, and all edge and all local can be regarded as the comparison mechanism of TSCOM. The experimental results not only prove the effectiveness of the overall mechanism proposed in this paper, but also prove the respective roles of MFBM and TSCOM. For each mechanism, add up the energy consumption, the proposed mechanism is 40.08%, 10.96%, 31.57% and 32.52% lower than technique1, 2, 3 and 4.

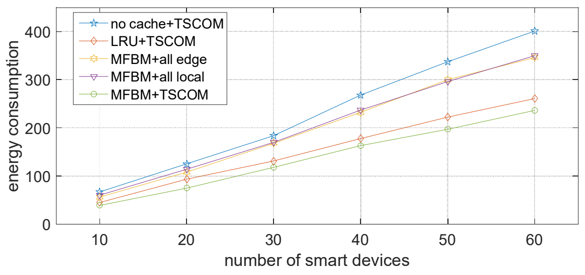


Fig. 2. Energy consumption as the number of smart devices increases

Fig.3 shows that as the average data size of computation task changes, the proposed mechanism can still obtain the minimum energy consumption. For each mechanism, add up the energy consumption, the proposed mechanism is 24.15%, 7.69%, 13.87% and 19.18% lower than technique1, 2, 3, 4.

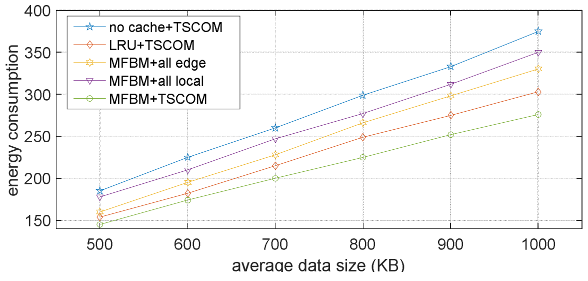


Fig. 3. Energy consumption as the average data size increases

Fig.4 shows that with the increasing of average computation complexity, the energy consumption of the proposed mechanism is always lower than other techniques. For each mechanism, add up the energy consumption, we can get that the proposed mechanism is 14.16%, 6.75%, 8.84% and 17.69% lower than technique1, 2, 3 and 4. This performance will become more significant as the average computation complexity increases.

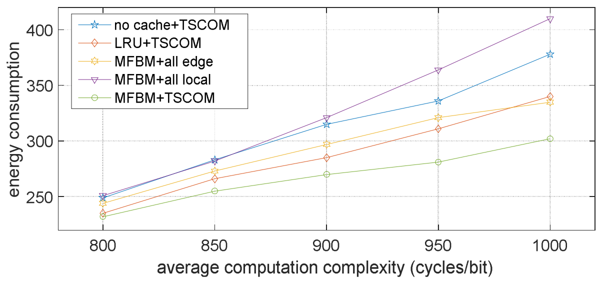


Fig. 4. Energy consumption as the computation complexity increases

Fig.5 shows that as the size of edge cache space increases, the energy consumption decreases, and the energy consumption related to proposed mechanism decreases significantly. When the edge cache space size is 100, the energy consumption of proposed mechanism is 32.58%, 15.49%, 35.94% and 25.62% lower than technique1, 2, 3, 4.

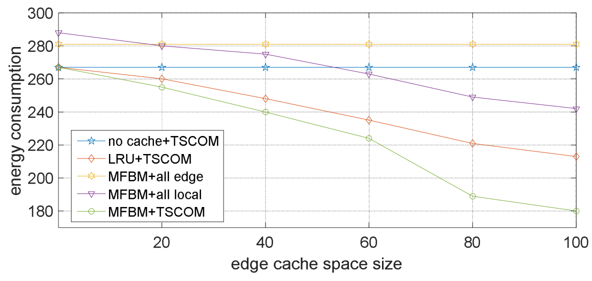


Fig. 5. Energy consumption as the edge cache space size increases

It can be seen from the above simulation results that adding sharable cache on the edge server can significantly reduce system energy consumption (compared with technique 1). The multi-factor-based edge server sharable cache management and update mechanism (MFBM) proposed in this paper is effective (compared to technique 2). What's more, the two-scenario computation task offloading mechanism (TSCOM) proposed in this paper also shows better results (compared to technique 3 and 4).

VI. Conclusion

In this paper, we propose a computation offloading mechanism based on edge server shareable cache. We first propose a four-tier architecture. Subsequently, a resource constraint optimization model combining communication, computing, and caching is established. Then, we design a multi-factor-based edge server sharable cache management and update mechanism, and propose a two-scenario computation offloading mechanism. Finally, we prove the effectiveness of the proposed mechanism through simulation.

Acknowledgment

This work is Supported by National Key R&D Program of China(2019YFB2103202, 2019YFB2103200), Science and Technology Project from Headquarters of State Grid Corporation of China: "Key technology development and application demonstration of high-confidence intelligent sensing and interactive integrated service system(52110418002V)", and the general project of the National Natural Science Foundation of China (62072049).

REFERENCES

[1] X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," in IEEE Transactions on Vehicular Technology, vol. 68, no. 1, pp. 856-868, Jan. 2019..

[2] Y. Dong, S. Guo, J. Liu and Y. Yang, "Energy-Efficient Fair Cooperation Fog Computing in Mobile Edge Networks for Smart City," in IEEE Internet of Things Journal, vol. 6, no. 5, pp. 7543-7554, Oct. 2019.

[3] L. Ferdouse, A. Anpalagan and S. Erkucuk, "Joint Communication and Computing Resource Allocation in 5G Cloud Radio Access Networks," in IEEE Transactions on Vehicular Technology, vol. 68, no. 9, pp. 9122-9135, Sept. 2019.

[4] J. Liu and C. Shi, "Optimization of Network-Based Caching and Forwarding Using Mobile Edge Computing," in IEEE Access, vol. 7, pp. 181855-181866, 2019

[5] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao and X. Shen, "Cooperative Edge Caching in User-Centric Clustered Mobile Networks," in IEEE Transactions on Mobile Computing, vol. 17, no. 8, pp. 1791-1805, 1 Aug. 2018

[6] H. Wei, H. Luo, Y. Sun and M. S. Obaidat, "Cache-Aware Computation Offloading in IoT Systems," in IEEE Systems Journal, vol. 14, no. 1, pp. 61-72, March 2020.

[7] P. Liu, G. Xu, K. Yang, K. Wang and X. Meng, "Jointly Optimized Energy-Minimal Resource Allocation in Cache-Enhanced Mobile Edge Computing Systems," in IEEE Access, vol. 7, pp. 3336-3347, 2019.

[8] Y. Hao, M. Chen, L. Hu, M. S. Hossain and A. Ghoneim, "Energy Efficient Task Caching and Offloading for Mobile Edge Computing," in IEEE Access, vol. 6, pp. 11365-11373, 2018.

[9] F. Cicirelli et al., "Edge Computing and Social Internet of Things for Large-Scale Smart Environments Development," in IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2557-2571, Aug. 2018.

[10] Q. Li, J. Zhao and Y. Gong, "Computation offloading and resource allocation for mobile edge computing with multiple access points," in IET Communications, vol. 13, no. 17, pp. 2668-2677, 29 10 2019.

[11] H. Abdi and L. J. Williams, "Principal component analysis," Wiley Inter-discipl. Rev., Comput. Statist., vol. 2, no. 4, pp. 433–459, 2010.

[12] M. I. A. Zahed, I. Ahmad, D. Habibi and Q. V. Phung, "Green and Secure Computation Offloading for Cache-Enabled IoT Networks," in IEEE Access, vol. 8, pp. 63840-63855, 2020.

[13] B. Jiang, P. Nain and D. Towsley, "LRU Cache under Stationary Requests". in ACM SIGMETRICS Performance Evaluation Review, vol. 45, no. 2, pp. 24-26. 2017