CMU: Towards Cooperative Content Caching with User Device in Mobile Edge Networks

Zhenbei Guo[1,2], Fuliang Li[1], Yuchao Zhang[3]
Changsheng Zhang[1], Tian Pan[3], Weichao Li[2], and Yi Wang[2]

[1] Northeastern University, Shenyang, P.R. China
[2] Peng Cheng Laboratory, Shenzhen, P.R. China
[3] Beijing University of Posts and Telecommunications, Beijing, P.R. China

Abstract. Content caching in mobile edge networks has stirred up tremendous research attention. However, most existing studies focus on predicting content popularity in mobile edge servers (MESs). In addition, they overlook how the content is cached, especially how to cache the content with user devices. In this paper, we propose CMU, a three-layer (Cloud-MES-Users) content caching framework and investigate the performance of different caching strategies under this framework. A user device who has cached the content can offer the content sharing service to other user devices through device-to-device communication. In addition, we prove that optimizing the transmission performance of CMU is an NP-hard problem. We provide a solution to solve this problem and describe how to calculate the number of distributed caching nodes under different parameters, including time, energy and storage. Finally, we evaluate CMU through a numerical analysis. Experiment results show that content caching with user devices could reduce the requests to Cloud and MESs, and decrease the content delivery time as well.

Keywords: Mobile Edge Networks · cooperative caching · device-to-device communication · content delivery.

1 Introduction

To cope with massive data traffic brought by the ever-increasing mobile users, the traditional centralized network model exhibits the disadvantages of high delay, poor real-time and high energy consumption. To solve these problems, mobile edge network model is proposed, which can provide cloud computing and caching capabilities at the edge of cellular networks. According to the survey report by Cisco [1], the number of connected devices and connections will reach 28.5 billion by 2022, especially mobile devices, such as smartphones, which will reach 12.3 billion. Cisco also points out that traffic of video will account for 82% of the total IP traffic. Due to the massive content transmission traffic generated by user requests, content caching is regarded as a research hotspot of mobile edge network[2, 3]. Liu et al.[4] also indicates that mobile edge caching can efficiently reduce the backhaul capacity requirement by 35%.

When a user requests the content in traditional centralized network, the content will be provided by remote server or Cloud, wherein there is usually duplicated traffic during the content transmission. By caching the content from the Cloud to the edge of the network (e.g., gateway and base station), the duplicated traffic can be avoided when the user chooses the closest mobile edge server. At the same time, it has a better network quality than the traditional centralized network. Due to the limited storage space, user devices cannot cache all the contents. But with the development of the hardware, the computing and storage capabilities of mobile devices have been improved greatly. Even though the storage capabilities of a user device cannot be compared with that of Cloud and MES, we can build a huge local content caching network relying on the explosive growth of mobile devices, which could use D2D to provide content sharing services [15, 16]. More and more studies show that a local content caching network has a great potential to achieve content sharing.

In this paper, we apply a three-layer content caching framework in the mobile edge network and aim to investigate the performance of different caching ways under this framework. Caching the content from Cloud and MES to user device could reduce the requests to Cloud and MES, as well as save the valuable bandwidth resources. It can also reduce the content transmission time and energy consumption. Paakkonen et al.[20] studies the performance of local content caching network when using different caching ways. We refer their caching strategies and make these caching strategies applicable to both MESs and user devices. We assume that the content popularity is known and it conforms to the ZipF model [24]. The main contributions of this paper can be summarized as follows.(1) We combine MESs and user devices to cache the content to create a multi-layer mobile edge caching framework. To the best of our knowledge, content caching that is assisted by user devices in mobile edge network has not been well studied in previous work. (2) We prove that minimizing transmission costs between MESs and user devices is an NP-Hard problem. We provide a solution to solve this problem and describe how to calculate the number of distributed caching nodes in a cluster. (3) We evaluate the performance of different caching strategies under the proposed framework through a numerical analysis. Results show that caching contents with user devices is feasible and effective.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces the proposed framework. Problem description and theoretical derivation are presented in Section 4. Numerical analysis results are shown in Section 5. Finally, the paper is concluded in Section 6.

2 Related Work

In recent content caching studies of mobile edge network, they use different machine learning methods to predict the popular content or apply other technologies to maximize the storage and computing capabilities of MESs [5–14]. For example, Chen et al.[5] used a self-supervising deep neural network to train the data and predict the distribution of contents. Liu et al.[10] proposed a novel MES-

enabled blockchain framework, wherein mobile miners utilize the closest MES to compute or cache the content (storing the cryptographic hashs of blocks). In these studies, even there are many caching strategies, the predicted contents are simply copied in MESs and the main optimization objective is the MES. A survey of content caching for mobile edge network [3] indicated that mobile traffic could reduce one to two thirds by caching at the edge of the network. In addition, D2D as one of the key technologies of 5G could easily help user to utilize the locally stored resources. And the QoE of users could have a great improvement by local content caching [17, 19]. Video traffic accounts for the vast majority of IP traffic [1], and Wu et al.[18] proposed a user-centric video transmission mechanism based on D2D communications that allows mobile users to cache and share videos with each other. Paakkonen et al.[20] also investigated different D2D caching strategies.

Existing studies have combined the local devices with the cellular network for traffic offloading. Kai et al.[25] considered a D2D-assisted MES scenario that achieves the task offloading. Zhang et al.[26] focused on the requested contents cached in the nearby peer mobile devices. They aimed to optimize the D2D throughput while guaranteeing the quality of D2D channels. However, devices randomly cached the popular contents [26], and there was not a global content caching policy. In addition, devices are distributed within a small range so that the communication could be established at any time. But in practice, we should consider the scenario where the requesting device is far from the caching device.

3 System model

The mobile edge network structure used in this paper is shown in Figure 1. The top layer is the Cloud, followed by the mobile edge server layer and the local user device layer. In the local user device layer, the devices (nodes) are divided into two types, i.e., Normal node and Caching node. When a device requests the content, the sequence of the request is Local >> MES >> Cloud.

3.1 Caching strategies

Four caching strategies are adopted, and some contents are randomly distributed across the MESs at the beginning.

(1) Default Caching (DC): Some contents are randomly stored in the MESs and remain unchanged. If the MESs do not have the requested content, the request will be sent to the Cloud.

(2) Replication Caching (RC): User caching nodes are called the caching nodes, and MES is called the caching server. The MESs and user nodes store a full copy of each content when the transmission process is finished. User nodes are regarded as the content servers after they cache the contents. They could send the desired content to the requesting device by the D2D communication. But if the caching node leaves, all the caching contents are lost. It has to request the desired content from the farther caching nodes or the MESs.

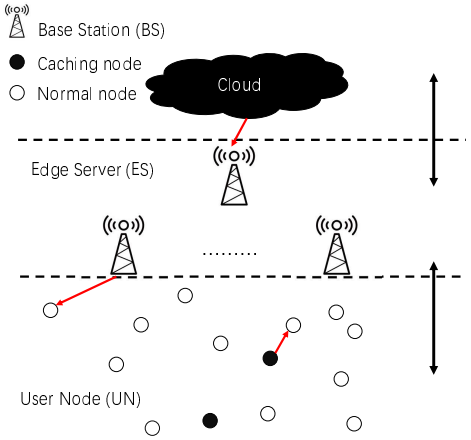


Fig. 1. Architecture of the mobile edge network.

(3) Distributed Replication Caching (DRC): User caching nodes are called the distributed caching nodes, and MES is called the distributed caching server. Assuming that the number of user nodes is N, and n ($n << N$) nodes are selected as the distributed caching nodes by MESs. Distributed caching nodes are responsible for content sharing to the normal nodes. Note that, distributed caching nodes do not actively request any content for themselves. If a distributed caching node requests the content for a normal node from the upper layer, all distributed caching nodes cache the full copy of each content, which is returned from the upper layer. That is to say each content is cached in n nodes. When a distributed caching node leaves its cluster, an available node from that cluster will replace it. Using this strategy, the cached contents are not easily lost. But there is an additional compensation consumption when a new distributed caching node recovers the previously cached content.

(4) Distributed Fragment Caching (DFC): Different from DRC, the content is divided into several parts according to the number of distributed caching nodes in the requesting cluster. That is to say each distributed caching node stores a part of the content. The upper layer performs the content fragmentation automatically during the experiment. Note that MESs use the DRC caching strategy when user nodes use DFC.

3.2 User State

We assume that state space of a user is $State = \{Leave, Stay, Request\}$, and the state-transition is a Markov chain with two reflective walls on a straight line as shown in Figure 2.

Where *Leave* means the user leaves and deletes the cached contents. The probability of *Leave* is p_l. We assume that a user returns back to the cluster conforming to the Poisson process. *Stay* means the user stay in the place with no

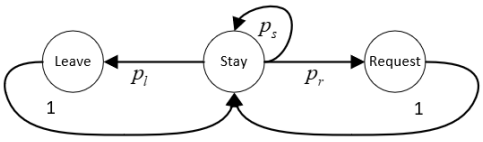


Fig. 2. The state-transition diagram.

actions, and the probability of *Stay* is p_s. *Request* means the user requests the desired content, and the probability of *Request* is p_r. We have $p_s + p_r + p_l = 1$. We assume that the number of all the users is $\mathcal{N}$, $\mathcal{N} = \{1, 2, ..., n\}$. The state matrix of the user is denoted by $State = \{State_{ij} \mid i \in N, j = 1, 2, 3\}$. We use 0 and 1 to represent the user state, where 1 represents that the user is in the corresponding state. The users can utilize the Service Set Identifier (SSID) to access the cached content and sense the state of other users. In addition, the users could receive the data from the MESs while they share the data to the other user devices. To avoid the transmission game, we define a game strategy set of $a_n \in \{0\} \cup \mathcal{N}$, wherein $a_n > 0$ means the user of n is willing to provide the content sharing service. The value of a_n is subtracted by 1 whenever the content is transmitted successfully. If $a_n = 0$, the user n is willing to request its desired content, and set a_n with the default value after successfully requesting.

4 Theoretical Analysis

In this section, we first introduce the NP-Hard problem and present the performance calculation formula for each caching strategy. When coming to the distributed caching, the formula derivation of calculating the number of the distributed caching nodes is presented.

4.1 Problem Formulation and Solution

The contents are denoted as $\mathcal{R} = \{r_1, ..., r_E\}$, and the size of each content is set to s. The popularity of each content is denoted as $\mathcal{W} = \{w_1, ..., w_E\}$. There are m MESs and the set of MESs is denoted as $\mathcal{S} = \{S_1, ..., S_m\}$. The set of $\alpha = \{\alpha_1, ..., \alpha_m\}$ means the storage space of the MESs, while $\beta = \{\beta_1, ..., \beta_n\}$ is the storage space of the users. And we have $\beta << \alpha$. The content caching matrix of the MESs is $X = \{X_{m,E} \mid S_m \in \mathcal{S}, r_E \in \mathcal{R}\}$, and the content matrix of the users is $Y = \{Y_{n,E} \mid n \in \mathcal{N}, r_E \in \mathcal{R}\}$. If a mobile edge server of m has cached the content of E, then $X_{m,E} = 1$, otherwise $X_{m,E} = 0$. The content matrix of the users has the same operations. The storage space of the MESs and the users should meet the following constraints.

$$\begin{cases} \sum\limits_{i=1}^{E} X_{m,i} \cdot s \leq \alpha_m, S_m \in \mathcal{S} \\ \sum\limits_{i=1}^{E} Y_{n,i} \cdot s \leq \beta_n, n \in \mathcal{N} \end{cases} \tag{1}$$

When the storage space reaches to the maximum capacity and a copy of new content needs to be cached, the Least Recently Used (LRU) algorithm is utilized to realize the space release and content replace.

In addition, we assume that the transmission consumption of the users is η_u, and the transmission rate is TR_u. The values for each MES are marked as η_s and TR_s. Similarly, the symbols of the Cloud are η_c and TR_c. We assume that C^E is the energy consumption and T^E is the time overhead caused by transmitting the content of E. We should make sure that the energy consumption and the time overhead are minimized during each transmission. e.g., node k transmits the content of E to node o, $C^E_{k,o}$ and $T^E_{k,o}$ should reach the minimum. That is to say the distance or the route path between them should be minimized. The distance matrix of the users is denoted as $DU = \{DU_{i,j} \mid i,j = 1,2,...,n; i \neq j\}$, while for each MES, the expression is $DS = \{DS_{i,j} \mid i,j = 1,2,...,m; i \neq j\}$. When the content of E is transmitted from node k to node o, the following constraints (Problem 1, P1) should be satisfied.

$$
\begin{aligned}
&(P1) \qquad Min \qquad (C^E + T^E) \\
&s.t. \qquad \sum_{i=1}^{E} X_{k,i} \cdot s \leq \alpha_k, S_k \in S \\
&\qquad\quad \sum_{i=1}^{E} Y_{k,i} \cdot s \leq \beta_k, k \in N \\
&DS_{k,o} = \underset{g \neq o}{Min} \{DS_{g,o} \mid g = 1,2,...,m; X_{g,E} = 1\} \\
&DU_{k,o} = \underset{g \neq o}{Min} \{DU_{g,o} \mid g = 1,2,...,n; Y_{g,E} = 1\}
\end{aligned}
$$

Where, after the transmission is completed, the required caching space should not exceed the maximum storage capacity of the node. During the transmission process, the shortest distance or route path should be chosen.

Theorem 1. *Energy consumption and time overhead reach the minimum in P1 is an NP-Hard problem.*

Proof. Travel Sale-man Problem (TSP) is a classical NP-Hard problem. During the experiment, we assume that any object (i.e., any user) of $o \in \{\mathcal{N}\}$ requests the content from the Cloud. The Cloud sends the content to the request object of o going through the MES layer and the user layer. The route path of the user layer is R_{user}, and the consumption of the single-hop routing is ν_u. Let R_{sever} denote the route path of the MES layer, and the consumption of the single-hop routing is ν_s. We have $\nu_{all} = \nu_u * R_{user} + \nu_s * R_{sever}$, and our goal is to minimize ν_{all}. We treat the request object of o as the Sale-man and consider the Cloud as the destination. The total number of the travel itinerary between them is $R_{user} + R_{sever}$, and there are $R_{user}^{R_{sever}}$ schemes. We need to find the smallest ν_{all} of these schemes. Therefore, P1 is equivalent to the Travel Sale-man Problem, i.e., P1 is NP-Hard.

Since P1 is NP-hard, we need to resort to a sub-optimal solution for P1. We use a clustering algorithm to reduce the number of the available route paths, e.g., the improved K-means++ algorithm. In RC, the requesting nodes give priority

to scanning other nodes within their own clusters. Frequent communication can enrich the cache community greatly. In the distributed caching strategies, each cluster has its own distributed caching nodes, wherein the normal nodes could request the contents from the closest distributed caching nodes.

The clustering algorithms can be used for both the MES layer and the user layer, and we take the user layer as an example to illustrate. The coordinate vectors of the users is regarded as the training sample, which is denoted as $I = \{I_i \mid I_i \in R^2, i = 1, 2, 3, ..., n\}$, e.g., $I_i = \{x_i, y_i\}$. There are K clusters, e.g., $\Phi = \{\Phi_1, ..., \Phi_K\}$. The center of the cluster is denoted as $\pi = \{\pi_i, ..., \pi_K\}$. The number of the cluster members is denoted as $\varphi = \{\varphi_1, ..., \varphi_K\}$. We calculate the euclidean distance of each sample from the center. For $\forall I_i \in \Phi_i$, we have:

$$\|I_i - \pi_i\|_2 = \min_j \|I_i - \pi_j\|_2. \tag{2}$$

The indicator function g is used to check whether the distance between the center and each sample is the shortest, which is expressed as

$$g_{i,j} = \begin{cases} 1, & \|I_i - \pi_i\|_2 = \min_j \|I_i - \pi_j\|_2 \\ 0, & Otherwise. \end{cases} \tag{3}$$

When $\sum_{i=1}^{n} \sum_{j=1}^{K} g_{i,j}$ is at its maximum, the error rate of cluster is the lowest. The new centres will be calculated by the following expression.

$$\pi_j = \frac{\sum_{i=1}^{n} g_{i,j} I_i}{\sum_{i=1}^{n} g_{i,j}}. \tag{4}$$

We use the improved K-means++ algorithm as shown in Algorithm 1.

If the center nodes are randomly selected and they are too close to each other, the convergence will become slow, which degrades the performance of K-means. Therefore, how to select the initial center nodes is optimized in Algorithm 1 (line 1 - line 11). We select the farthest node from the existing centres every time, so the selected center could cover all the samples as far as possible. When the center selection process is completed, the iteration process is enabled (line 12-line 17).

4.2 Performance Formulation

To better describe the performance and check whether the content is transmitted at the same layer, $\mathcal{B} = \{b_1, ..., b_n, b_{n+1}, ..., b_{n+m}\}$. $b_u^E = 1$ is utilized to represent that the content of E is transmitted by the user node of u.

Default Caching and **Replication Caching** have the same performance expressions from the aspects of storage, energy and time. The utilized storage

Algorithm 1 K-means++

Input: I, K
Output: Φ

Initialize the number of cluster center $k = 0$, the number of iteration $n = 0$, distance array $D = \{0\}$, cluster center array $\pi = \{0\}$;
Let $k = 1$, a node $\forall I_k \in I$ is randomly selected as the first center, update π;
while $k < K$ **do**
  $D = \{0\}$;
  for each non-central node $\forall I_i \in I$ **do**
    Calculate the shortest distance by equation (2);
    Update D;
  end for
  Select the node with the maximum value in D as the new center;
  Update k and π;
end while
Let the number of iteration $n = 1$.
repeat
  Partition each non-central node from I by equation (3).
  Calculate the new centres by equation (4) and update π ;
until $\pi_j(n+1) == \pi_j(n), \forall j = 1, 2, ..., K$

space is represented as follows.

$$(\sum_{i=1}^{m}\sum_{j=1}^{E} X_{i,j} + \sum_{i=1}^{n}\sum_{j=1}^{E} Y_{i,j}) \cdot s. \tag{5}$$

C^E is expressed as:

$$C^E = \begin{cases} \eta_u \cdot s & b_u^E = 1 \\ \eta_s \cdot s & b_u^E = 0, b_s^E = 1 \\ (\eta_s + \eta_c) \cdot s & b_u^E = 0, b_s^E = 0 \end{cases} \tag{6}$$

T^E is expressed as:

$$T^E = \begin{cases} \frac{s}{TR_u} & b_u^E = 1 \\ \frac{s}{TR_s} & b_u^E = 0, b_s^E = 1 \\ (\frac{s}{TR_s} + \frac{s}{TR_c}) & b_u^E = 0, b_s^E = 0 \end{cases} \tag{7}$$

Distributed Caching When using the caching strategies of DRC and DFC, the nodes first request a distributed caching node. If the required content is not found, the distributed caching node will request the closest distributed caching server. Then, the required content is cached by the distributed caching node and transmitted to the requesting nodes by D2D communication. If the distributed caching server does not have the requested content, it will request the Cloud with the same process. There are K clusters in the user layer divided by K-means++. However, each cluster cannot specify only one node as the distributed caching

node. If there is only one distributed caching node, it will be difficult to handle a large number of concurrent requests, because transmitting the content to the requesting nodes from one distributed caching node by D2D communication is infeasible.

Taking the user cluster as an example, ξ denotes the number of distributed caching nodes. It is assumed that ξ_K nodes are selected from cluster K as the distributed caching nodes to cache and share the contents. For the distributed caching strategies of DRC and DFC, the multi-objective optimization problem (Problem 2, P2) is expressed as follows.

$$\begin{aligned} P2: \quad & min \quad f_1 = \sum_{i=1}^{\xi_K} \sum_{j=1}^{E} Y_{i,j} \cdot s \\ & min \quad f_2 = \left\lceil \frac{\varphi_K - \xi_K}{\xi_K} \right\rceil \cdot T \\ & min \quad f_3 = \sum_{i=1}^{\varphi_K - \xi_K} \cdot C_i \\ s.t. \quad & 0 < \xi_K < \varphi_K \\ & \sum_{i=1}^{E} Y_{k,i} \cdot s \leq \beta_k, \quad k \in \xi_K. \end{aligned}$$

Where, f_1 denotes the storage space, and f_1 of each node can not exceed the limit value. Similarly, f_2 denotes the time required to complete all of the requests, and f_3 denotes the energy consumption caused by all of the successful requests. The optimization object of f_1 is opposite to f_2 and f_3. When f_1 decreases, fewer distributed caching nodes provide the content sharing services, but at the same time, f_2 and f_3 increase. When minimizing f_2 and f_3, f_1 increases.

In order to solve P2, the Linear Weighting Method is adopted to make P2 become a single-objective optimization problem. The weight vectors are denoted as $u = \{u_i \mid u_i \geq 0, i = 1, 2, 3\}$, and $\sum_{i=1}^{3} u_i = 1$. Each weight coefficient is corresponding to f_i in P2. Then P2 can be expressed as follows.

$$\min_{\xi_K} \sum_{i=1}^{3} u_i f_i(\xi_K). \tag{8}$$

Let $u_i = \xi_i$. And ξ_i is satisfied by expression (9).

$$f_i(\xi_i) = \min_{\xi_k \in \varphi_K} f_i(\xi_k) \qquad i = 1, 2, 3. \tag{9}$$

Where, ξ_i is the optimization value making the object of f_i reach the minimum. Then we have the following expression.

$$u = [1, \varphi_K - 1, \varphi_K - 1]^T. \tag{10}$$

We substitute equation (10) into equation (8).

$$\begin{aligned} min \quad F(\xi_K) = & \frac{\bar{\beta}}{2\varphi_K - 1} \cdot \xi_K + \frac{(\varphi_K - 1) \cdot \bar{T}}{2\varphi_K - 1} \cdot \frac{\varphi_K - \xi_K}{\xi_K} \\ & + \frac{(\varphi_K - 1) \cdot \bar{C}}{2\varphi_K - 1} \cdot (\varphi_K - \xi_K). \end{aligned} \tag{11}$$

$\bar{\beta}$ is the average caching space, $\bar{T}$ is the average transmission time and $\bar{C}$ is the average energy consumption. We take the derivative of ξ_K and set equation (11) to 0. Then we have the following expression.

$$\frac{\bar{\beta}}{2\varphi_K - 1} - \frac{(\varphi_K - 1)\cdot \bar{T}}{2\varphi_K - 1} \cdot \frac{\varphi_K}{\xi_K^2} - \frac{\varphi_K - 1}{2\varphi_K - 1} \cdot \bar{C} = 0$$

$$\bar{\beta} - \frac{(\varphi_K^2 - \varphi_K)\cdot \bar{T}}{\xi_K^2} - (\varphi_K - 1)\cdot \bar{C} = 0$$

$$\frac{(\varphi_K^2 - \varphi_K)\cdot \bar{T}}{\xi_K^2} = \bar{\beta} - (\varphi_K - 1)\cdot \bar{C}$$

$$\xi_K = \sqrt{\left|\frac{(\varphi_K^2 - \varphi_K)\cdot \bar{T}}{\bar{\beta} - (\varphi_K - 1)\cdot \bar{C}}\right|}.$$

The result is an efficient solution to equation (11). It is round up to an integer and considered as the number of the distributed caching nodes. We then discuss the performance of DRC and DFC.

1) **Distributed Replication Caching (DRC)** : MES layer has KS clusters, e.g., $\psi = \{\psi_1, ..., \psi_{KS}\}$. The number of the distributed caching servers in each cluster is denoted as $\bar{S}$. The utilized caching space can be estimated by the following expression.

$$\begin{gathered} \sum_{i=1}^{KS}\sum_{j=1}^{E} X_{m,j}\cdot s\cdot \bar{S}_i + \sum_{k=1}^{K}\sum_{j=1}^{E} Y_{n,j}\cdot s\cdot \xi_k \\ S_m \in \psi_i, \quad n \in \Phi_k. \end{gathered} \tag{12}$$

The distributed caching servers within a cluster cache the same contents so that only the copies of the content cached in one distributed caching server need to be recorded. Then, multiplying the copies by the content size and the number of the distributed caching servers, the result is the utilized caching space. In addition, in order to guarantee the consistency of the distributed caching servers, Algorithm 2 is applied to add and delete the content. We take the MES layer as an example to illustrate the algorithm. The energy consumption can be calculated by equation (6). But if the content of E is transmitted from the upper layer, it needs to be cached in the distributed caching server, as well as the distributed caching nodes. So each layer will have additional transmission consumption: multiplying the number of distributed caching servers or nodes by C^E. When a distributed caching node leaves, an available node from the same cluster will be selected as the new distributed caching node. The selection rule follows Algorithm 1 (line 1 - line 11). The extra consumption caused by recovering the cached content is also calculated by equation (6). We denote $\bar{N}_r$ as the number of nodes requesting content of E simultaneously. If the requesting nodes are more than the distributed caching nodes, some requesting nodes have to wait until

Algorithm 2 Adding and deleting the content

Input: r_E, S, α, X
Output: X

```
Initialize the content r = null;
for i=0 to the number of the distributed caching servers do
    while the caching space of S_i >= α_i do
        Find r by LRU;
        Delete r and update X.
    end while
end for
for i=0 to the number of the distributed caching servers do
    if X_{i,E} == 0 then
        Cache r_E and update X_{i,E} = 1.
    end if
end for
```

the distributed caching nodes have the capacity to provide the content sharing services. The required time is calculated by the following expression.

$$T^E = \begin{cases} \frac{s}{TR_u}\left\lceil\frac{\bar{N}_r}{\xi}\right\rceil & b_u^E = 1 \\ \frac{s}{TR_u}\left\lceil\frac{\bar{N}_r}{\xi}\right\rceil + \frac{s}{TR_s} & b_u^E = 0, b_s^E = 1 \\ \frac{s}{TR_u}\left\lceil\frac{\bar{N}_r}{\xi}\right\rceil + \frac{s}{TR_s} + \frac{s}{TR_c} & b_u^E = 0, b_s^E = 0. \end{cases} \tag{13}$$

2) **Distributed Fragment Caching (DFC)**: The content is divided into several parts. The size of each part of the content is different across the clusters. The caching space is expressed as follows.

$$\begin{gathered} \sum_{i=1}^{KS}\sum_{j=1}^{E} X_{m,j}\cdot s + \sum_{k=1}^{K}\sum_{j=1}^{E} Y_{n,j}\cdot s \\ S_m \in \psi_i, \quad n \in \Phi_k. \end{gathered} \tag{14}$$

If the target distributed caching node is busy, other parts of the required content will be transmitted from other distributed caching nodes. The required time is expressed as follows.

$$T^E = \begin{cases} \frac{s}{TR_u}\left\lceil\frac{\bar{N}_r}{\xi}\right\rceil & b_u^E = 1 \\ \frac{s}{TR_u}\left\lceil\frac{\bar{N}_r}{\xi}\right\rceil + \frac{s}{TR_s\cdot\xi} & b_u^E = 0, b_s^E = 1 \\ \frac{s}{TR_u}\left\lceil\frac{\bar{N}_r}{\xi}\right\rceil + \frac{s}{TR_s\cdot\xi} + \frac{s}{TR_c} & b_u^E = 0, b_s^E = 0. \end{cases} \tag{15}$$

The energy consumption can be calculated by equation (6). There is no extra transmission consumption when the content is transmitted from the upper layer, because the total size that needs to be transmitted is equal to the size of the original content. When a node is selected as the new distributed caching node, the extra consumption caused by recovering the cached content is calculated by

$\sum_{i=1}^{E} C^i$, wherein

$$C^i = \begin{cases} \eta_u \cdot \frac{s}{\xi} & if \quad Y_i, = 1 \\ 0 & Otherwise. \end{cases} \tag{16}$$

If all the distributed caching nodes are busy, the MESs will help to recover the cached content. Then, we have $C^i = \eta_s \cdot \frac{s}{\xi}$.

5 Experiment

5.1 Experiment Setup

10 MESs and 100 user nodes are randomly distributed within 500x500 meters. The simulation ends after the transmission is performed for 5000 times successfully. The caching space of each MES is 100 MB, while the size of each user node is 20 MB. We assume that the expectation of the Poisson process for the leaving nodes is 100. All the MESs are distributed caching servers in their own clusters, and they will not leave their places. The transmission parameters are set according to a survey report [27], where η_u is 5 j/MB. The MESs and the Cloud utilize the cellular network, where η_s and η_c are 100 j/MB. The transmission rate of the user is between 54 Mb/s and 11 Mb/s within 200 meters, while for the MESs, the value is between 1 Mb/s and 0.5 Mb/s within 500 meters. The value for the Cloud is set to 0.5 Mb/s. There are 1000 video items and the size of each item is 20 Mb. The popularity distribution of the items is generally modeled as a ZipF distribution. The default shape parameter θ is set to 0.56 [28]. The default game strategy value of a is set to 1. The default requesting probability of p_r is 0.6, and p_s is equal to p_l, i.e., 0.2.

In addition, we add two comparison objects, i.e., DRC-Extra and DFC-Extra. All user nodes in RC could cache the contents, while only the distributed caching nodes have the caching ability in DRC and DFC. Therefore, to guarantee the same caching capacity, we increase the caching space of the distributed caching nodes in DRC-Extra and DFC-Extra. The new caching space of each distributed caching node is calculated as follows.

$$new\ caching\ space = \frac{the\ total\ caching\ space\ of\ RC}{the\ number\ of\ the\ distributed\ caching\ nodes}. \tag{17}$$

We calculate the **average transmission time (ATT)**, **average energy consumption (AEC)** and **the number of local requests (LR)** that are served by the caching nodes or the distributed caching nodes. These metrics are used to evaluate the effect on reducing the requests to the MESs and the Cloud. The **caching space (CS)** is adopted for choosing an appropriate caching strategy when the caching space is restricted.

5.2 Numerical Analysis

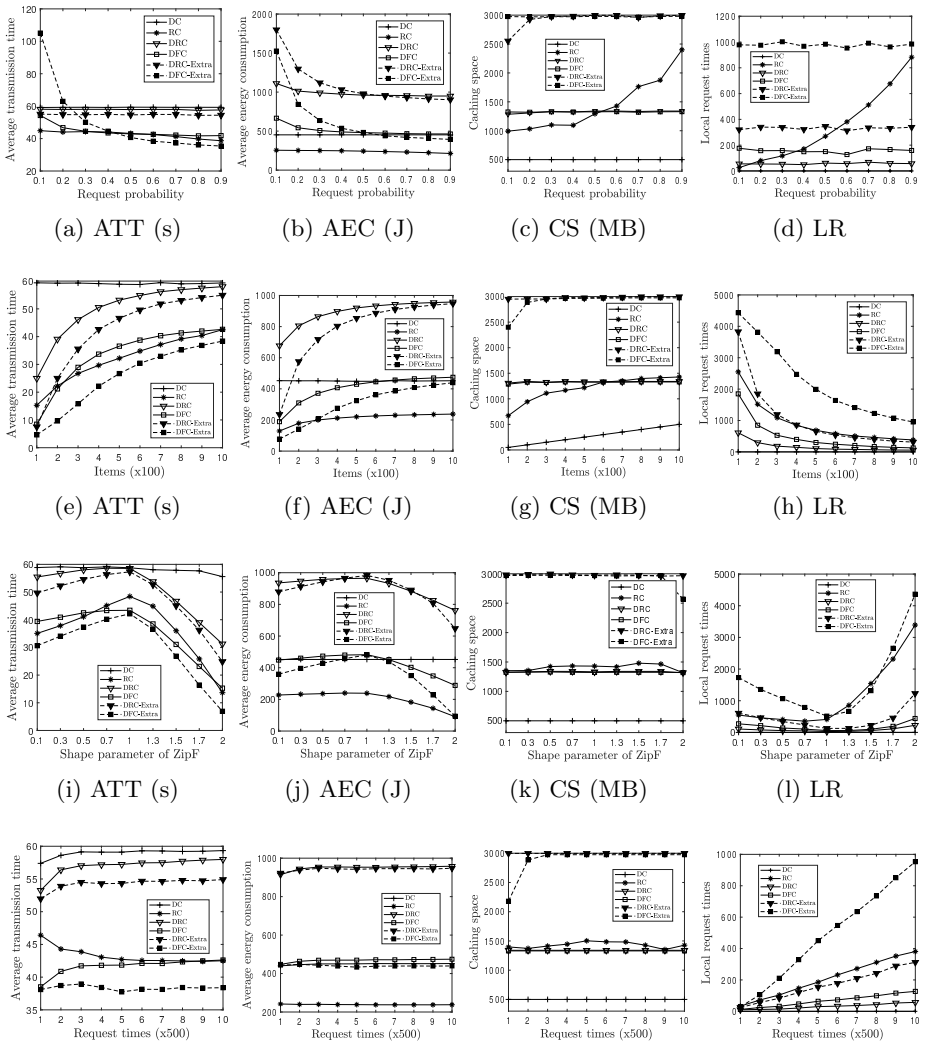
The DC strategy is utilized as a baseline for numerical analysis. In DC, contents are randomly stored across the MESs and the cached contents remain unchanged

during the whole experiment. The numerical analysis results under different parameters are shown in Fig. 3a to Fig. 3o.

In Fig. 3a to Fig. 3d, *items* and θ are set to 1000 and 0.56 respectively, while p_r changes from 0.1 to 0.9. The increase of the request probability means fewer nodes leave away and more different contents are cached. As a result, the caching space increases (Seen in Fig.3c), while the energy consumption decreases (Seen in Fig.3b). For DFC, DFC-Extra and RC, the copies of local contents increase with the growth of p_r. So the transmission time of them decreases. For DRC and DRC-Extra, since they have to cache the full copy of each content and the caching space is limited, the amount of the content cached locally is small. They have to request new contents from the upper layer frequently, so the transmission time of DRC and DRC-extra remains unchanged. Since the amount of content cached on MESs increases, DRC and DRC-Extra spend less time than DC. Note that the time consumption of DFC-Extra is high when p_r is equal to 0.1. The low request rate causes a large number of distributed caching nodes to leave, and the requesting nodes have to wait for new distributed caching nodes to recover the cached content. Compared with RC, cached contents are not lost when using distributed caching strategies so that the caching space remains unchanged. Meanwhile, the extra consumption of recovering the cached contents causes higher energy consumption when using distributed caching strategies.

In Fig. 3e to Fig. 3h, θ and p_r are set to 0.56 and 0.6 respectively, while *items* change from 100 to 1000. When the number of items is 100 (most of the contents are cached locally), the performance of each strategy is optimal, especially DRC and DRC-Extra. It means that the distributed caching strategies perform better with a larger caching space. The increase of items means more different contents can be requested and the content caching update is more frequent, which causes the increase of the transmission time (Seen in Fig.3e), the energy consumption (Seen in Fig.3f) and the caching space (Seen in Fig.3g). Meanwhile, it reduces the number of local requests (Seen in Fig.3h). The caching space of distributed caching strategies remains unchanged because the number of the distributed servers is fixed and the cached contents are not lost. The caching space of RC fluctuates greatly caused by the leaving caching nodes. DRC and DRC-Extra have to cache several full backups after each successful request, resulting in the highest energy consumption (Seen in Fig.3f).

In Fig. 3i to Fig. 3l, $p_r = 0.6$ and *items* are set to 0.6 and 1000 respectively, while θ changes from 0.1 to 2. The parameter of θ determines the popularity of the items. When θ is too small or too large, some contents will be requested frequently. The performance of each strategy is optimal when θ is equal to 2, where the popularity of the items is high. When θ changes from 0.1 to 1, the transmission time increase s (Seen in Fig.3i), wherein the popularity of items are evenly distributed gradually. The cached contents update frequently, resulting in more energy consumption (Seen in Fig.3j) and fewer local requests (Seen in Fig.3l). When θ is greater than 1, some contents are requested frequently. The transmission time and energy consumption decrease, while the number of local



(a) ATT (s) (b) AEC (J) (c) CS (MB) (d) LR

(e) ATT (s) (f) AEC (J) (g) CS (MB) (h) LR

(i) ATT (s) (j) AEC (J) (k) CS (MB) (l) LR

(m) ATT (s) (n) AEC (J) (o) CS (MB) (p) LR

Fig. 3. Experimental results under different parameters. Fig. 3a to Fig. 3d, $items = 1000$, $\theta = 0.56$, $p_r \in [0.1, 0.9]$. Fig. 3e to Fig. 3h, $\theta = 0.56$, $p_r = 0.6$, $items \in [100, 1000]$. Fig. 3i to Fig. 3l, $p_r = 0.6$, $items = 1000$, $\theta \in [0.1, 2]$. Fig. 3m to Fig. 3p, $p_r = 0.6$, $\theta = 0.56$, $items = 1000$.

requests increases. The caching space (Seen in Fig.3k) remains high because other requested contents are still cached.

In Fig. 3m to Fig. 3p, p_r, $items$ and θ are set to 0.6, 1000 and 0.56 respectively. We evaluate the performance under different successful requests. As depicted in Fig.3m, DFC-Extra has the minimum transmission time, and the transmission time of RC and DFC is gradually approaching to each other. At the beginning, the amount of the cached content of RC is small. However, the cached content of RC is enriched gradually and the transmission time decreases, while the transmission time of the distributed caching strategies increases due to the small amount of the cached content and frequent content caching updates. The energy consumption increases slightly (Seen in Fig.3n) caused by updating and recovering the content. The caching space (Seen in Fig.3o) of the distributed caching strategies remains unchanged after it reaches the maximum limit, while the caching space of RC still fluctuates. The local requests (Seen in Fig.3p) of DFC-Extra and RC tend to grow faster due to caching a large number of popular contents.

The findings are summarized as follows. (1) The transmission time and energy consumption of RC, DFC and DFC-Extra are better than other caching strategies in most cases. DFC-Extra performs best, but it needs a large cache space and presents high energy consumption as well. (2) RC performs well in all the scenarios. However, the cached contents are easily lost and the performance degrades if the nodes in the community are not willing to share the contents. (3) All the distributed caching strategies work badly when the nodes leave frequently (e.g., $p_r = 0.1$). However, all of them work well under high request probability and popularity (e.g., $\theta = 2$). The distributed caching strategies need more caching space to work better, but they also cause higher energy consumption. DRC and DRC-Extra seem to present the worst performance, but they can keep working when only one distributed caching node provides service. In DFC and DFC-Extra, the nodes have to wait when one of the distributed caching nodes is lost. (4) More importantly, using user devices to cache the content could not only reduce the requests to the Cloud and the MESs, but also decrease the content transmission time as shown in Fig.3a and Fig.3d.

6 Conclusions

In this paper, we propose CMU, a three-layer (Cloud-MESs-Users) content caching framework, which could offload the traffic from Cloud-MESs to user devices. We describe the content caching problem and evaluate the performance of different caching strategies under this framework. Numerical results show that content caching with user devices could reduce the requests to Cloud-MESs, as well as decrease the content transmission time. We will evaluate CMU with a real testbed in the future. To attract more nodes to participate in content caching and sharing, an incentive mechanism is also needed for the distributed caching strategies.

Acknowledgment

This work is supported by the National Key Research and Development Program of China under Grant Nos. 2018YFB1702000; the project of PCL Future Regional Network Facilities for Large-scale Experiments and Applications under Grant No. PCL2018KP001.

References

1. Cisco, "Cisco visual networking index: Global mobile data traffic forecast update 2017-2022," 2019.
2. X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, February 2014.
3. S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.
4. D. Liu and C. Yang, "Energy efficiency of downlink networks with caching at base stations," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 907–922, April 2016.
5. J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu, "iraf: A deep reinforcement learning approach for collaborative mobile edge computing iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7011–7024, 2019.
6. W. Jiang, G. Feng, S. Qin, and Y. Liang, "Learning-based cooperative content caching policy for mobile edge computing," in *2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019*, 2019, pp. 1–6.
7. L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5520–5530, 2019.
8. T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," *Int. J. Communication Systems*, vol. 31, no. 11, 2018. [Online]. Available:
9. Q. Cui, Z. Gong, W. Ni, Y. Hou, X. Chen, X. Tao, and P. Zhang, "Stochastic online learning for mobile edge computing: Learning from changes," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 63–69, 2019.
10. M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Vehicular Technology*, vol. 67, no. 11, pp. 11 008–11 021, 2018.
11. S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mob. Comput.*, vol. 17, no. 8, pp. 1791–1805, 2018.
12. M. Sheraz, F. Wang, and Y. Li, "Enhancing mobile user performance through data caching over edge computing wireless networks," in *14th International Wireless Communications & Mobile Computing Conference, IWCMC 2018, Limassol, Cyprus, June 25-29, 2018*, 2018, pp. 1148–1153.
13. C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Joint computation offloading, resource allocation and content caching in cellular networks with mobile edge computing," in *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*, 2017, pp. 1–6.

14. J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, 2018, pp. 207–215.
15. D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-device communications with wi-fi direct: overview and experimentation," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 96–104, June 2013.
16. Z. Wang, F. Li, X. Wang, T. Li, and T. Hong, "A wifi-direct based local communication system," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, June 2018, pp. 1–6.
17. B. Bai, L. Wang, Z. Han, W. Chen, and T. Svensson, "Caching based socially-aware d2d communications in wireless content delivery networks: a hypergraph framework," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 74–81, August 2016.
18. D. Wu, Q. Liu, H. Wang, Q. Yang, and R. Wang, "Cache less for more: Exploiting cooperative video caching and delivery in d2d communications," *IEEE Transactions on Multimedia*, vol. 21, no. 7, pp. 1788–1798, July 2019.
19. W. Zhang, D. Wu, W. Yang, and Y. Cai, "Caching on the move: A user interest-driven caching strategy for d2d content sharing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2958–2971, March 2019.
20. J. Paakkonen, A. Barreal, C. Hollanti, and O. Tirkkonen, "Coded caching clusters with device-to-device communications," *IEEE Transactions on Mobile Computing*, vol. 18, no. 02, pp. 264–275, feb 2019.
21. A. Asheralieva and D. Niyato, "Game theory and lyapunov optimization for cloud-based content delivery networks with device-to-device and uav-enabled caching," *IEEE Transactions on Vehicular Technology*, vol. PP, pp. 1–1, 08 2019.
22. J. Li, M. Liu, J. Lu, F. Shu, Y. Zhang, S. Bayat, and D. N. K. Jayakody, "On social-aware content caching for d2d-enabled cellular networks with matching theory," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 297–310, Feb 2019.
23. J. Wang, M. Cheng, Q. Yan, and X. Tang, "On the placement delivery array design for coded caching scheme in d2d networks," *IEEE Transactions on Communications*, vol. PP, 12 2017.
24. L. Jiang, G. Feng, and S. Qin, "Cooperative content distribution for 5g systems based on distributed cloud service network," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, June 2015, pp. 1125–1130.
25. Y. Kai, J. Wang, and H. Zhu, "Energy minimization for d2d-assisted mobile edge computing networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.
26. X. Zhang and Q. Zhu, "Distributed mobile devices caching over edge computing wireless networks," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 127–132.
27. O. Hayat, R. Ngah, S. Hashim, M. Dahri, R. Malik, and Y. Rahayu, "Device discovery in d2d communication: A survey," *IEEE Access*, vol. PP, pp. 1–1, 09 2019.
28. P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 1897–1903.