2018 IEEE 16th Int. Conf. on Dependable, Autonomic & Secure Comp., 16th Int. Conf. on Pervasive Intelligence & Comp., 4th Int. Conf. on Big Data Intelligence & Comp., and 3rd Cyber Sci. & Tech. Cong.

Cooperative Caching Plan of Popular Videos for Mobile Users by Grouping Preferences

Yi-Ting Chen Chia-Cheng Yen
CS Department
National Tsing Hua University
Hsinchu, Taiwan
melissana21@gmail.com setsuna01gn@gmail.com

Yu-Tai Lin Jia-Shung Wang
CS Department
National Tsing Hua University
Hsinchu, Taiwan
ytlin1993@gmail.com jswang@cs.nthu.edu.tw

***Abstract*—Mobile traffic has grown fast in recent years, particularly for delivering popular video clips at anywhere. Based on a combination of Macro Cells and several Small Cells (SC) technologies, HetNets is gaining increasing attention due to the surge demand on high-quality mobile video services. To avoid bottleneck in the limited capacity of backhaul link, Mobile Edge Computing (MEC) is a promising solution, which computing and caching in the mobile edge in such a way that the buffered video can be delivered with less network latency and traffic load. Our goal is to build a cooperative caching plan for serving popular video clips over HetNets under the Joint Transmission (JT) method in MEC environment with least possible backhaul traffic. In the training phase, categorize similar users to clusters and to SCs using the well-known spectral clustering algorithm. Then aggregate the users' requests to be the request profile of the corresponding SCs. Third, share the caching space among cooperated SCs with the help of distributed LT codes. During the serving phase, new coming users will be assigned to appropriate SCs based on similarity between users and SCs. Our simulation results show that the backhaul traffic rate can decrease from 38% to 10% (or 62% to 19%) if cache space is acceptable.**

Index Terms*—*Caching; Clustering; LT codes; MEC; Small Cells; Joint Transmission

I. Introduction

As the CISCO's VNI Mobile Forecast report [1] mentioned that mobile data traffic has grown 4,000-fold over the past 10 years. Mobile video will grow at a CAGR of 62 percent between 2015 and 2020, and three-fourths of the world's mobile data traffic will be video by 2020.

To cope with large amount of mobile video requests under limited backhaul capacity, HetNets is gaining increasing attention due to the surge demand on high-quality mobile video services. To avoid bottleneck in the limited capacity of backhaul link to the core network, with the emergence of Mobile Edge Computing (MEC) [2] technology, which computing and caching in the mobile edge (such as Small Cells, SCs) in such a way that the buffered video can be delivered with less network latency and traffic load to lessen network stress. Caching popular video clips locally can help to reduce network traffic load. However, due to the capacity of cache is limited, traditional replacement policy may lead to unacceptable amount of cache miss. There are two different ways to design the caching plan: assigning requests to SCs or assigning users to SCs. For the method of assigning requests to SCs, large amounts of requests of a certain video cause the problem of load balancing. In the paper, we propose a cooperative caching solution of assigning users to SCs, based on the promising Joint Transmission (JT) method of CoMP scheme considered for LTE-A systems [3].

In training phase, users are grouping and assigned to SCs. To cope with limited cache size, SCs cooperate to share their caching space by LT codes. SCs are equipped with computational apparatus and cache space, meaning they have ability to communicate with each other. The mobile users can download the video data from multiple cooperative nodes to achieve bandwidth aggregation under JT.

Our goal is to maximize the *Local Delivery Rate (LDR)*:

$$LDR = \frac{\#\ of\ requests\ served\ in\ cache}{\#\ of\ total\ requests}$$

Or minimize the *Global Download Rate (GDR)*:

$$GDR = \frac{\#\ of\ requests\ not\ served\ in\ cache}{\#\ of\ total\ requests}$$

The contributions of this paper are summarized as follows:

(i) A cooperative caching plan of MEC is proposed for serving popular video clips over HetNets with least possible backhaul traffic.
(ii) A valuable clustering scheme of users with recapping users' preferences.
(iii) SCs download the encoded video by LT codes and decode cooperatively to share their caching space. It also has benefit of load balancing.
(iv) The (LT) coded packages can be randomly distributed among SCs. Also, can be randomly download instead of selecting the specific data.
(v) In service phase, users can achieve the goal of bandwidth aggregation.

We briefly describe the related works and the concept of LT codes in Section 2. In Section 3, the clustering method of similar users, the cooperative caching plan of MEC, and the parallel downloading scheme are described and discussed. The experimental and simulation results are presented in Section 4. Some conclusions drawn are presented in Section 5.

II. Related Works

A. LT Codes with Belief Propagation

Luby Transform codes (LT codes) [4] is one of the Fountain codes [5], proposed by Michael Luby in 2002. The coding complexity is quite low because LT codes employ the particularly simple operation XOR. The concept of LT codes is users, which can recover *K* original source symbol with high probability when thy receive enough (little larger than *K*) encoded packets which generated from a given set of source symbols by the servers. Fig. 1 (a) shows the encoding process of LT codes. For example, the packet of degree 3 is encoded with source symbols ($2 \oplus 3 \oplus 5$).

Belief Propagation (BP) is a well-known and efficient algorithm for decoding LT encoded data. As the example of Fig. 1 (a), the BP algorithm chooses the degree 1 packets as the ripple set, so the first ripple set is {2}. The encoded packets connected with symbol 2 are removed. Then symbol 5 can be decoded, so the new ripple set is {5}, and so on.

B. Video Services in Wireless Network

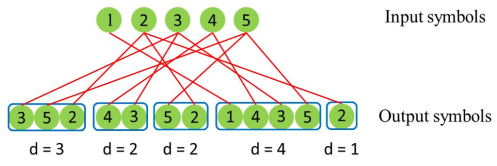
According to the widespread of mobile devices, mobile data traffic has grown rapidly. To cope with the problem, many researchers have proposed methods or designs to deliver videos efficiently.

1) Cooperative on mobile devices

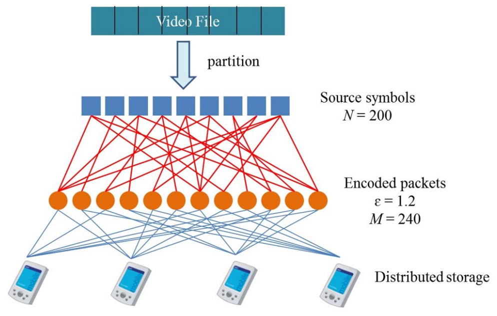
In [6], Keller et al. consider that a group of users, within proximity of each other, request for the same video in a short time. For example, a group of friends want to watch the same show together, but sharing on one phone screen is not comfortable. Hence, they all want to download and watch in their own devices. A system was proposed to cooperative streaming by smartphones, scheduling which parts of the video each phone should download from which server. Notice that, in in our solution, the servers do not have to worry about partitioning and scheduling tasks because, with LT codes, the coded data can be randomly distributed and randomly downloaded. By device-to-device connection, each user can watch the complete video by downloading some segments only.

2) Content Delivery by Cooperative Caching

The idea of using wireless distributed caching (assignment of files to cache services) to help in networking was proposed and examined in [7] [8] [9][10]. The contribution of [7] and [8] is to formalize the distributed caching problem, prove that it is NP-hard, and give an approximation algorithm to minimize the expected download time. Considering that SC users have different preferences over different video types, a caching strategy was proposed in [9]. By a reinforcement learning algorithm, each SC learns the popularity distribution of each group to decide how to cache. In [10], a version of parallel LT downloading and decoding was proposed to save the cache space. In [11], address issue on user mobility and user QoS prediction to improve on the accuracy of service recommendation in mobile edge computing.



(a) A simple example of LT encoding.



(b) LT encoding process with source symbols $N = 200$, $\varepsilon = 1.2$.

Figure 1. LT encoding and delivery

C. Community Detection in Social Networks and Web Videos Categorization

The linking graph of objects (users or videos) is tremendous large in YouTube-like services thus categorization is needed for analyzing its grouping structures. Some methods are proposed [12] [13] [14] to solve the community detection problem. In [14], a cluster affiliation model for big networks was proposed. It also presented a new way of calculating to discover the overlapping community. In [15], a so-called spectral clustering technique was presented and analyzed, it makes use of the eigenvalues (spectrum) of the similarity matrix of the data to perform dimensionality reduction.

III. Proposed Methods

In this section, we proposed two methods to decrease the global download rate for different applications. One is clustering mobile users with similar preference. Section III.A proposes such a cooperative caching plan. In the training phase, categorize similar users to clusters and to SCs using the spectral clustering algorithm. Then aggregate the users' requests to be the request profile of the corresponding SCs. Third, share the caching space among cooperated SCs with the help of distributed LT codes. During the serving phase, new coming users will be assigned to appropriate SCs based on similarity between users and SCs. Another is streaming a video cooperatively with mobile devices by users. Section III.B presents such a cooperative mobile scheme. Here, the global download rate is decreased by multiple mobile users while downloading the same video and decoding cooperatively.

Our methods are designated for MEC, or precisely the HetNets with Joint Transmission. Assume there are some mobile users ($u_1, u_2, \ldots, u_m$) and corresponding request profiles (***uP^1***, ***uP^2***, …, ***uP^m***). Every request profile consists of a sequence of video clips issued by the user.

A. Cooperative Caching Plan for Small Cells

Given users' preferences, first, we aim to find an assignment (users to small cells) such that the bandwidth of backhaul services is minimized under the constraints that the caching space of each SC is limited. The flow chart is shown in Fig. 2. In training phase, users with similar preference are grouped together as a cluster through the spectral clustering algorithm [15]. Note that some inappropriate clusters are further recombined so as to balance the requests in the post-process stage. Then, clusters are assigned to SCs. Finally, based on the design of LT codes, we can let some SCs (cooperative SCs in the figure) with highly similarity to share their cache space eventually. In serving phase, new users are assigned to SCs according to the (profile) similarity between users and SCs. In this paper, we assume that the users' preferences will not change rapidly, therefore, each cluster may sustain its momentum a good while.

1) User Clustering with the spectral clustering [15]

Given users' profiles, to calculate the similarity between users, we exploit the BIGCLAM (Cluster Affiliation model for Big Networks) [14] to explore the similarity between users i and j, $p(i,j)$:

$$P(i,j) = 1 - exp(-uP^{i} \cdot uP^{j}) \qquad \text{Eq. (1)}$$

More precisely, if users i and j issued the request of the same video v closely, this probability will be adjusted with a weighting factor $w(i, j)$, where the average cache time will be discussed in Section IV.

$$w(i,j) = \sum_{v \in V} \frac{average\ cache\ time}{time(i,v) - time(j,v)}.$$

So the probability $p(i,j)$ will be modified as the following form:

$$P(i,j) = 1 - \exp\left(-w(i,j) \times (uP^{i} \cdot uP^{j})\right) \qquad \text{Eq. (2)}$$

Then, the spectral clustering algorithm [15] is utilized to categorize users. The pseudo codes is listed in Algorithm 1.

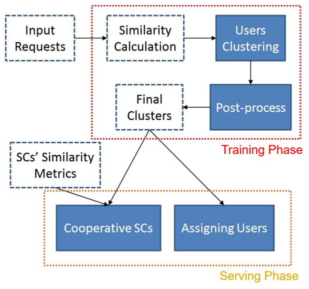


Figure 2. Flow chart of the cooperative caching plan.

Algorithm 1. Spectral Clustering

```
Input:
  Similarity matrix S = [P (i, j)] as in Eq. (2) and number of cluster $k$;
Output:
  $k$ clusters: {$C_1, C_2, \ldots, C_k$};
1. Let W = S
2. Calculate the diagonal degree matrix $D$ with $d_i = \sum_{j=1}^{m} P(i,j)$.
3. Calculate $L = D - W$
4. Normalize $L$ by $L = D^{-1/2} L D^{-1/2}$.
5. Find the k smaller eigenvalues of $L$ and corresponding eigenvectors.
6. Let the eigenvectors as columns and use k-means to cluster the row.
```

The number of cluster k is estimated by calculating modularity Q of partitioning subgraph b of graph G [16], see below:

$$Q \propto \sum_{b \in B} [\,(\#\ edges\ within\ group\ b) - (expected\ \#\ edges\ within\ group\ b)]$$

Higher value of Q means better partition and the value is ranging from 0.3 to 0.7. Here we set k to be 10 because of the highest value of Q in our demonstrative benchmarks.

2) Post Processing

In this stage, some larger clusters and some smaller ones will be well-adjusted. We observed that user groups are separated into two particularly characteristics: topic-specific users and (YouTube) addicted users. As shown in Table 1(a), the topic-specific users usually request few specific videos, and the (YouTube) addicted users watch almost every popular video, e.g. Cluster 3. Notice that this phenomenon, there is a large number of clusters of topic-specific, and a few number (may be one or two) of clusters of addicted-users, is common. And the latter always have the larger cluster sizes.

Assigning each cluster to a small cell straightforwardly will lead to some problems, such as the unbalance issue. Consequently, some SCs that serve oversized requests, the cached videos will be changed frequently, since the cache space is limited,

To balance the requests, we split the larger clusters by sorting out users randomly and merge the smaller ones. As the case in Table 1(a), Clusters 0, 1, 7 are merged to Cluster 0 in Table 1 (b), the Clusters 5, 8 are merged to Cluster 1. And Cluster 3 is split into Clusters 2, 3 and 4 in Table 1(b). After post processing, each cluster will be assigned to a SC finally.

3) Cooperative Caching (Serving Phase)

After training phase, in each cluster (or SCs), the users' requests will be aggregated to be the request profile of each cluster (or SCs).

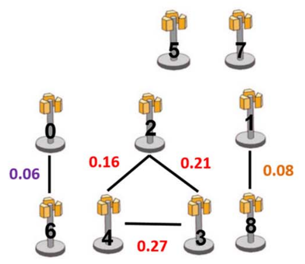
With the help of LT coding, our goal is to share the cache space among similar SCs. Then, several SCs can cooperate to download and (LT) decode together, that is, each SC just needs to download and cache *1/m* of video size if *m* SCs working corporately. The six group of corporative SCs are illustrated in Table 1(c). According to the similarity between clusters (SCs), there are two styles of grouping as shown in Table 1(c), the first one, which are separated from a large cluster, will connect to each other, e.g. Clusters 2, 3, and 4. Other clusters, which are similar enough (• threshold), will cooperate according to the similarity of SCs' profiles. As Table 1(c) shown, Cluster 0 cooperates with 6, and Cluster 1 cooperates with 8.

	Clus0	Clus1	Clus2	Clus3	Clus4	Clus5	Clus6	Clus7	Clus8	Clus9	Total
V_0	0	0	0	275	2	0	0	0	0	0	277
V_1	0	0	0	31	13	0	210	0	3	16	273
V_2	0	4	196	25	3	1	0	0	0	5	234
V_3	0	0	0	25	0	0	0	0	2	178	205
V_4	0	2	0	49	17	0	2	0	112	16	198
V_5	66	0	1	89	0	1	1	0	3	8	169
V_6	0	75	0	41	3	0	1	0	11	34	165
V_7	0	0	0	25	134	0	0	0	0	2	161
V_8	1	1	1	28	17	83	6	0	2	10	149
V_9	3	3	1	20	26	1	3	44	2	10	113
Total	70	85	199	608	215	86	223	44	135	279	1944
User	42	62	78	193	113	73	113	35	76	140	925

(a) Clustering result by spectral clustering.

	Clus0	Clus1	Clus2	Clus3	Clus4	Clus5	Clus6	Clus7	Clus8	Total
V_0	0	0	64	113	98	0	2	0	0	277
V_1	0	3	6	2	23	0	13	210	16	273
V_2	1	4	13	0	12	196	3	0	5	234
V_3	0	2	12	5	8	0	0	0	178	205
V_4	0	114	22	5	22	0	17	2	16	198
V_5	67	3	24	42	23	1	0	1	8	169
V_6	0	86	24	1	16	0	3	1	34	165
V_7	0	0	8	5	12	0	134	0	2	161
V_8	84	3	16	2	10	1	17	6	10	149
V_9	48	5	17	0	3	1	26	3	10	113
Total	200	220	206	175	227	199	215	223	279	1944
User	150	138	64	64	65	78	113	113	140	925

(b) Clustering result by post processing.



(c) The six groups of cooperative SCs corresponding to Table 1.

TABLE 1. CLUSTERING RESULTS: BEFORE AND AFTER POST PROCESSING.

4) *Assignment of Users to Clusters* (*Serving Phase*)

Each trained user was already assigned to a cluster in the training phase. Here, the new coming users will be assigned to the appropriate clusters on account of the similarity measure between users and clusters' profile using Eq. (1). Because of the cold start problem for the new users, we let the assignment change a few times till converge. As mentioned before, we have observed a categorization phenomenon, one or two re-assignments is enough for both the topic-specific and addicted-users clusters.

5) *Cache Replacement* (*Serving Phase*)

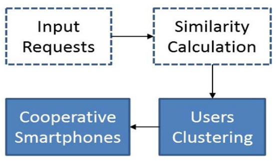
Due to limitation of cache space, requesting a video which is not cached may cause cache replacement. Consider the hit ratio of a cache, which defines how often a request (video) is actually kept. Accurately, this index is concerned with our global download rate. Several efficient replacement policies are proposed to improve the hit rate (for a given cache size). According to our experience and others, there is no significant performance difference between the Least Frequency Used (LFU) policy and the Least Recently Used (LRU) policy, so LFU is employed in the paper. Indeed, caching the most popular video clips in each group has the benefit of increasing times of serving directly instead of downloading to lead higher backhaul loading.

B. Cooperative Streaming with Mobile Devices

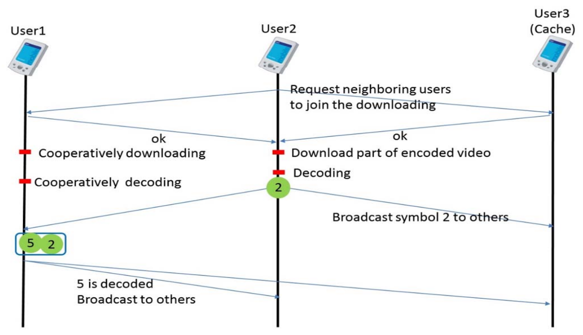
Here we briefly discuss another application of streaming video cooperatively with mobile devices to demonstrate the performance gain of parallel downloading of LT codes. The flow chart is shown in Fig. 3(a). The method of clustering is the same as discussed in Section III.A. After clustering, in each group, the probability of the users who watching the same videos simultaneously looks high. So, each user can easily discover the suitable partners to download the video cooperatively.

As illustrated in Fig. 3 (b), this example shows that User 2 can find and ask Users 1 and 3 to join the downloading corporately. If both agree, one of the users is assigned to be the role of *cache* (who does not have to download the video), say User 3. Other users download 1/2 of encoded video packets and decode together. As shown in Fig. 3(b), when a symbol is decoded by User 2, it then will be broadcasted to others.

The role of *cache* plays an important part to avoid decoding failure when some packet loss occurs. If the *cache* receives nothing within a period, say one second, the *cache* will send a message to the other nodes, and ask them re-broadcast the missing symbols again.



(a) Flow chart of cooperative streaming with smartphones.


User1
User2
User3 (Cache)
Request neighboring users to join the downloading
ok
ok
Cooperatively downloading
Download part of encoded video
Cooperatively decoding
Decoding
Broadcast symbol 2 to others
5 is decoded
Broadcast to others

(b) Illustration of cooperative streaming with smartphones.

Figure 3. Cooperative video streaming among three users.

IV. EXPERIMENTAL AND SIMULATION RESULTS

In this section we first present and discuss the implementation of LT packets dispersing and decoding for 5 mobile devices and show the experimental results as well. Then give the simulation results of cooperative video streaming. Finally, the simulation of cooperative caching plan for small cells is presented and discussed.

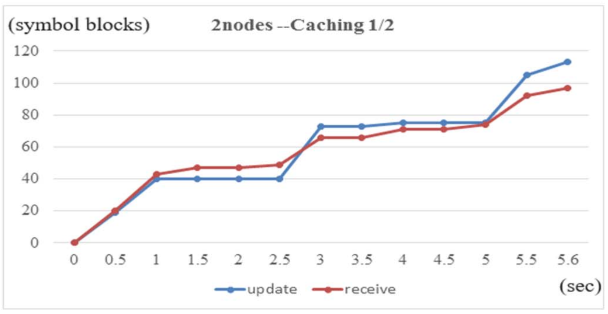
A. LT Codes Implementation on Mobile Devices

1) Implementation Details

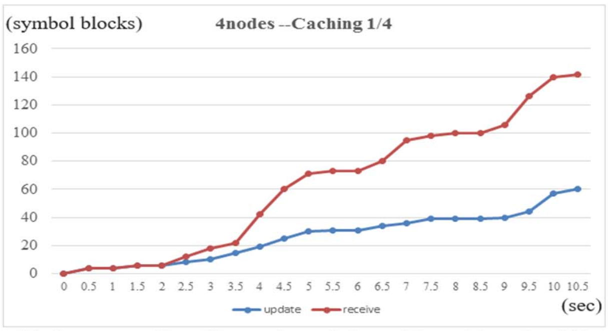
The test video is "Vidyo.yuv" of length of 11 seconds. The working devices consist of two tablets (HTC Nexus 9 and ASUS Nexus 7) and three smartphones (Sony Xperia SP, Sony Xperia Z5, and HTC M9). The steps of LT encoding and dispersing as shown in Fig. 1 (b). First, the video of length of 11 seconds is LT encoded to M packets (with redundancy ε). Second, these packets are dispersed devices randomly. Meanwhile, all devices decode together through trading (sending and receiving) their belongings to regenerate the whole video. The parameters M and ε are set as 240 and 1.2, respectively.

2) Results and Discussions

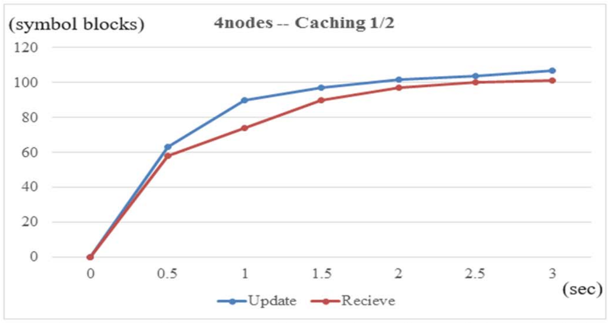
Fig. 4 (a), (b), and (c) depicted computation and communication costs for various settings (# of decoding nodes and cache size). The red line represents the number of sources received from other decoding nodes, and the blue line is the number of sources decoded by the node itself. For example, as shown in Fig. 4(a), in case of (2 nodes, 1/2 cache), each node decodes half of total sources locally, and receives remaining sources from the other node. Comparatively, in Fig. 4(b), in case of (4 nodes, 1/4 cache), each of them decodes less and receives more from the other three nodes. In Fig. 4(c), (4 nodes, 1/2 cache), lots of sources can be decoded rapidly. That is, the more the cooperative nodes, the more the performance gain in decoding time.



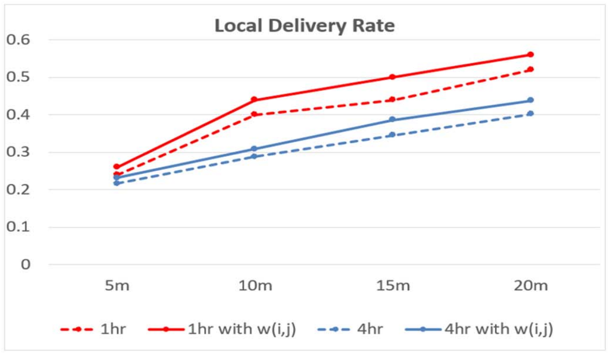
(a) Amounts of sending and receiving with caching size 1/2.



(b) Amounts of sending and receiving with caching size 1/4.



(c) Amounts of sending and receiving with caching size 1/2 and 4 nodes.



(d) Local delivery rate of cooperative streaming with smartphones.

Figure 4. The results of LT packets dispersing and decoding on mobile devices.

B. *Cooperative Streaming with Mobile devices*

1) *Parameters Settings*

Our simulation is implemented over a range of 20×20 square meters. Users are distributed by the following Poisson distribution, where k is the number of nodes, λ represents the density of nodes per unit area, and A is the area size.

$$P(N(A)=k)=\frac{e^{-\lambda||A||}(\lambda||A||)^k}{k!}$$

The real traces from a campus network measurement on YouTube traffic between June 2007 and March 2008 [17] is employed. The busiest ranges of 1 hour and 4 hours within the trace in 01/29/08 are chosen for assessment. 10 popular videos and corresponding 37 users are chosen within the busiest 1-hour interval, 30 popular videos and corresponding 97 users are chosen within the busiest 4-hour interval. The length of video is set to 10 minutes. For example, when User 2 requests for a video, the user will ask his neighboring users to download cooperatively. When user 2 is still watching the video and User 1 also requests the same video, the user does not need to download again because the video has already been cached by User 1.

2) *Results and Discussions*

Fig. 4 (d) shows the *Local Delivery Rate* (*LDR*). The red lines represent 1-hour case, and the blue lines represent 4-hour case. The solid lines are the results of using Eq. (1), and the dashed lines are the results of using Eq. (2). The figures show the benefit of considering time interval between requests timestamp. By comparing two cases, the LDR in 1-hour case is higher because the considering requests are concentrated on a smaller period. In contrast, users find more neighboring users to cooperate in the 4-hour case, however the LDR is lower. Users with similar preferences may not request the same video closely because of the sparse requests.

For considering the limitation of distance between users, if the users are close to each other within 5 meters, the LDR will become lower because most users cannot find cooperative neighbors. When the distance limitation increases up to 15 meters, the LDR is elevated to 50%. The highest LDR only approach to 60% because the requests of real traces are not dense enough, users who send requests closely are limited.

C. *Cooperative Caching Plan for Small Cells*

1) *Parameter Settings*

The real traces from a campus network measurement on YouTube traffic between June 2007 and March 2008 [17][18] are used. The traces from 01/29/08 to 02/12/08 are chosen. The first week traces are for training, and the second week traces are for testing. In our simulation, the most popular 10 videos, corresponding 925 users are selected in the training phase. The users are clustered to 9 clusters and assigned to 9 small cells. Assumed that videos have the same size. Various cache space sizes are compared in our simulation.

Our simulation is implemented on personal computer with Intel(R) Core(TM) i5-4440 3.10GHz CPU, 8.00GB RAM and Win7-64bit OS. In this paper, three *similarity* measures are compared, first measure, Eq. (2), is presented in Section III.A, second measure modifies Eq. (2) using cosine similarity, and third measure (baseline, for the reference purpose only) is random, that is, users are assigned to SCs randomly.

2) *Results and Discussions*

Consider Fig. 5(a)-(b). Three *similarity* measures (Exp (Eq. 2): blue line, Cos: orange, Random: green) are compared with different cache sizes (1-3 cached videos). The solid lines are the results with cooperative caching, and the dashed lines are the results without cooperative caching. Again, the first week traces are for training, and the second week traces are for testing.

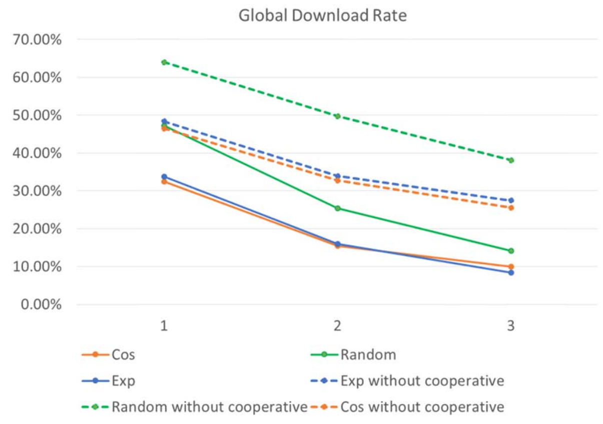
As shown in Fig. 5(a), only the users who have already assigned in the training phase are tested. There are only 218 trained users who issue requests of 10 popular videos are selected in training phase. Consider the Global Download Rate (GDR) depicted in Fig. 5(a) with 3 cached videos. Both the Cos and Exp. with cooperative caching are below 10%, however the Random without cooperative caching is near 38%. It is clear that, the GDR result of clustering is completely better than that of the random assignment. And the GDR is quite high (over 60%) when the cache size is one if without cooperative, since the cache cannot share. And when cache size is too small to cache videos, cache replacement occurs. Sharing cache can decrease the miss rate thus lower the GDR.

As shown in Fig. 5(b), all the users are considered in the testing phase. New coming users are assigned to the appropriate SCs according to the similarity between user's requests and the SCs' profile. There are 908 (218 in case 5(a)) new coming users requesting for top-10 videos in the second week. Most of them change groups less than twice because the number of their sending requests is less. As mentioned before, we have observed a phenomenon, the clusters are either topic-specific or addicted-users. Some new coming users issue requests larger than 10 times, but they change clusters once only. It indicates that users are interested in specific videos, thus they will not change clusters often. The total number of requests is 1950. The improvement of user assignment is revealed even they are not trained. Fig. 5(b) show that our method of assigning the new users is promising. Comparing to the random assignment, when cache size is of three videos (satisfactory), the GDR of our proposed (Exp. with cooperative caching) is 19%, and the random assignment without cooperative caching is above 62%. The meaning of 62% implies cached videos will be frequently replaced, especially for the (YouTube) addicted users, therefore, the data replacement becomes worse.

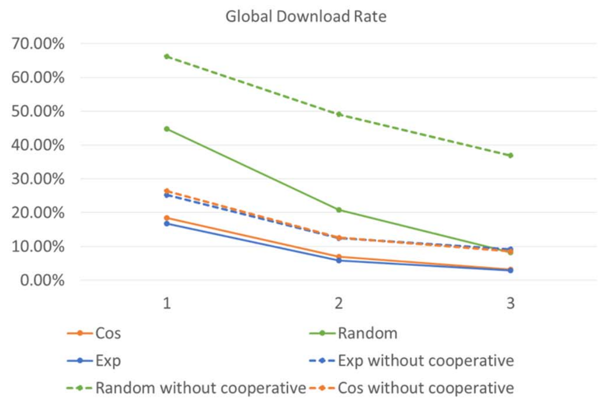
The proposed cooperative caching plan has the benefit of load balancing, as depicted in Fig. 5(e), That is, downloading the encoded packets using LT codes and decoding cooperatively really share and balance the caching space.

Also, the requests of the top-30 videos are taken to evaluate the performance gain. As shown in Fig. 5(c), comparing with Fig. 5(a), the performance gain demotes because more videos are required to be cached concurrently. Limited (poor) cache space leads to more cache miss.

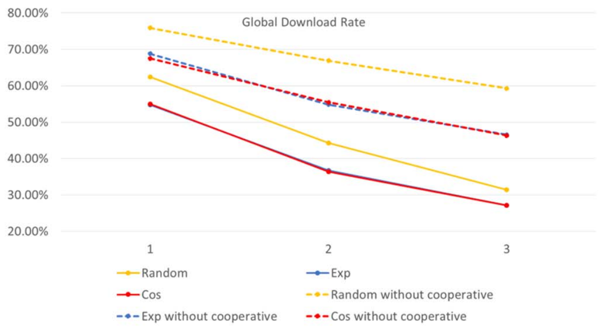
Fig. 5(d) is corresponding to Fig. 5(b), the better performance gain due to plenty of new coming users are topic-specific, therefore, assigning new users to a suitable clusters (SCs) becomes more effective. Comparing to the random assignment, when cache is of three videos, the GDR of proposed method is 33%, however the random assignment without cooperative caching is 64%.



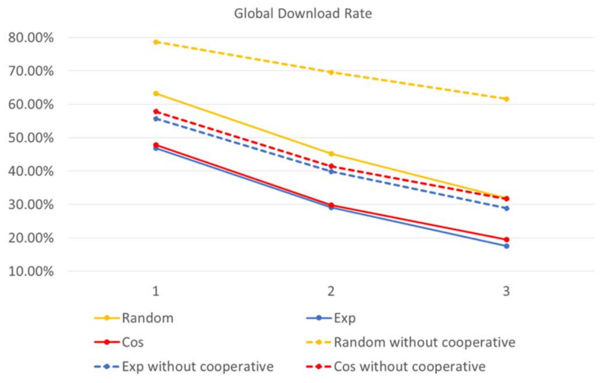
(a) Global download rate and cache sizes for serving trained users.



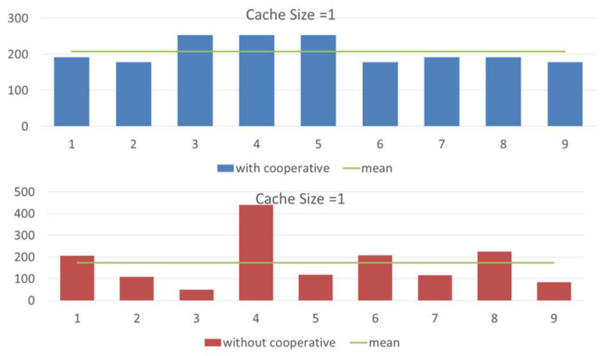
(b) Global download rate and cache sizes for serving all testing users.



(c) Global download rate and cache sizes for serving trained users with 30 videos.



(d) Global download rate and cache sizes for serving all testing users with 30 videos.



(e) Load balancing with cooperative caching.

Figure 5. The results of the proposed cooperative caching plan for small cells

V. Conclusion

Mobile traffic is growing fast, especially mobile video downloading. Caching in the network edge is a way to decrease network latency. A cooperative caching plan is proposed in the paper to achieve the goal of building a cooperative caching method for serving popular video clips over MEC environment with least possible backhaul traffic.

In the training phase, categorize similar users to clusters and to SCs using the spectral clustering algorithm. We reveal a useful phenomenon: there is many clusters of topic-specific, and a few number (may be one or two) of clusters of addicted-users. And the latter always have the larger cluster sizes. We aggregate the users' requests to be the request profile of the corresponding SCs. And, share the caching space among cooperated SCs with the help of distributed LT codes. The (LT) coded packages can be randomly distributed among SCs. Also, can be randomly download instead of selecting the specific data. Furthermore, it also has benefit of load balancing. During the serving phase, new coming users will be assigned to appropriate SCs based on similarity between users and SCs. And, users can achieve the goal of bandwidth aggregation. Our simulation results show that the performance gain (GDR) can decrease from 38% to 10% for trained users, or 62% to 19% for all test users, if cache space is of 3 videos. And decreases from 64% to 33% for trained users, or 77% to 53% for all test users if cache space is of one video only.

References

[1] Forecast, Cisco VNI, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper," *Cisco Public Information*, February 2016.

[2]Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher and Valerie Young, Mobile Edge Computing: A key technology towards 5G, *ETSI White Paper No.* 11, September 2015.

[3] 3GPP TR 36.819, "*Coordinated multi-point operation for LTE physical layer aspects*," V.11.1.0, December 2011.

[4] M Luby, "LT Codes," in Proceedings of the 43rd *Annual IEEE Symposium on Foundations of Computer Science*, November 2002, pp. 271–280.

[5] J.W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," *Proceedings of ACM SIGCOMM* '98, Vancouver, September 1998, pp. 56-67.

[6] Keller, Lorenzo, et al, "MicroCast: Cooperative Video Streaming on Smartphones," the 10th *ACM international conference on Mobile systems, applications, and services*, June 2012, pp. 57-70.

[7] Golrezaei, N., Shanmugam, K., Dimakis, A. G., Molisch, A. F., & Caire, G, "Femtocaching: Wireless Video Content Delivery through Distributed Caching Helpers," *IEEE INFOCOM*, March 2012, pp. 1107-1115.

[8] Shanmugam, K., Golrezaei, N., Dimakis, A. G., Molisch, A. F., & Caire, G. "Femtocaching: Wireless Content Delivery through Distributed Caching Helpers," *IEEE Transactions on Information Theory*, December 2013, pp. 8402-8413.

[9] M. S. ElBamby, M. Bennis, W. Saad, and M. Latva-aho, "Content-aware user clustering and caching in wireless small cell networks," *IEEE Intl. Symp. on Wireless Communications Systems* (ISWCS), Barcelona, Spain, August 2014, pp. 945–949.

[10] Chia-Cheng Yen and Jia-Shung Wang, "Distributed delivery of popular videos over Ultra-dense networks," *IEEE Symposium on Computers and Communication* (ISCC), Larnaca, Cyprus, July 2015, pp. 116-121.

[11] Shangguang Wang, Yali Zhao, Lin Huang, Jinliang Xu and Ching-Hsien Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Comput*ing, Accepted 22 September 2017.

[12] Wang, Zheshen, et al, "Youtubecat: Learning to categorize wild web videos," *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), June 2010. pp. 879-886.

[13] U. Gargi, W. Lu, V. Mirrokni, and S. Yoon, "Large-scale community detection on YouTube for topic discovery and exploration," the Fifth *International AAAI Conference on Weblogs and Social Media*, July 2011.

[14] Yang, Jaewon, and Jure Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," the sixth *ACM international conference on Web search and data mining*, February 2013, pp. 587-596.

[15] Ng, Andrew Y., Michael I. Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, 2002, pp. 849-856.

[16] Newman, Mark EJ. "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, 2006, pp. 8577-8582.

[17] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network-measurements, models, and implications," *Computer Networks*, Vol. 53, No. 4, March 2009, pp. 501-514.

[18] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube Traffic Characterization: A View from the Edge," the *ACM SIGCOMM conference on Internet Measurement* (IMC), 2007, pp.15-28.