See in the Future: Dynamic Coded Caching Facilitated Content Distribution over Mobile Internet

Author
Affliation
e-mail

***Abstract*—Recently, a distributed caching technique called coded caching is shown to significantly reduce the peak-traffic rate over the shared wireless link such as in the cellular networks. This technique can utilize the distributed memory resources and coded multicast gain to alleviate the pressure of wireless interface. However, the current codec schemes assume uniform and static content popularity, while the real experience reveals that the content popularity is nonuniform and dynamic. Thus they are infeasible in practical scenario. In this paper, we first point out that the traditional dynamic codec scheme Least Recently Sent is categorized into pre-fetching the most popular contents based on a Lyapunov stability analysis and show its inefficiency. Moreover, we construct a more efficient coded caching framework to deal with such explosive data traffic considering the characteristics of content popularity. Under the specific problem structure, two key techniques are proposed. The first technique adopts an nonlinear optimization model to determine the optimal cache allocation under given nonuniform content popularity and codec scheme. Based on the excavated user navigation pattern, the second technique takes a high-order Markov model to update the local cache. We periodically combine these two techniques so as to minimize the long-term average traffic rate over the shared link. A trace-based simulation shows that our scheme can reduce the traffic volume in long time scale compared to the traditional art.**

I. Introduction

Data traffic in wireless network is predicted to increase in a more and more massive scale and beyond 2020, it is 1000 times higher than that in 2010 [1], which will place tremendous pressure in modern infrastructure such as backhaul and base station. This explosive tendency is driven mainly by the proliferation of smart user terminals (UT) and various multimedia request. One of the most promising ways to achieve data traffic decrease of such dreadful scale is to bring the content closer to the user, which is implemented by deploying cache into the edge of the wireless network such as UTs. Practically, the traffic volume in wireless network shows high temporality such that caching technique can balance this traffic load in long time scenario: during low-traffic time, most popular contents are prefetched into the local memory space; while in peak-traffic time, only the requests beyond the local cache are required to be delivered thereby reduce the traffic volume over the wireless link. There are large amount of works [3]– [6] investigating such *local cache gain* in different caching networks.

The *local cache gain* only focuses on how to localize the traffic volume to the maximum, i.e., how to best assign the contents into different cache so that users can retrieve their requests via local traffic. This technique is implemented by unicasting, which does not consider the homogeneity of user requests, thus its efficiency is dependent on the proportion of prefetched contents and is linear to the cache size. To exploit the potential of users' distributed cache, a novel scheme called coded cache is proposed to achieve a *global cache gain*, which jointly designs the placement and delivery phase [7]. The main feature of this scheme is prefetching the segments of the contents in the placement phase and multicasting the coded data of several content segments upon users' requests in delivery phase, based on which the users can retrieve the requests. [8] extends this technique to the decentralized case without prior knowledge of the number of users. The above techniques are proved to reduce the traffic volume significantly in the theoretical sense, however, they are under the assumption of uniform content popularity and static user demands, which is unreasonable in practice scenario.

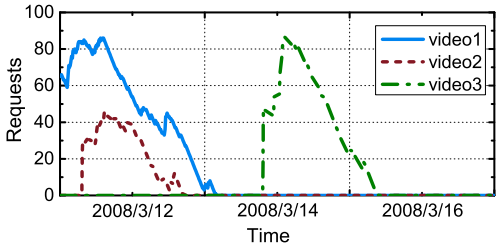


Fig. 1: Number of requests in the UMass for three Youtube videos from 2008/3/11 to 2008/3/17.

We observe that demands for video streaming services are nonuniform within contents and shifty at different times. Fig. 1 shows the requests of three videos in 7 days by analyzing 810,000 worth of operational traces in Massachusetts Amherst University (UMass) network [30]. As we can see, the number of requests increases rapidly on the release day, stays relatively high for a number of hours and then drops, which suggests the dynamic of the content popularity. Besides, the number of requests for three videos are different at certain time, which suggests the nonuniformity of the content popularity.

In this paper, we propose a more practical codec scenario to deal with the "mobile data tsunami", consisting of single origin server connected through a shared link to a number of users each equipped with a cache of finite size. The users issue a sequence of requests and the server is required not only multicasting the coded data but also dynamically updating the user's local cache. Since the coded multicasting scheme is sort

of linear combination and is independent of content characteristics, in particular, we formulate the following problem:

Dynamic Coded Caching Problem: For given user storage capacity and nonuniform user requests of each time, how should the user local cache be dynamically allocated such that the long-term traffic volume over the shared link is minimized?

In static case, Niesen et al. propose to classify the contents into groups based on their popularity to operate the codec scheme in each sub-group independently [9] and optimize the cache allocation for each sub-group, which may miss the iter-group coding opportunities and assumes that user demands are static. While in dynamic case, if the content popularity is uniform, Pedarsani et.al design a coded *Least Recently Sent* (LRS) strategy [10], which incorporates a simple cache update strategy with the codec scheme in [8]. If considering the nonuniform content popularity, we show that LRS strategy is approximate to prefetching the most popular contents and thus inefficient in codec scheme. In this situation, how to best assign the memory space in local cache and make the cache update becomes a much more complicated issue, as we will demonstrate. The contribution of this work is three-fold:

- We model the traditional cache update strategy LRS as a stochastic iterative process for the first time and point out this technique is approximate to prefetching the most popular contents based on constructing the comparison principle and Lyapunov stability analysis, and show its insufficiency in nonuniform codec scenario.
- We propose a practical codec scenario considering the content popularity characteristics and regard this problem as a periodical combination of two sub-problems: optimal cache allocation and cache update. To solve the first problem, we adopt a non-linear optimization model to determine the optimal caching distribution under the codec scheme in [8].
- To solve the second problem, we excavate the user navigation pattern based on a trace of Youtube video requests in Umass campus network and find that the user always requests a video related to the video he requested before and has a preference of specific group of contents. On account of the above pattern, we propose a high-order Markov prediction method to predict the future content popularity, by which we determine an ultramodern optimal caching distribution and make the cache updating of all users.

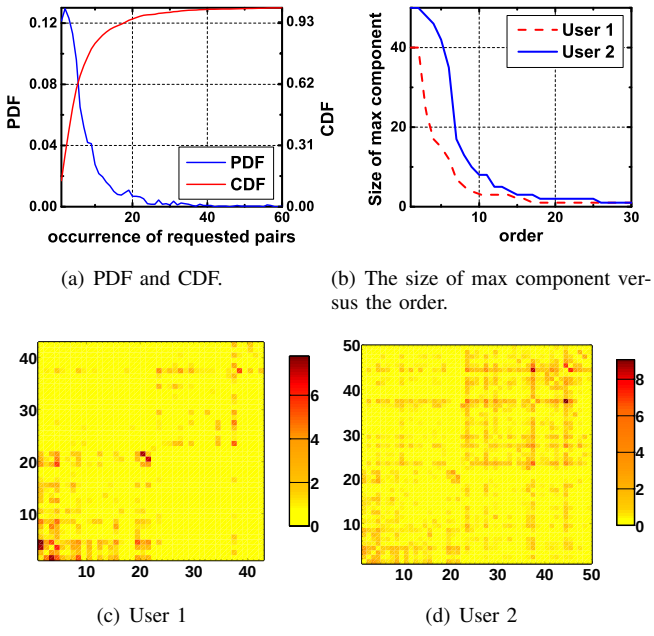
The remainder of the paper is organized as follows. The preliminary is spread in Section II. Next, we present the service model and design challenges. In Section IV, we formulate the optimal caching distribution problem by providing the coded caching optimization model. In Section V, we present our theoretical analysis on the traditional technique LRS and propose a prediction method. Simulation results are given in Section VI. Finally, we summarize this work and present the future in last section.

II. Preliminary

In this section, we present the observations of user navigation pattern and introduce the traditional codec scheme.

A. User navigation model

The way users navigating in the content pool depends not only on their interests, but also on the content structure. A user, who has currently requested content A, will request the content mainly among the homogeneous content set. Namely, the future request is influenced by the previously requested contents. For example, Lennon watches the compilations of soccer clips about Argentina versus Netherland in Youtube, then there is a good chance that he will watch the clips about Germany versus Brazil, because they are all categorized into the semifinal in World Cup 2014. The above case implies dependencies among the requested sequences. If the dependencies correlate only pairs of contents, then they are called *first order dependencies*. Otherwise, they are called *high order dependencies* [11]. Particularly, the user navigation model describes a Markovian process over a graph. The nodes in the graph represent contents and there exists an arc from node A to node B if the user requests content B after requesting content A.



(a) PDF and CDF.

(b) The size of max component versus the order.

(c) User 1

(d) User 2

Fig. 2: Statistics about occurrence of requested pairs.

To better understand the user's request pattern, we examine the traces from the UMass campus network [30]. It collects 243023 client requests for YouTube video clips from 2008/3/11 to 2008/3/17. Our study focuses on the dependencies among user request sequences and discovers the following trends.

Uneven distribution of first order dependencies: Fig. 2(a) is the PDF and CDF for the occurrence of different requested

pairs of 4 users's requested sequences[1]. While 87%[2] of the requested pairs have occurrence larger than 1, only 3% of them have an occurrence larger than 2. More visibly, Fig. 2(c) and Fig. 2(d) uses different levels of colormap scale to reflect the occurrence of different requested pairs of two users' requested sequence (Note that here we combine the multiple users' request into one user). It can be found that there exists some hotspots, i.e., the user 1 always requests content 21 after requesting content 20, the user 2 always request content 38 after requesting content 45, several other requested pairs also exhibit high occurrence.

Existence of high order dependencies: The high-order dependencies is much more complicated than the first order. For example, we define an ordered pair $\langle i, j\rangle$ to represent the arc from node i to node j. In high order dependencies, the pair $\langle i, j\rangle$ and $\langle j, k\rangle$ contains the pair $\langle j, k\rangle$.

To investigate the high order dependencies of the user navigating pattern, we define a *k-order Navigating Graph* for a specific user. In this graph, the nodes represents user's requested contents. If the times that the user requests content j after requesting content i is larger than k, we elicit an arc from node i to node j. Standing on the user navigating graph, we determine the maximum strongly connected components via *Tarjan* algorithm [12]. Fig. 2(b) presents the size of maximum strongly connected component versus the order k and shows a trend that when the order k becomes large, the size firstly decreases slowly, then drops rapidly when k is larger than a threshold, and finally becomes flatten. This phenomenon implies there is a giant component in $1-$order navigating graph and exists strong high-order dependency. On the other hand, when the order increases, the only giant component will break up into two or more components, thus the size of max component will show exponential fall, which means that the arc's weight of this giant component is uniform and the user has a fair preference for these contents.

The above observations show that the user navigate the Internet in a specific *associate rule* and make a case for predicting the user's future requests such that pre-operating the user's local cache. The potential benefit is two-fold: increased accuracy of predicting content future popularity, in addition, reducing the future traffic over the wireless link.

B. Coded cache

Coded caching, recently introduced in [7]– [9], is a novel technique to mitigate the wireless traffic volume during the peak-traffic time. It introduces a *global cache gain* on network load based on distributed cache and coded multicast delivery. Namely, the data delivered over the wireless link are coded data over multiple contents, so that multiple requests for even different contents can be satisfied with a single coded transmission. We make essential use of the offline coded caching scheme from [8] in the present paper. Therefore, we now briefly overview this scheme.

[1]explain four users
[2]explain 87%

Algorithm 1: Decentralized coded caching scheme with uniform demands

Placement Phase
for $(k = 0; k < K; k++)$ **do**
  for $(n = 0; n < N; n++)$ **do**
    user k randomly prefetches MF/N bits of content n ;

Delivery Phase
for $(k = K, k > 0; k--)$ **do**
  for *choose* k *users from* K *users to form a subset* U **do**
    server sends $\oplus_{k\in U} V_{k,U/\{k\}}$ to users in U.

The set in [8] is an offline version, which assumes that the popularity of each content is fixed and uniform. It considers a similar architecture that has N contents and K users. Each user has a local cache of size MF bits with $M \leq N$. The coded caching scheme under such scenario operates as follows. In the placement phase, each user randomly caches the same amount MF/N bits of each content. Then in the later delivery phase, all users send their requests and the server transmits multiple coded signals $\oplus_{k\in U} V_{k,U/\{k\}}$[3] over the shared link to users to retrieve their requested contents. This scheme achieves a peak traffic rate of

$$R(M, N, K)F = K\cdot(1-\frac{M}{N})\cdot\frac{N}{KM}\cdot(1-(1-M/N)^K)F, \quad (1)$$

which is shown to be within a constant multiplicative gap of the information theoretical lower bound. Illustration of this scheme is provided in Example 1.

Example 1 (Codec scheme in [8]). Suppose that there is a simple system distributing $N = 2$ contents A and B to $K = 2$ users, each with the cache size MF bits. In the placement phase, each user randomly caches $MF/2$ bits of content A and B independently. Let us focus on content A. The operations of placement phase partitions content A into four subcontents,

$$A = (A_\emptyset, A_1, A_2, A_{1,2})$$

where $U \subset \{1, 2\}$, A_U denotes the bits of content A that are prefetched in the memories of users in U. For example, A_1 represents the bits of A only available in first user's memory. We adopt $|\cdot|$ as the operation of size, thus,

$$|A_\emptyset| = (1 - M/2)^2 F \text{ bits};$$
$$|A_1| = |A_2| = (M/2)(1 - M/2)F \text{ bits};$$
$$|A_{1,2}| = (M/2)^2 F \text{ bits}.$$

The same analysis holds for content B.

In the delivery phase, we assume that user 1 and user 2 request content A and B, respectively. User 1 has accessed to subcontent A_1 and $A_{1,2}$ in its local cache and lacks $A_\emptyset$ and A_2. Similarly, user 2 has already accessed to B_2 and $B_{1,2}$,

[3]$V_{k,U/\{k\}}$ denotes the bits of the content d_k requested by user k cached exclusively at users in U

and lacks $B_\emptyset$ and B_1. In traditional uncoded caching scheme, the server is required to unicast $A_\emptyset$ and A_2 to user 1 and unicast $B_\emptyset$ and B_1 to user 2. The total rate is

$$2(M/2)(1-M/2)F + 2(M/2)^2F = MF \text{ bits},$$

Under the coded caching manner, the server can satisfy the requests by transmitting $A_\emptyset$, $B_\emptyset$ and $A_2 \oplus B_1$ over the shared link, where $\oplus$ denotes the bit-wise XOR operation. The rate over the shared link is

$$(M/2)(1-M/2)F+2(M/2)^2F = R(M,2,2)F < MF \text{ bits},$$

with $R(M,N,K)$ is defined in (1). For other possible user requests, this rate is also achievable.

From **Example 1**, it can be found the gain of uncoded cache scheme comes from the isolated cache memory, called the *local cache gain*, captured by a factor $(1 - M/N)$ in (1), which linearly decreases with the cache size MF. Instead, the main gain of coded cache scheme comes from the multicasting opportunities created by a jointly designed placement and delivery phase. More concretely, when the size of memory space increases, this gain will increase as well and can be captured by a factor called *global cache gain*, $N/K/M \cdot (1-(1-M/N)^K)$ in (1). This gain is inverse proportional to the cache size M and much more significant in aspects of reducing the traffic volume.

This codec scheme is implemented by a decentralized manner, which requires that the user's memory is filled independently and randomly and multiple coded signals are delivered.

III. Service Model and Design Challenges

A. Service Model

The coded caching network architecture studied in this paper is illustrated in Fig. 3 and main notations are listed in Table I. There is an origin server connected to K users through a shared link, i.e., the long term evolution (LTE) broadcasting system [17], where each user has a local cache space of size MF bits. In our model, we make the following assumptions:

- Contents are with various levels of popularity that is measured by the probability the content will be requested.
- The size of content is assumed to be identical with F bits. This assumption is mainly for notational convenience, and could be easily lifted, because the main body of content objects can be tailored as the same size for coded caching based distribution and the rest in a small quantity can be distributed in the traditional way.
- The size of content set V is fixed to N. Since there exists a birth-death process in the content set, the number of contents is fluctuating within the time scale. We introduce a more finely ingrained metric, called the content popularity evolution, to describe this process. For example, the popularity of each content is dynamic and change to zero if a content leaves the origin server.
- The content popularity distribution evolves slowly. For example, the popular news and short videos are updated every 2 hours – 3 hours, new movies are posted every week, new musics are posted about every month. This assumption makes the content distribution more tractable and boosts the predictive procedure.

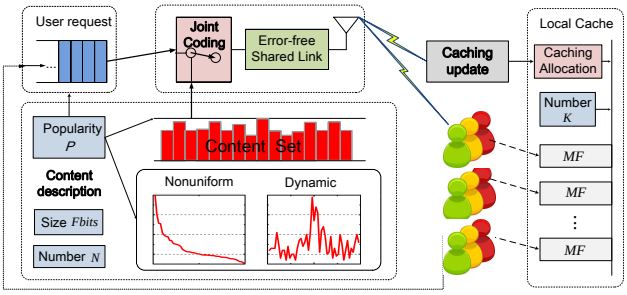


Fig. 3: Network architecture of coded caching

TABLE I: Main notations

N	The number of contents to be distributed.
K	The number of users in the system.
F	The size of each content.
V_i	Content i and $V_i \in V$.
$P(t)$	The content popularity distribution, $P = [p_1(t), p_2(t), ..., p_N(t)]$, where $p_i(t)$ is the popularity of content i measured by the probability content i is requested in time t.
$Q(t)$	The caching distribution, $Q(t) = [q_1(t), q_2(t), ...q_N(t)]$, where $q_i(t)$ is the proportion of cache space allocated for content i in time t.
M	The number of contents can be prefetched by each user.
R_t	$R(P(t), Q(t))$, the average traffic volume under popularity distribution $P(t)$, caching distribution $Q(t)$ and specific codec scheme.
$d_k(t)$	The index of the content requested by user k in time t.
$\vec{s}$	$\vec{s} = [\alpha_1, \alpha_2, ...\alpha_N]$, the request situation, where users request α_1 content 1, α_2 content 2,..., and α_N content N.
S	All possible request situations.
P_s	$P_s(\vec{s}, P(t))$, the probability that situation $\vec{s}$ occurs under the popularity distribution P.
R_s	$R_s(\vec{s}, Q(t))$, the average traffic volume of situation $\vec{s}$ under caching distribution $Q(t)$ and code caching scheme.

In the initial placement phase, part of each content is prefetched in each user's local cache, and the amount of cache space allocated for each content depends on the initial caching distribution $Q(1) = [q_1(0), q_2(1), \ldots, q_N(1)]$, where $q_i(t)$ is the proportion of cache space allocated for contetnt i. At time $t \in T$, each user k requests a content $d_k(t)$ from a content set V with cardinality $N \geq K$. The K user requests, collectively denoted by a vector $d(t) = [d_1(t), d_2(t), \ldots, d_K(t)]$, are chosen based on content popularity distribution $P(t) = [p_1(t), p_2(t), \ldots, p_N(t)]$ from content set V. The server responds request $d(t)$ by transmitting a coded signal as shown in Algorithm 1 with size of $R(t)F$ bits. Simultaneously, each user update their local cache.

B. Design Challenges

Since the popularity distribution is nouniform and dynamic in the long time range, the offline version in [8] is inappropriate. The following example shows the inefficiency of this scheme.

Example 2 (Dynamic coded caching) Consider a system with $N = 3$ contents and there are $K = 2$ users with cache size of $M = 1$. The initial caching distribution is $Q(1) = [1/2, 1/2, 0]$ and we assume the content set is $\{A, B, C\}$.

t=1: The popularity distribution is $P(1) = [1/2, 1/2, 0]$, thus the content set is factually $\{A, B\}$. Assume the users requests are $d(1) = [A, B]$. Both requested contents are partially prefetched at the user-side. In the delivery phase, the server sends $A_\emptyset$, $B_\emptyset$ and $A_2 \oplus B_1$. This results in a volume of $R(1, 2, 2) = 1$, with $R(M, N, K)$ defined in (1).

t=2: The popularity distribution evolves to $P(2) = [0, 1/2, 1/2]$, thus the content A departures and content C arrives to the server. Assume the user requests are $d(2) = [B, C]$. Since content C is not prefetched, in the delivery phase, the server sends $B_\emptyset$, B_1 and C, which results in a volume of $(1-1/2)^2+(1/2)(1-1/2)+1 = 3/2$. For comparison, we define two cache distributions $Q_u(3) = [0, 1/2, 1/2]$ and $Q_{nu}(3) = Q(2)$. Note that $Q_u(3)$ takes a cache update and evicts the content A and is replaced by content C.

t=3: The popularity distribution is $P(3) = [0, 1/2, 1/2]$. Assume the requests are $d(3) = [B, C]$. Under the caching distribution $Q_u(3)$, both of the requested contents are partially prefetched at the user-side. The server only sends $B_\emptyset$, $C_\emptyset$ and $B_2 \oplus C_1$ with rate $R(1, 2, 2) = 1$. Instead, if we do not take a cache update and adopt caching distribution $Q_{nu}(3)$, it will yield a rate $3/2$!

Therefore, a cache update procedure is required to make the contents at users as "fresh" as possible. Two possible solutions to this problem are the famous *Least-recently used* (LRU) eviction policy [19] and the LRS strategy, which is modified from LRU. However, this strategy analyzes the theoretical performance based on an arrival/departure process without considering the content popularity. Moreover, we will show this strategy approximates to prefetching the most popular contents under the nonuniform content popularity and show its inefficiency in the following sections.[4]

Practically, the users' requested sequence is related to the popularity distribution, thus it is possible to update the user cache based on the popularity distribution to reduce the traffic volume and we divide the dynamic coded caching problem into the following two sub-problems.

Optimal caching distribution problem: If given the content popularity distribution and the cache capacity, how to allocate the local cache, i.e., determine a caching distribution, to minimize the traffic volume over the shared link?

Popularity prediction problem: If given the past user-requested sequence or previous content popularity distribution, how to predict the future content popularity distribution?

Based on the above sub-problems, we can take a periodical cache update procedure: firstly predict the future content popularity distribution P_{new}; then determine the optimal caching distribution Q_{new} based on P_{new} ; finally, update the user cache according to Q_{new}. The complete procedure is illustrated in Algorithm 2. Note that $\varepsilon(x)$ denotes the step function.

[4]say LRS is uniform

Algorithm 2: Dynamic Coded Caching

while *in time* t **do**
- New $\hat{P}(t) \longleftarrow$ predicting based on previous user requests $d(t-1), d(t-2), \ldots, d(t_0)$;
- Determine the optimal caching distribution $\hat{Q}(t)$ under the $\hat{P}(t)$;
- **Cache update**
- **for** $(k = 0; k < K; k++)$ **do**
  - **for** $(n = 0; n < N; n++)$ **do**
    - Sever randomly sends $\varepsilon\{\hat{q}_n(t) - q_n(t-1)\}M$ bits of content n.
- **Coded delivery**
  - Collect users' requests and take the delivery phase of Algorithm 1;

A natural intuition for the first problem is that the optimal caching distribution Q should follow the content popularity P. However, the following section will show that the influence of P on Q is not so straightforward as intuitively imagined and this problem has a non-trivial structure.

Predicting popularity has received much attention in the field of Web prefetching, which aims to prefetch the most popular Web documents and alleviate the future access [13]. It focuses on mining the "associate rule" among user accesses, which is user-oriented. Different users may have different predictive structures, i.e., a dependence graph or a PPM tree [14]–[16]. However, there is an important component in coded cache that the configuration of each user's local cache is identical. We cannot adopt the traditional Web prefetching technique straightforwardly in the cache update phase of dynamic codec scheme. Instead, we should further exploit the unique problem structure of coded cache and excavate the content request pattern to develop a solution that is specially tailored to our needs.

IV. Analysis for LRS strategy

Before presenting our scheme, we first analyze the intrinsic anomaly of the current LRS strategy under the nonuniform demands. In the dynamic codec scenario, the prefetching content of user k at time t is function of the prefetching content of the same user at previous time $t-1$, the coded signal over the shared link at time $t-1$ and the requests d_τ with $\tau < t$. In the aspects of the long time scale, the prefetching content of user k at time t is the function of the recent content popularity distribution and the prefetching content of the same user at the previous time. More concretely, how the recent content popularity distribution affects the caching distribution is based on the precise cache update strategy. The LRS strategy proposed in [21] is shown by the following Definition:

Definition 1. ***(LRS)*** *In each time t, all users replaces the least recently sent contents in their cache with a random subset of*

MF/N' *bits of each current requested contents, where* $N' = \alpha N, \alpha \geq 1$.

The LRS updates each user's local cache in an uniform manner and adopts an eviction rule in accord with all user's request $d(t)$. Under our dynamic codec scenario, the property of arrival and departure of the contents in the server is embodied by the evolution of content popularity distribution. Subsequently, the caching distribution is also evolutive and, based on the previous study, this evolution procedure can be defined by the following stochastic iterative function:

$$Q(t+1) = \Psi(Q(t), P(t+1)) \tag{2}$$

Since the content popularity evolves slowly, we assume there are plenty of cache update procedures from time t to $t+1$. Formally, we can regard the LRS strategy in time period $\mathbf{T_t}$ as the following stochastic iterative process:

$$x(k+1) = x(k) + f[x(k), w] \tag{3}$$

where $x(k) \in \mathbb{R}^N$ is the state of the caching distribution in step $k \in \mathbf{T_t} = t_0 + \{0, 1, 2, \ldots\}$ and w is a multiple random variables with an underlying complete probability space $(\Omega, \mathfrak{F}, P)$. The initial state $x_0 = x(k_0)$ is $\mathfrak{F}$-measurable. We assume the function f is Borel measurable to guarantee the existence of the $\mathfrak{F}$-measurable process $x(k; k_0, x_0)$ for all $k, k_0 \in T_t, k \geq k_0$, and $x_0 \in \mathbb{R}^N$. Therefore, the entire cache update procedure is constructed by the function (2), each of which can be regarded as a finite stochastic iterative process (3). Here we derive $f[k, x(k), w(k)]$ of the LRS strategy.

There are three states of $x_i(k+1)$: 1) zero (content i is least recently sent one in time t); 2) $x_i(k)+x_r(k)$ (content i is new requested one in time t, where r is the index of the replaced content); 3) $x_i(k)$ (content i is nether the least recently sent one nor the new requested one). Therefore, the $x(k+1)$ is linearly dependent on $x(k)$, thus the $f[k, x(k), w(k)]$ is a linear function and Borel measurable that $f[k, x(k), w(k)] = Ax(k)$ and the stochastic iterative process for LRS is

$$x(k+1) = x(k) + Ax(k) \tag{4}$$

Lemma 1. *A is a matrix of $N \times N$ random variables, and obeys the following bernoulli distribution $\mathfrak{F}_c$:*

$$a_{ii} = \begin{cases} 0, 1 - \mathbb{P}(H_i) \cdot \sum_{j \neq i} \frac{p_j}{1 - \mathbb{P}(H_j)} \\ -1, \mathbb{P}(H_i) \cdot \sum_{j \neq i} \frac{p_j}{1 - \mathbb{P}(H_j)} \end{cases}, 1 \leq i \leq N \tag{5}$$

$$a_{ij} = \begin{cases} 0, p_i \cdot \frac{\mathbb{P}(H_j)}{1 - \mathbb{P}(H_i)} \\ -1, 1 - p_i \cdot \frac{\mathbb{P}(H_j)}{1 - \mathbb{P}(H_i)} \end{cases}, j \neq i, 1 \leq j \leq N \tag{6}$$

where H_i represents the event that content i is least recently sent one and $\mathbb{P}(H_i) = p_i e^{-p_i \kappa}$, κ is the solution of equation $\sum_{i=1}^{n} e^{-p_i \kappa} = N - M$.

The Lemma 1 presents the $\sigma-$field of the complete probability space for LRS and the proof can be found in Appendix VII-B. Based on above results, we then focus on analyzing the stability of the stochastic iterative process (4) and determining its stable point.

Definition 2. ***(Mean Convergence [22])*** *An iterative process (3) is said to be convergent in the mean x^* if for every $k_0 \in \mathbf{T_t}$ and any $\epsilon > 0$ there exists $k_1(k_0, \epsilon) \in \mathbf{T_t}, k_1 > k_0$, and $\delta(k_0, \epsilon) > 0$ such that $\|x_0\| \leq \delta(k_0, \epsilon)$ implies*

$$\mathbb{E}[\|x(k; k_0, x_0) - x^*\| | \mathfrak{F}_c] < \epsilon, \forall k \geq k_1 \tag{8}$$

As in the case of convergence in the moment given in Definition 2, the mean convergence is concerned with the average convergence of all realizations of the process. Factually, there are many kinds of convergence, such as *almost sure convergence, p-convergence* and *mean square convergence*. The reason we choose *mean convergence* is that we should investigate the average performance of this strategy. To establish this convergence, we construct the following lemma and introduce the following theorems.

Lemma 2. *$\overline{A}$ is a constant matrix that the element $\bar{a}_{ij} = \mathbb{E}[a_{ij} | \mathfrak{F}]$ and an $\overline{A}-$dominated iterative process is defined as*

$$x(k+1) = \overline{A} x(k) \tag{9}$$

Under the distribution (5)(6) and the condition that $\sum_{i=1}^{N} x_i(k) = 1$, the process (9) has unique asymptotically stable point x^* if there exists a positive definite matrix P such that $\bar{A}^T P \bar{A} - P = -I$, and

$$x_i^* = \frac{e^{p_i \kappa} / (1 - p_i e^{p_i \kappa})}{\sum_{j=1}^{N} e^{p_i \kappa} (1 - p_j e^{p_j \kappa})}, 1 \leq i \leq N. \tag{10}$$

Proof: According to definition, the $\bar{a}_{ii} = \mathbb{E}[a_{ii} | \mathfrak{F}_c] = 1 - \mathbb{P}(H_i) \cdot \sum_{j \neq i} p_j / (1 - \mathbb{P}(H_j))$ and $\bar{a}_{ij} = \mathbb{E}[a_{ij} | \mathfrak{F}_c] = p_i \cdot \mathbb{P}(H_j)(1 - \mathbb{P}(H_i))$. The x^* satisfies $\overline{A} x^* = x^*$, solving this linear simultaneous equations under the condition $\sum_{i=1}^{N} x_i^* = 1$ and (38), we can get (10). Based on Liapunov's stability theorem, the x^* is the stable point if there exists a positive definite matrix P such that $\bar{A}^T P \bar{A} - P = -Q$, where Q is a given positive definite matrix. Due to the extreme complexity of deriving matrix P, we omit the asymptotically stable analysis of process (9) and it is easy to realize this procedure under the numerical P. ■

The Lemma 2 presents a stable point of process (9) under specific condition. We further show the process (4) is mean convergent to the point x^*. Consider a comparison equation:

$$b(k+1) = b(k) + z[b(k)] \tag{11}$$

where $b(k) \in \mathbb{R}^N$ and $b(k_0) = b_0$.

Theorem 1. *There exists a function $v \in C[\mathbb{R}^N, \mathbb{R}_+]$ satisfies the inequality $\mathfrak{L} v(x) \leq z[v(k, x)]$, where $z \in C[\mathbb{R}_+, \mathbb{R}_+]$ and is a concave function. The $\mathbb{E}[v(x) | \mathfrak{F}_c]$ exists for any solution process $x(k)$ of (3). Then*

$$\mathbb{E}[v(x) | \mathfrak{F}_c] \leq s(k), \forall k > k_0 \tag{12}$$

where $s(k)$ is the solution of process (11) for $k > k_0$.

The detail proof is shown in Appendix VII-C and the symbol $\mathfrak{L}$ is defined as

$$\mathfrak{L} v = \mathbb{E}\left\{ \int_0^1 \frac{d}{dx} v\left[x + \delta f(x, w)\right] d\delta \middle| \mathfrak{F}_c \right\}. \tag{13}$$

$$\begin{aligned}\mathfrak{L}v(x) =&\mathbb{E}\left\{\int_0^1 v_x\left[(x-x^*)+\delta A(x-x^*)\right]d\delta\middle|\mathfrak{F}_c\right\}\\ =&x^T(\bar{A}^TD+\bar{A}^TD\bar{A}+DA)x-(x^*)^T(D+2DA)x-x^T(2\bar{A}^TD+D)x^*+3(x^*)^TDx^*\end{aligned}\tag{7}$$

Theorem 2. *If there exists a positive definite matrix P such that $\bar{A}^TP\bar{A}-P=-I$, the stochastic iterative process (4) is convergent to x^* in the sense defined in the Definition 2.*

Proof: We define a function $v(x)$ as

$$v(x)=(x-x^*)^TD(x-x^*)\tag{14}$$

where $D=diag\{d_1,d_2,\ldots,d_N\}$ and $d_i>0$ and formulate $\mathfrak{L}v(x)$ with respect to (4) and $\overline{A}x^*=x^*$, we get (7). After a complicated algebraic operation of (7), we arrive at the comparison inequality

$$\mathfrak{L}v(x)\le z[v(k)]=-\rho v(x)+h(k),\tag{15}$$

where ρ is a constant satisfies $\rho>0$ and $h(k)=x^T(A^TD+DA+A^TDA+\rho D)x-(x^*)^T[(1+\rho)D+2DA]x-x^T(2A^TD+(1+\rho)D)x^*+(3+\rho)(x^*)^TDx^*$. With the corresponding comparison equation

$$b(k+1)=b(k)+z[b(x)]=b(k)-\rho b(k)+h(k).\tag{16}$$

By applying Theorem 1, we conclude that

$$\mathbb{E}[v(x)|\mathfrak{F}_c]\le s(k),\forall k>k_0,\tag{17}$$

where $s(k)$ is the solution of (16) expressed as

$$s(k)=(1-\rho)^{k-k_0}b_0+\sum_{i=k_0}^{k-1}h(i)(1-\rho)^{k-1-i},\tag{18}$$

and $\mathbb{E}[v(x_0)]\le b_0$. Considering the fact $1-t\le e^{-t}$, facilitate us to estimate $s(k)$ as

$$s(k)\le b_0e^{-\rho(k-k_0)}+h(k)\left[1-e^{-\rho(k-k_0)}\right].\tag{19}$$

When $k\to\infty$, $h(k)=\rho(x-x^*)^TD(x-x^*)$. For any given $\epsilon>0$, there exists $k_2(\epsilon)$ and constant matrix D such that $h(k)<\epsilon/2, k>k_2$. Thus we get

$$s(k)\le b_0e^{-\rho(k-k_0)}+\frac{\epsilon}{2}\left[1-e^{-\rho(k-k_0)}\right].\tag{20}$$

Using the fact that $\lim_{k\to\infty}-\rho(k-k_0)=0,\rho>0$, we conclude from (20) that $s(k)\to0$ as $k\to0$. For the function $v[x(k)]$ defined in (14), we assume existence of the estimates

$$\phi_1(\|x-x^*\|)\le v[x(k)]\le\phi_2(\|x-x^*\|)\tag{21}$$

where $\phi_1\in\mathfrak{K}_e,\phi_2\in\mathfrak{K}_a$. The symbols $\mathfrak{K}_e$ and $\mathfrak{K}_a$ denotes the sets of convex and concave $\mathfrak{K}$ functions, respectively [25]. The condition (21) imposed on function $v(x)$ plus property of $s(k)$ implies the desired convergence of the iterative process (4). ∎

The Theorem 2 shows that the iterative process driven by LRS strategy is convergent in the mean x^* under specific condition, which can be easily realized by most numerical P. We assume that content popularity distribution in time t is nonuniform $p_1\ge p_2\ge\cdots p_N$ and take the following softmax approximation:

$$x_i^*\approx\frac{e^{p_i\kappa}}{\sum_{j=1}^{N}e^{p_j\kappa}}\approx\begin{cases}1/M, 1\le i\le M\\ 0, M<i\le N\end{cases}\tag{22}$$

The LRS is designed under the assumption that $p_1=p_2=\cdots=p_N$ and the caching distribution x^* will also tend to be uniform. However, when the content popularity distribution is skewed, it can be found x^* is approximate to prefetching the most popular contents. Previous work [9] [10] have pointed out that caching the most popular contents in codec scenario will produce much higher traffic volume, because it only prefetches a few contents and diminishes the multicast opportunity in the deliver phase. Therefore, the LRS strategy is inappropriate in the nonuniform codec scenario.

V. Prediction-based Dynamic Coded Caching

In this section, we first construct a nonlinear optimization model to determine the optimal caching distribution under nonuniform demands. Then, we fully exploit the user navigation pattern to design a high-order Markov prediction method.

A. Optimal caching distribution

We develop an optimization model to find the optimal caching distribution, with the influence of all important factors P, Q and specific codec scheme. Remark that we omit the parameter t.

$$\textbf{OPT:}\quad \text{Min}\quad R(P,Q)=\sum_{\vec{s}\in S}\mathbb{P}(\vec{s},P)\cdot R_s(\vec{s},Q)\tag{23}$$

$$\text{s.t.}\quad \sum_{i=1}^{N}q_i=1, 0\le q_i\le\frac{1}{M},\tag{24}$$

The objective function (23) characterizes the average traffic volume under the P, Q and specific codec scheme, where (24) is to avoid violating caching capacity constraints and caching redundant segments for each content. $\mathbb{P}(\vec{s},P)$ is the probability of request situation $\vec{s}$ under the general content popularity distribution P and can be calculated as:

$$\begin{aligned}\mathbb{P}(\vec{s},P)&=C_K^{\alpha_1}\cdot C_{K-\alpha_1}^{\alpha_2}\cdot\cdots C_{\alpha_N}^{\alpha_N}\cdot\prod_{i=1}^{N}p_i^{\alpha_i}\\ &=\frac{K!}{\alpha_1!\cdot\alpha_2!\cdot\cdots\alpha_N!}\cdot\prod_{i=1}^{N}p_i^{\alpha_i}.\end{aligned}$$

$R_s(\vec{s}, Q)$ denotes the traffic volume incurred by the situation $\vec{s}$, which is a function of caching distribution Q, request situation $\vec{s}$ and the specific codec scheme . Here we modify the placement phase in Algorithm 1 into the following nonuniform case: user k randomly prefetches $q_i MF$ bits of content n. Note that the elements of $\oplus$ operation of Algorithm 1 are assumed to be zero padded to the length of the longest element.

Lemma 3. *Given the request vector $[d_1, d_2, \ldots, d_K]$ and the codec scheme the incurred traffic volume is*

$$\sum_{i=1}^{K} \sum_{v \subset \{1,2,\ldots,K\}, |v|=i} \max_{j \in v} \{(q_{d_j} M)^{i-1} (1 - q_{d_j} M)^{K-i+1}\} \tag{25}$$

Proof: For a particular bit in any content i, the probability for the bit to be prefetched by any given user is $p = C^1_{q_i MF}/C^1_F = q_i M$. For any t-sized subset of K users, the probability that the bit is prefetched by those t users is $(q_i M)^t (1 - q_i M)^{K-t}$. Hence, the average number of bits of content i that are cached at those t users is $F(q_i M)^t (1 - q_i M)^{K-t}$. Suppose that $|U/\{k\}| = k - 1$, the expected size of $U_{k,U/\{k\}}$ is

$$F(q_i M)^{k-1} (1 - q_i M)^{K-k+1} \pm o(F)$$

with high probability. The term $o(F)$ is a constant in the order of less than F, which is ignored in the following derivation for a clearer presentation. Thus, the size of $\oplus_{k \in U} V_{k,U/\{k\}}$ is

$$\max_{i \in U} \left\{ F(q_i M)^{k-1} (1 - q_i M)^{K-k+1} \right\}$$

Traversing all possible k and user subsets U, we can get the formula (4). ∎

Lemma 4. *Any request vector $[d_1, d_2, \ldots, d_K]$ satisfying $\vec{s} = [\alpha_1, \alpha_2, \ldots, \alpha_N]$ incurs the same traffic volume as shown in (25).*

The Lemma shows that the traffic volume is only related to the number of requests for each content and independent of who request the contents. The proof can be found in Appendix A. With Lemma 1 and Lemma 2, we can present the codec traffic volume.

Theorem 3. *With the given $\vec{s}$, Q and codec scheme, the traffic volume incurred is*

$$R_s(\vec{s}, Q) = \sum_{i=1}^{K} \sum_{\substack{v \subset [K] \\ |v|=i}} \max_{j \in v} \{(q_{d_j} M)^{i-1} (1 - q_{d_j} M)^{K-i+1}\}, \tag{26}$$

where $[d_1, d_2, \ldots, d_K]$ is any requested vector that satisfies situation $\vec{s}$.

Then the optimal caching distribution Q^* is given by

$$Q^* = \arg\min_{Q} R(P, Q) \tag{27}$$

The problem OPT falls into the category of nonlinear programming. With the standardized methods such as Rosen's gradient projection method [20], the optimal solution Q^* can be obtained.

B. High-order markov prediction

The statistical and theoretical analysis of former sections present three concepts: 1) traditional cache update strategy LRS is inefficient in nonuniform codec scenario; 2) the user navigates in the content pool in a specific associate rule and make a case for predicting the user's future requests; 3) it is possible to determine the optimal caching distribution under given content popularity distribution. These concepts driven a natural intuition is that periodically operating users' local cache based on a new caching distribution, which is determined under the predictive content popularity and the **OPT** model.

Then we are required to predict the content future popularity distribution. Based on the previous investigation on user navigation model, user always requests the content related to the previously requested content. Here we take a high-order Markov prediction model to describe the user navigation pattern and make the prediction.

We define user i as a system with regarding a sequence of user requests $\langle d_i(0), d_i(1), \ldots, d_i(t) \rangle$ as a series of random variables, in such a way that the state at one time epoch depends on m states in the previous time. We consider a stochastic process $\{X_t, t = 0, 1, 2, \ldots\}$ that takes on a finite set $V = \{V_1, V_2, \ldots, V_N\}$, which is denoted by all contents. Then the conditional probability that next access for user i will be $d_i(t+1)$ is $\mathbb{P}(d_i(t+1) \big| d_i(t), \ldots, d_i(t-m+1))$.

Definition 3. *A $m-$order Markov predictor is defined to be a scheme for the calculation of conditional probabilities*

$$P^i_{d_i(t+1),\ldots,d_i(t-m+1)} = \mathbb{P}(d_i(t+1) \big| d_i(t), \ldots, d_i(t-m+1))$$

based on user i's requested sequence.

Note that there are K different Markov predictors for all users. In Web prefetching, the DG algorithm uses the *first order* Markov predictors, which calculates the probability $\mathbb{P}(d_i(t+1) \big| d_i(t))$ for user i, while the $m-$order PPM algorithm uses $1, 2, \ldots, k-$order Markov predictors and calculates conditional probabilities of the form $\mathbb{P}(d_i(t+1) \big| d_i(t)), \mathbb{P}(d_i(t+1) \big| d_i(t), d_i(t-1)), \ldots$ and $\mathbb{P}(d_i(t+1) \big| d_i(t), d_i(t-1), d_i(t-2))$ for user i. These techniques focus on predicting future request of each user and update each user's cache respectively. However, in our codec scenario, the users' local cache is operated in an uniform manner and it seems that this user-oriented Markov model is only suited to predict specific user's future request. Fortunately, we can still use the Markov model to predict the content popularity based on the following lemmas.

Lemma 5. *The moment estimation of popularity $p_j(t+1)$ is $\sum_{i=1}^{K} P^i_{j,S_i(t)}/K$, where $S_i(t) = \langle d_i(t-1), d_i(t-2), \ldots, d_i(t-m+1) \rangle$ is the $m-$order previous sequence of user i.*

Proof: Consider a random variable $X_j(t+1), 1 \le j \le N$ to represents the number of content j being requested in time $t+1$ and $N \cdot K$ random variables $Y^j_{i,S_i(t)}(t+1) = \{0,1\}, 1 \le j \le N, 1 \le i \le K$ with $Y^j_{i,S_i(t)}(t+1) = 1$ implying that user i requesting content j in time $t+1$. Then we can derive the moment estimation of popularity $p_j(t+1)$,

$$\hat{p}_j(t+1) = \mathbb{E}[X_j(t+1)]/K = \mathbb{E}\left[\sum_{i=1}^{K} Y^j_{i,S_i(t)}(t+1)\right]/K$$
$$= \sum_{i=1}^{K} \mathbb{E}\left[Y^j_{i,S_i(t)}(t+1)\right]/K = \sum_{i=1}^{K} P^i_{j,S_i(t)}/K.$$

■

The lemma 5 shows that when user i's previous request sequence is $S_i(t)$, calculating the conditional probability that user i's next request is content j and taking the average probability over all users as the predicting popularity of content j's in time $t+1$. In practice, the conditional probability $\mathbb{P}\left(d_i(t+1) = j \middle| S_i(t)\right)$ is calculated by

$$\frac{fr(j, S_i(t))}{fr(S_i(t))} \tag{28}$$

where $fr(S)$ denotes the frequency of sequence S.

It is obviously that $m-$order Markov predictor has $(N-1)N^m$ parameters (the transition probabilities), which is hard to deploy since the number of parameters increases exponentially with respect to the order of the model. For the purpose of space-time-saving, we introduce a novel method to calculate the conditional probability. In [26], Raftery proposed a description method of high-order Markov chain model which contains only one additional parameter for each extra lag. The method for user i can be defined as follows:

$$\mathbb{P}(d_i(t+1) \big| d_i(t), \ldots, d_i(t-m+1)) = \sum_{l=1}^{m} \lambda_i Q^i_{d_i(t+1)d_i(t+1-l)} \tag{29}$$

where

$$\sum_{j=1}^{m} \lambda^i_j = 1 \tag{30}$$

and $Q = [q_{j_1 j_2}]$ is a transition matrix such that

$$\sum_{j_2=1}^{N} Q^i_{j_1 j_2} = 1, 1 \le j_1 \le N \tag{31}$$

$$0 \le \sum_{i=1}^{m} \lambda_i Q^i_{j_1 j_2} \le 1, j_1, j_2 \in V \tag{32}$$

The constraint in (30), (31) and (32)are to guarantee the non-negativity and standardability of the conditional probability distribution. The number of parameters in this method is reduced to $(m+N^2)$. Moreover, the parameters $Q^i_{j_1 j_2}$ and λ^i_j can be estimated numerically by the following maximizing the log-likelihood of (29),

$$L(Q^i, \lambda^i) = \sum_{S_i(t) \in V^m} fr\left[S_i(t)\right] \cdot \log\left(\sum_{l=1}^{m} \lambda_i Q^i_{d_i(t+1)d_i(t+1-l)}\right) \tag{33}$$

where V^m is request sequence set with length m, subject to (30), (31) and (32).

It is noted this method requires solving a highly non-linear optimization problem. Since the content popularity evolves slowly, we can extend the concept of the time t, in which sense there is a time period T^t, server solves (33) for each user in the initiation of T^t and take the results into the quick prediction in time $t \in T^t$.

VI. Trace-Based Simulation

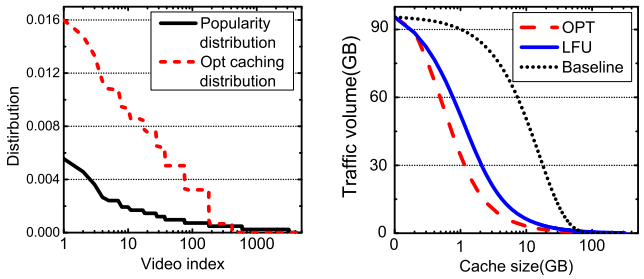
In this section, evaluation of our dynamic codec scheme is presented. We target on a circular cell with no inter-cell interference from other cells, which has a radius of $600m$ and users are randomly and independently distributed in the cell. The users only request for Youtube like content. We assume all the requests are of size 100MB, which is reasonable by assuming a screen size 1280×720 and 150 seconds playing time. The simulations and conclusions are also hold for larger file sizes. We uses a trace of 243,023 requests in Umass campus network from 2008/3/11 to 2008/3/17 [30], which consists of the unique user IP address, video's ID and the time stamp.

A. Optimal caching distribution

We choose the requests of two hours on 2008/3/11 to calculate each content's popularity and determine the corresponding optimal caching distribution via the **OPT** model. There are $N = 4143$ videos and $K = 954$ users. We compare the performance of **OPT** model with the well-known Least Recently Used (LRU) caching scheme and the baseline scheme in [9].

LFU: In such a scheme, each user prefetches the M most popular contents in its local cache.

Baseline Scheme: This scheme divides the contents set into groups and operates codec scheme in each sub-group, separately. It optimizes the cache allocation for each group.



(a) P and Q^* (b) Traffic versus cache size

Fig. 4: Basic results under requests during 2 hours on 2008/3/11.

As shown in Fig. 4(a), the video popularity distribution has a along tail and follows a Zipf law. The most popular video has a popularity of 0.5% and only 10% videos have popularity larger than 0.2%, while the popularity of most videos approximates to 0. The corresponding optimal caching distribution is much more skewed and only the top 400 videos are require to be prefetched.

Further, we compare the performance of our optimal caching distribution with the LFU and baseline scheme via requests of these two hours. Fig. 4(b) shows the traffic volumes versus the cache size under different schemes. It can be found that our **OPT** model produce the lowest traffic volume and the baseline scheme is even worse than the LFU strategy.

B. Popularity prediction

We use the trace date from 2008/3/11 to 2008/3/17. We define $\Delta = 3600s$ as the sampling interval of the popularity shifty and form the data 1 of video popularity versus time. In general, we generate a series of popularity evolutive curve to form data 2 based on the exponential decay model [28] [29]:

$$p_i(t) = p_i(0)\left[1 - \left(1 + \frac{\left(\eta^{\frac{1}{\nu_i}} - 1\right)(t-\theta_i)}{\tau_i}\right)^{-\nu_i}\right] + \xi_i(t) \tag{34}$$

where θ_i denotes the release time of video i following a uniform distribution, τ_i is interpreted as the video i's lifetime following a exponential distribution, ν_i is a form-determining parameter following a gamma distribution and $\xi_i(t)$ denotes the noise item following a normal distribution. Note there is a normalization procedure of popularity in each time t. We compare the performance of three strategies:

Correlation prediction: Szabo et. al [27] observes that there exists strong correlation between early and later time in the popularity of Youtube video and derives an exponential prediction model. Here we point out this model is categorized into the grey forecasting model GM(1,1) and adopt GM(1,1) by choosing 10% data as training set. The GM(1,1)is constructed by establishing a first order differential equation for accumulated sequence of $p_i(t)$ as $dp_i(t)/dt + ap_i(t) = b$, that is

$$p_i(t) = (p_i(t_0) - \hat{b}/\hat{a})(1 - e^{\hat{a}})e^{-\hat{a}(k-1)}, \tag{35}$$

where parameters a, b is measured by using least square method. If taking a logarithmic operation of (35) and take two time point t_e and t_l to represents the early and later time, we can get $\ln p_i(t_l) = \ln p_i(t_e) + \sum_{i=t_e}^{t_l} \hat{b}$, which is indeed the correlation model in [27].

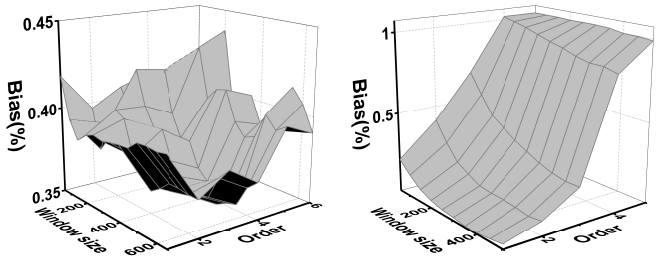
Fit prediction: We take a polynomial fit to predict popularity, where the order is fixed to 10 without generality.

Markov prediction: We consider two kinds of Markov prediction model: original Markov model and adding $W-$window Markov model. Since each video has a life span and some of them will not be required after a certain time, it is inappropriate to take the entire log file to construct Markov predictor. We define a window of size W, among which the data is considered to construct the transition probability matrix.

TABLE II: Main n

Bias(%)	Grey	Fit	1-M	2-M	3-M	4-M	5-M
data 1	2.182	3.247	1.159	0.809	0.711	0.701	2.015
data 2	3.142	8.095	2.955	2.998	3.528	4.110	5.001

Table. II presents the bias between predictive and practical popularity under different strategies. Note that Grey denotes the correlation model and $k-$M denotes the $k-$order Markov prediction. Under the data 1, the Markov predictor of all orders is much better than the correlation model and fit model, and $4-$order Markov makes highest predictive accuracy. Instead, under the data 2, the correlation model is better than the Markov predictor from order 3 to 5, and fit model also shows worst performance. There are two reasons: 1) the request of data 2 is generated randomly in accord with the specific popularity and contains weak associate rule, while the data 1 is a real trace and contains strong user navigation pattern, which can be fully utilized; 2) the popularity 1 has a concrete form of (34) and exhibits smooth trend that can be well fitted by the correlation model, while the popularity of data 2 shows strong fluctuations that is difficult to captured by a concrete kinetic equation.



(a) Data 1. (b) Data 2.

Fig. 5: The prediction bias versus the window size and the order.

Moreover, we investigate the adding window Markov predictor and simply explore the best order and window size of the Markov predictor under different data. As shown in Fig. 5, we plot the prediction bias versus the window size and the order. We can found that there exists a valley point in the bias surface. It shows that the order 4 and the window size 400 is the best under the data 1, the order 1 and the window size 400 is best under the data 2. The reason behind this trend is that the data 1 has at most order associate rule and average active life span of each video, while data 2 does not contain such high order relation.

C. Long-term traffic volume

We also use the trace data from 2008/3/11 to 2008/3/17 and compare the long-term traffic[5] yielded by following three systems:

Baseline system: It unicasts the requested video to each user and update each user's local cache based on the LRU rule, respectively.

LRS system: It multicasts the coded signal to users and update users' local cache based on the LRS rule.

OPT-Markov system: It multicasts the coded signal to users and update the user's local cache based on the Algorithm 2. In particular, we consider two OPT-Markov systems: one is the simple $1-$order Markov model, the other is the $4-$order

[5]The traffic in our simulation consists of two part: traffic in delivery phase and traffic in update phase.

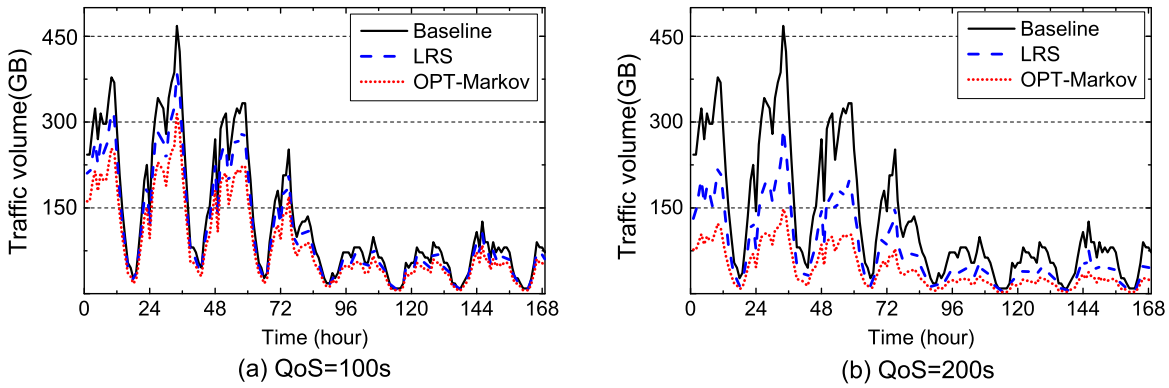


Fig. 6: Traffic from 2008/3/11 to 2008/3/17.

Markov model with window size 400, which is the best setting for this trace data.

The codec scheme requires that server first collects every users' requests and then transmits coded data according to their requests. In practice, the arrival of each user's request is asynchronous and waiting for all requests may lead to large waiting delay. While the baseline system takes an unicast delivery that can respond each user's request respectively, thus reduce the waiting delay. For comparison, we define the Quality of Service (QoS) as a barrier. Each system should serve every user not exceeding a given QoS (in seconds). Namely, the codec system, LRS and OPT-Markov system, cannot wait for all users' requests and should serve them in group. Here we consider two QoS levels: 100s and 200s.

Fig. 6 shows the traffic of each hour under different QoS levels. It can be found that OPT-Markov is strictly better than the baseline system and LRS system. When QoS=100s, the traffic produced by codec system is similar to the Baseline system, when QoS increase to 200s, the traffic produced by codec system is reduced and much better than the baseline system, since there are more users being satisfied simultaneously with higher QoS.

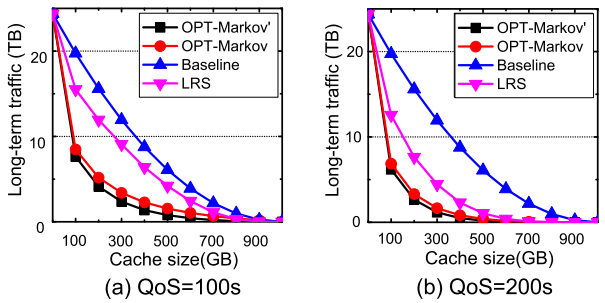


Fig. 7: Long-term traffic versus the cache size.

Fig. 7 shows the long-term traffic versus the cache size. Note that OPT-Markov and OPT-Markov' refers to $1-$order Markvo model and $4-$order Markov model with window of size 400, respectively. There is a common conclusion that the traffic will decrease when the size of local cache increases. The OPT-Markov with window size 400 and order 4 yields lowest traffic and original OPT-Markov is extremely close to it, which means that, although the OPT-Markov with window size 400 and order 4 predicts the popularity mostly accurately, we can take a simple $1-$order Markov model to make the prediction. The main reason is that the difference of the predictive bias between these two models is very little that cannot lead to much change in the output (optimal caching distribution), thus producing the similar traffic. Besides that, this traffic-cache tradeoff of OPT-Markov and LRS is more skewed under high QoS level, since there are more users can be satisfied simultaneously under higher QoS level such that increasing the coded multicasting opportunity, thus enhancing the trend when cache size increases.

VII. Conclusion and Future Work

In this paper, we introduced a practical coded caching scenario to efficiently handle the data traffic explosive over the wireless network. The key idea is to use the decentralized codec scheme with nonuniform demands and highly efficient cache updating procedure, i.e., the users prefetch the segments of contents according to the caching distribution in low-traffic time, the base station (server) multicasts the coded data upon requests during peak-traffic time and then predicts the future content popularity to determine a new optimal caching distribution, based on which each user updates their local cache. Our experimental evaluation uses a real trace of Youtube requests and demonstrates the traffic volume reduce on the order of at reasonable Qos levels. With storage capacity becoming exceptionally cheap, our conclusion is this cache-exchanging-spectrum technique seems to be a more advanced and powerful way to alleviate the bottlenecks in wireless network. Our future work is concentrated in two aspects:

The dynamic codec scheme in heterogeneous network: Since the wireless access points (AP) such as Wifi is very popular in our daily life, it is possible to extend the codec architecture, i.e., deploying the cache into both APs and users, the users could communicate to both APs and base station (server) and designating the multi-level codec scheme in such heterogenous network.

Nonlinear codec scheme: We adopt the linear coding in our dynamic codec scheme and it is shown that nonlinear coding is much more efficient [7]. Thus it is possible to design the nonlinear codec scheme in such scenario.

REFERENCES

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update 2012-2017," *Whiter paper*, 2013.

[2] V. Chandrasekhar, J. G. Andrews, and A. Gatherer, "Femtocell networks: a survey," *IEEE Communications Magzine*, vol. 46, no. 9, pp. 59–67, 2008.

[3] N. Golrezaei, A. F. Molisch, A. G. Dimakis, G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp.142–149, 2013

[4] J. Llorca, A. M. Tulino, K. Guan, J. Esteban, M. Varvello, N. Choi, and D. Kilper, "Network-coded caching-aided multicast for efficient content delivery," in *Proc. IEEE ICC*, 2013, pp.3557–3562.

[5] J. Llorca and A. M. Tulino, "The content distribution problem and its complexity classification," *Alcatel-Lucent technical report*, 2013.

[6] S. Gitzenis, G. S. Paschos, L. Tassiulas, "Asymptotic laws for joint content replication and delivery in wireless networks," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 2760–2776, 2013

[7] M. A. Maddah-Ali, U. Niesen, "Fundamental limits of caching," in *Proc. IEEE ISIT*, 2013, pp.1077–1081.

[8] M. A. Maddah-Ali, Niesen U, "Decentralized coded caching attains order-optimal memory-rate tradeoff," Jan. 24, 2013, Online: http://arxiv.org/abs/1301.5848.

[9] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands,"Aug. 1, 2013, Online: http://arxiv.org/abs/1308.0178.

[10] M. Ji, A. M. Tulino, J. Llorca, G. Caire, "Order optimal coded caching-aided multicast under Zipf demand distributions," Feb. 19, 2014, Online: http://arxiv.org/abs/1402.4576.

[11] A. Nanopoulos, D. Katsaros and Y. Manolopoulos, "A data mining algorithm for generalized web prefetching," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no.5, pp. 1155–1169, 2003.

[12] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no.2, pp. 146–160, 1972.

[13] J. G. Cleary and I. H. Witten, "Using Adaptive Coding and Partial String Matching," *IEEE Transactions on Communication*, vol. 32, no.4, pp. 396–402, 1984.

[14] V. Padmanabhan and J. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *ACM SIGCOMM Computer Communication Review*, vol. 26, no.3, pp. 22–36, 1996.

[15] K. M. Curewitz, P. Krishnan, and J. S. Vitter, "Practical Prefetching via Data Compression," in *Proc. ACM SIGMOD*, 1993, pp. 257–266.

[16] L. Fan, P. Cao, W. Lin and Q. Jacobson, "Web Prefetching between Low-Bandwidth Clients and Proxies: Potential and Performance," in *Proc. ACM SIGMETRICS*, 1999, pp. 178–187.

[17] EXPWAY, Online: http://www.expway.com/.

[18] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *ACM Communications*, vol. 28, no.2, pp. 202–208, 1985.

[19] M. Chrobak and J. Noga, "LRU is better than FIFO," in *Proc. ACM SODA*, 1999, pp. 78–81.

[20] J. B. Rosen, "The gradient projection method for nonlinear programming. Part I. Linear constraints," *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no.1, pp. 181–217, 1960.

[21] R. Pedarsani, M. A. Maddah-Ali and U. Niesen "Online coded caching,"Nov. 14, 2013, Online: http://arxiv.org/abs/1311.3646.

[22] G. S. Ladde and D. D. Siljak, "Convergence and stability of distributed stochastic iterative processes," *IEEE Transactions on Automatic Control*, vol. 35, no. 6, pp. 665–672, 1990.

[23] C. Fricker, P. Robert and J. A. Roberts, "versatile and accurate approximation for LRU cache performance," in *Proc. International Teletraffic Congress*, 2012, pp. 8–15.

[24] G. S. Ladde and M. Sambandham, "andom difference inequalities," *North-Holland Mathematics Studies*, vol. 110, pp. 231–240, 1985.

[25] W. Hanhn, *Stability of motion*. Springer, 1967

[26] A. E. Raftery "A model for high-order Markov chains," *Journal of the Royal Statistical Society*, vol. 47, no. 3, pp. 528–539, 1985.

[27] G. Szabo and B. A. Huberman, "redicting the popularity of online content," *ACM Communications*, vol. 53, no. 8, pp. 80–88, 2010.

[28] M. Cha, H. Kwak, P. Rodriguez, Y-Y. Ahn and S. Moon, "I tube, You Tube, Everybody Tubes: Analyzing the Worlds Largest UserGenerated Content Video System," in *Proc. ACM SIGCOMM conference on Internet measurement*, 2007, pp. 1–14.

[29] X. Cheng, J. Liu and C. Dale, "Understanding the characteristics of internet short video sharing: A YouTube-based measurement study," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1184–1194, 2013.

[30] http://traces.cs.umass.edu/index.php/Network/Network.

APPENDIX

A. Proof of Lemma 4

Proof: We consider two requested vectors:

$$(d_1, \ldots, d_m, \ldots, d_n, \ldots, d_K),$$

$$(d_1, , \ldots, d_n^* = d_m, \ldots, d_m^* = d_n, \ldots, d_K).$$

Remark that these two requested vectors satisfying request situation: $\vec{s} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ and the difference between them is that user m and user n exchanges their requests. Factually, the different request vectors satisfying the same request situation can be converted to each other via finite exchange. Then, we will show that, under these two requested vectors, the traffic rate is equal.

The equation () refers to the traffic rate under $(d_1, \ldots, d_m, \ldots, d_n, \ldots, d_K)$ and the equation () refers to the traffic rate under $(d_1, \ldots, d_m^*, \ldots, d_n^*, \ldots, d_K)$

Consider $d_m^* = d_n$ and $d_n^* = d_m$, we can get

$$R_m(\vec{s}, Q) = R_n^*(\vec{s}, Q), R_n(\vec{s}, Q) = R_m^*(\vec{s}, Q),$$
$$R_{m,n}^*(\vec{s}, Q) = R_{n,m}(\vec{s}, Q) = R_{m,n}^*(\vec{s}, Q), R_\emptyset(\vec{s}, Q) = R_\emptyset^*(\vec{s}, Q).$$

Hence, $R_s(\vec{s}, Q) = R_s^*(\vec{s}, Q)$. ■

B. Proof of Lemma 1

Proof: The case $a_{ii} = 0$ implies the content i is the replaced one when the new requested content is j that $j \neq i$. According to the whole probability formula, the probability of above case is $\mathbb{P}\{a_{ii} = -1\} = \sum_{j \neq i} \mathbb{P}(H_i|j) \cdot p_j = \sum_{j \neq i} p_j \cdot \mathbb{P}(H_i)/(1 - \mathbb{P}(H_j))$, where $H_i|j$ represents that content i is replaced and content j is the new request. Then $\mathbb{P}(a_{ii} = 0) = 1 - \mathbb{P}(a_{ii} = -1)$. The case $a_{ij} = 0$ denotes the case that content i is new requested one and content j is the replaced one. Thus the probability can be calculated as $\mathbb{P}\{a_{ij} = 0\} = p_i \cdot \mathbb{P}(H_j|i) = p_i \cdot \mathbb{P}(H_j)/(1 - \mathbb{P}(H_i))$ and $\mathbb{P}\{a_{ij} = -1\} = 1 - \mathbb{P}\{a_{ij} = 0\}$.

Previous work [23] delves the cache hit ratio for LRU strategy. It shows that, under the content popularity distribution P, the cache hit ratio for content i is $P_{C_i} = 1 - e^{-p_i \kappa}$, where κ is the solution of equation $\sum_{i=1}^{n} e^{-p_i \kappa} = N - M$. We assume the time is long enough that cache could traverse all contents and the number of replacement for content i is equal to the number of entrance into cache for content i, which is $Lp_i(1 - P_{C_i})$ under the total number of requests is L. Thus

$$\mathbb{P}(H_i) = p_i(1 - P_{C_i}) = p_i e^{-p_i \kappa}. \tag{38}$$

■

$$R_s(\vec{s},Q,\Gamma)=\sum_{i=1}^{K}R_m(\vec{s},Q,\Gamma)+R_n(\vec{s},Q,\Gamma)+R_{m,n}(\vec{s},Q,\Gamma)+R_{\emptyset}(\vec{s},Q,\Gamma). \tag{36}$$

Where

$$\begin{aligned}
R_m(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i,n\in v,m\notin v}\max\{(q_{d_m}M)^{i-1}(1-q_{d_m}M)^{K-i+1},\max_{j\in v/m}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}\},\\
R_n(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i,n\notin v,m\in v}\max\{(q_{d_n}M)^{i-1}(1-q_{d_n}M)^{K-i+1},\max_{j\in v/n}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}\},\\
R_{m,n}(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i,n\notin v,m\in v}\max\left\{\begin{array}{c}(q_{d_m}M)^{i-1}(1-q_{d_m}M)^{K-i+1},(q_{d_n}M)^{i-1}(1-q_{d_n}M)^{K-i+1},\\ \max\limits_{j\in v/n}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}\end{array}\right\},\\
R_{\emptyset}(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i}\max_{j\in v/\{m,n\}}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}.
\end{aligned}$$

$$R_s^*(\vec{s},Q,\Gamma)=\sum_{i=1}^{K}R_m^*(\vec{s},Q,\Gamma)+R_n^*(\vec{s},Q,\Gamma)+R_{m,n}^*(\vec{s},Q,\Gamma)+R_{\emptyset}^*(\vec{s},Q,\Gamma). \tag{37}$$

Where

$$\begin{aligned}
R_m^*(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i,n\in v,m\notin v}\max\{(q_{d_m}M)^{i-1}(1-q_{d_m}M)^{K-i+1},\max_{j\in v/m}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}\},\\
R_n^*(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i,n\notin v,m\in v}\max\{(q_{d_n}M)^{i-1}(1-q_{d_n}M)^{K-i+1},\max_{j\in v/n}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}\},\\
R_{m,n}^*(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i,n\notin v,m\in v}\max\left\{\begin{array}{c}(q_{d_m}M)^{i-1}(1-q_{d_m}M)^{K-i+1},(q_{d_n}M)^{i-1}(1-q_{d_n}M)^{K-i+1},\\ \max\limits_{j\in v/n}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}\end{array}\right\},\\
R_{\emptyset}^*(\vec{s},Q,\Gamma)&=\sum_{v\subset[K],|v|=i}\max_{j\in v/\{m,n\}}\{(q_{d_j}M)^{i-1}(1-q_{d_j}M)^{K-i+1}\}.
\end{aligned}$$

C. Proof of Theorem 1

Proof: The LADDE et al. has proved the comparison principle in the time-variant scenario [22]. For clarity, we show a same procedure in our time-invariant scenario.

Let $x(k)$ be any solution of (4) and we denote

$$I(\delta)=v\left\{x(k)+\delta[x(k+1)-x(k)]\right\}, \tag{39}$$

with $\delta\in[0,1]$, and compute

$$I'(\delta)=v_x\left\{x(k)+\delta[x(k+1)-x(k)]\right\}[x(k+1)-x(k)]. \tag{40}$$

By integrating (40) between 0 and 1, and adopting the expectation with respect to $\mathfrak{F}_c$, we get

From condition $\mathfrak{L}v(x)\leq z[v(k,x)]$ and (13), (40) reduces to

$$\mathbb{E}\left\{v[x(k+1)]-v[x(k)]\big|\mathfrak{F}_c\right\}\leq z[v[x(k)]] \tag{41}$$

Taking the expectation of (41) and using the concavity of z, as well as the existence of $\mathbb{E}\{v[x(k)]\}$, we arrive at

$$\mathbb{E}\left\{v[x(k+1)]-v[x(k)]\big|\mathfrak{F}_c\right\}\leq z[\mathbb{E}\{v[x(k)]\}] \tag{42}$$

Based on (42), we incorporate the comparison equation (11) and choose $b_0\geq\mathbb{E}[v(x_0)]$. Finally, we can get the solution inequality (12) via applying the basic results concerning difference inequalities [24]. ■