Low Subpacketization Schemes for Coded Caching

Li Tang and Aditya Ramamoorthy
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50010
Emails:{litang, adityar}@iastate.edu

***Abstract*—Coded caching is a technique that generalizes conventional caching and promises significant reductions in traffic over caching networks. However, the basic coded caching scheme requires that each file hosted in the server be partitioned into a large number (called the subpacketization level) of non-overlapping subfiles. From a practical perspective, this is problematic as it means that prior schemes are only applicable when the size of the files is extremely large. In this work, we propose coded caching schemes based on combinatorial structures called resolvable designs. These structures can be obtained in a natural manner from linear block codes whose generator matrices possess certain rank properties. We demonstrate that several schemes with subpacketization levels that are exponentially smaller than the basic scheme can be obtained.**

I. Introduction

Caching is a popular technique for facilitating large scale content delivery over the Internet. Traditionally, caching operates by storing popular content closer to the end users. Typically, the cache serves an end user's file request partially (or sometimes entirely) with the remainder of the content coming from the main server. Prior work in this area [1] demonstrates that allowing coding in the cache and coded transmission from the server (referred to as *coded caching*) to the end users can allow for significant reductions in the number of bits transmitted from the server to the end users. In particular, [1] considers a scenario where a single server containing N files, each of size F subfiles (a subfile is a basic unit of storage) connects to K users over a shared link and each user has a cache memory of size MF subfiles. Coded caching consists of two distinct phases: a *placement phase* and a *delivery phase*. In the placement phase, the caches of the users are populated. This phase does not depend on the user demands which are assumed to be arbitrary. In the delivery phase, the server sends a set of *coded* signals that are broadcast to each user such that each user's demand is satisfied.

However, the huge gains of coded caching require each file to be partitioned into $F_s = \binom{K}{\frac{KM}{N}}$ non-overlapping subfiles of equal size; F_s is referred to as the *subpacketization level*. It can be observed that for a fixed cache fraction $\frac{M}{N}$, F_s grows exponentially with K. This can be problematic in practical implementations. For instance, suppose that $K = 50$, with $\frac{M}{N} = 0.4$ so that $F_s = \binom{50}{20} \approx 10^{14}$. In this case, it is evident that at the bare minimum, the size of each file has to be at least 100 terabits for leveraging the gains in [1]. It is even worse in practice. The atomic unit of storage on present day hard drives is a sector of size 512 bytes and the trend in the disk drive industry is to move this to 4096 bytes. As a result, the minimum size of each file needs to be much higher than 100 terabits. Therefore, the scheme in [1] is not practical even for moderate values of K. Furthermore, even for smaller values of K, schemes with low subpacketization levels are desirable. This is because any practical scheme will require each of the subfiles to have some header information that allows for decoding at the end users. When there are a large number of subfiles, the header overhead may be non-negligible.

This issue has also been considered in the work of [2]. They proposed a low subpacketization scheme based on placement delivery arrays. Reference [3] viewed the problem from a hypergraph perspective and presented classes of coded caching schemes. The subpacketization issue in the decentralized coded caching setting was considered in the work of [4].

In this work, we propose low subpacketization level schemes for coded caching. Our proposed scheme leverages the properties of combinatorial structures known as resolvable designs and their natural relationship with linear block codes. We show that the construction proposed in [2] (and our own prior work [5]) is a special case of the present work.

This paper is organized as follows. Section II presents the proposed scheme, Section III discusses classes of linear codes that are useful for our application, Section IV compares our work with other contributions in the literature and Section V concludes the paper with a discussion of future work. Owing to space limitations, most of the proofs have been omitted. These can be found in [6].

II. Proposed low subpacketization level scheme

In this work we use combinatorial designs [7] to specify the placement scheme in the coded caching system.

Definition 1. A design is a pair $(X, \mathcal{A})$ such that

1) X is a set of elements called points, and
2) $\mathcal{A}$ is a collection of nonempty subsets of X called blocks, where each block contains the same number of points.

A parallel class $\mathcal{P}$ in a design $(X, \mathcal{A})$ is a subset of disjoint blocks from $\mathcal{A}$ whose union is X. A partition of $\mathcal{A}$ into several parallel classes is called a resolution, and $(X, \mathcal{A})$ is said to be a resolvable design if $\mathcal{A}$ has at least one resolution.

This work was supported in part by the National Science Foundation (NSF) by grants CCF-1320416, CCF-1149860 and DMS-1120597.

The incidence matrix $\mathcal{N}$ of a design $(X, \mathcal{A})$ is a 0-1 (binary) matrix of dimension $|X| \times |\mathcal{A}|$, where the rows and columns correspond to the points and blocks respectively. The (i, j)-th entry is 1 if point i belongs to block corresponding to column j and 0 otherwise.

Let $[n] = \{1, \ldots, n\}$. In coded caching, the placement scheme can be specified by a design. For example, let $X = [K]$ and $\mathcal{A} = \{B : B \subset [K], |B| = t\}$. In the scheme of [1], the users are associated with X and the subfiles with $\mathcal{A}$. Each file W_n is divided into subfiles $W_{n,B}$, $n = 1, \ldots, N$, $B \in \mathcal{A}$ and user $i \in [K]$ caches subfile $W_{n,B}$ if $i \in B$. In general, we can reverse the roles of the points and blocks and choose to associate the users with the blocks and subfiles with the points instead. Equivalently, the incidence matrix of the design (or its transpose) can be used to specify the placement scheme.

Our constructions stem from resolvable designs [7]. We begin by showing that any linear block code can be used to obtain a resolvable block design.

A. Resolvable Design Construction

Consider an (n, k) linear block code over $GF(q)$. We collect its q^k codewords and construct a matrix $\mathbf{T}$ of size $n \times q^k$ as follows.

$$\mathbf{T} = [\mathbf{c}_0^T, \mathbf{c}_1^T, \cdots, \mathbf{c}_{q^k-1}^T], \tag{1}$$

where the $1 \times n$ vector $\mathbf{c}_i$ represents the i-th codeword of the code. Let $X = \{0, 1, \cdots, q^k - 1\}$ be the point set and $\mathcal{A}$ be the collection of all subsets $B_{i,l}$ for $0 \le i \le n-1$ and $0 \le l \le q-1$, where

$$B_{i,l} = \{j : \mathbf{T}_{i,j} = l\}.$$

Using this construction, we can obtain the following result.

Lemma 1. The construction procedure above results in a design $(X, \mathcal{A})$ where $X = \{0, 1, \cdots, q^k - 1\}$ and $|B_{i,l}| = q^{k-1}$ for all $0 \le i \le n-1$ and $0 \le l \le q-1$. Furthermore, the design is resolvable with parallel classes given by $\mathcal{P}_i = \{B_{i,l} : 0 \le l \le q-1\}$, for $0 \le i \le n-1$.

Proof. Let $\mathbf{G} = [g_{ab}]$, for $0 \le a \le k-1$, $0 \le b \le n-1$ where $g_{ab} \in GF(q)$. Note that for $\Delta = [\Delta_0 \ \Delta_1 \ \ldots \ \Delta_{n-1}] = \mathbf{uG}$, we have

$$\Delta_b = \sum_{a=0}^{k-1} \mathbf{u}_a g_{ab},$$

where $\mathbf{u} = [\mathbf{u}_0, \cdots, \mathbf{u}_{k-1}]$. Let a^* be such that $g_{a^*b} \neq 0$. Consider the equation

$$\sum_{a \neq a^*} \mathbf{u}_a g_{ab} = \Delta_b - \mathbf{u}_{a^*} g_{a^*b},$$

where Δ_b is fixed. For arbitrary $\mathbf{u}_a$, $a \neq a^*$, this equation has a unique solution for $\mathbf{u}_{a^*}$, which implies for any Δ_b, $|B_{b,\Delta_b}| = q^{k-1}$ and that $\mathcal{P}_b$ forms a parallel class. □

Example 1. Consider a $(4, 2)$ linear block code over $GF(3)$ with generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}.$$

Collecting the nine codewords, $\mathbf{T}$ is constructed as follows.

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\ 0 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 0 \end{bmatrix}.$$

Using $\mathbf{T}$, we generate the resolvable block design $(X, \mathcal{A})$ where the point set is $X = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. For instance, block $B_{0,0}$ is obtained by identifying the column indexes of 0's in the first row of $\mathbf{T}$, i.e., $B_{0,0} = \{0, 1, 2\}$. Following this, we obtain

$$\begin{aligned}\mathcal{A} = &\{\{0,1,2\},\{3,4,5\},\{6,7,8\},\{0,3,6\},\{1,4,7\},\{2,5,8\},\\ &\{0,5,7\},\{1,3,8\},\{2,4,6\},\{0,4,8\},\{2,3,7\},\{1,5,6\}\}.\end{aligned}$$

It can be observed that $\mathcal{A}$ has a resolution (*cf.* Definition 1) with the following parallel classes.

$$\begin{aligned}\mathcal{P}_0 &= \{\{0,1,2\},\{3,4,5\},\{6,7,8\}\},\\ \mathcal{P}_1 &= \{\{0,3,6\},\{1,4,7\},\{2,5,8\}\},\\ \mathcal{P}_2 &= \{\{0,5,7\},\{1,3,8\},\{2,4,6\}\}, \text{ and}\\ \mathcal{P}_3 &= \{\{0,4,8\},\{2,3,7\},\{1,5,6\}\}.\end{aligned}$$

B. A special class of linear block codes

In this work, we consider a special class of linear block codes whose generator matrices satisfy specific rank properties. It turns out that resolvable designs obtained from these codes are especially suited for usage in coded caching.

Consider an (n, k) linear block code over $GF(q)$ and denote the generator matrix $\mathbf{G}$ as $\mathbf{G} = [g_{ij}]$, where $0 \le i \le k-1$, $0 \le j \le n-1$. Let $\mathbf{g}_j$ denote the j-th column of $\mathbf{G}$. Let z be the least positive integer such that $k+1$ divides nz (denoted by $k+1 \mid nz$). We let $(t)_n$ denote t mod n.

In our construction we will need to consider various collections of $k+1$ consecutive columns of $\mathbf{G}$ (wraparounds over the boundaries are allowed). For this purpose, let $\mathcal{T}_a = \{a(k+1), \cdots, a(k+1)+k\}$ and $\mathcal{S}_a = \{(t)_n \mid t \in \mathcal{T}_a\}$. Let $\mathbf{G}_{\mathcal{S}_a} = [\mathbf{g}_{i_0}, \cdots, \mathbf{g}_{i_k}]$ be a submatrix of $\mathbf{G}$ specified by the columns in $\mathcal{S}_a$, i.e., $\mathbf{g}_{i_j} \in \mathbf{G}_{\mathcal{S}_a}$ if $i_j \in \mathcal{S}_a$. Next, we define the $(k, k+1)$-consecutive column property that is central to the rest of the discussion.

Definition 2. *$(k, k+1)$-consecutive column property.* Consider the submatrices of $\mathbf{G}$ specified by $\mathbf{G}_{\mathcal{S}_a}$ for $0 \le a \le \frac{zn}{k+1} - 1$. We say that $\mathbf{G}$ satisfies the $(k, k+1)$-consecutive column property if all $k \times k$ submatrices of each $G_{\mathcal{S}_a}$ are full rank.

Henceforth, we abbreviate the $(k, k+1)$-consecutive column property as $(k, k+1)$-CCP or simply by CCP, if the value of k is clear from the context.

Example 2. In Example 1 we have $k = 2, n = 4$ and hence $z = 3$. Thus, $\mathcal{S}_0 = \{0, 1, 2\}, \mathcal{S}_1 = \{3, 0, 1\}, \mathcal{S}_2 = \{2, 3, 0\}$ and $\mathcal{S}_3 = \{1, 2, 3\}$. The corresponding generator matrix $\mathbf{G}$ satisfies the CCP as any two columns of the each of submatrices $\mathbf{G}_{\mathcal{S}_i}, i = 0, \ldots, 3$ are full-rank.

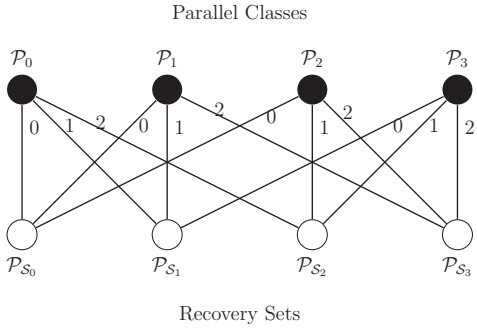

Fig. 1. Recovery set bipartite graph

C. Usage in a coded caching scenario

Our proposed placement scheme uses resolvable designs generated from linear block codes that satisfy the CCP. We associate the users with the blocks. Each subfile is associated with a point and an additional index as explained below. The delivery scheme leverages the structure imposed by the CCP.

Example 3. Consider the resolvable design from Example 1, where we recall that $z = 3$. The blocks in $\mathcal{A}$ correspond to twelve users U_{012}, U_{345}, U_{678}, U_{036}, U_{147}, U_{258}, U_{057}, U_{138}, U_{246}, U_{048}, U_{237}, U_{156}. Each file is partitioned into $F_s = 9 \times z = 27$ subfiles, each of which is denoted by $W^s_{n,t}$, $t = 0, \cdots, 8$ and $s = 0, 1, 2$. The cache in user U_{abc}, denoted Z_{abc} is specified as $Z_{abc} = \{W^s_{n,t} \mid t \in \{a, b, c\}, s \in \{0, 1, 2\}$ and $n \in [N]\}$. This corresponds to a coded caching system where each user caches $1/3$-rd of each file so that $M/N = 1/3$.

In general, we have $K = |\mathcal{A}| = nq$. Each file W_n, $n \in [N]$ is divided into $q^k z$ subfiles $W_n = \{W^s_{n,t} \mid 0 \leq t \leq q^k - 1, 0 \leq s \leq z - 1\}$. A subfile $W^s_{n,t}$ is cached in user U_B where $B \in \mathcal{A}$ if $t \in B$. Therefore, each user caches a total of $Nq^{k-1}z$ subfiles. This requires $Nq^{k-1}z \times \frac{F}{q^k z} = F\frac{N}{q} = FM$ subfiles of cache memory at each user so that $M/N = 1/q$.

It remains to show that we can design a delivery phase scheme that satisfies any possible demand pattern. Suppose that in the delivery phase user U_B requests file W_{d_B} where $d_B \in [N]$. The server responds by transmitting several equations that satisfy each user. Each equation allows $k + 1$ users from *different parallel classes* to simultaneously obtain a missing subfile. Our delivery scheme is such that the set of transmitted equations can be classified into various *recovery sets* that correspond to appropriate collections of parallel classes. It turns out that these recovery sets correspond precisely to the sets $\mathcal{S}_a, 0 \leq a \leq \frac{zn}{k+1} - 1$ defined earlier. We illustrate this by means of the example below.

Example 4. Consider the placement scheme specified in Example 3. Let each user U_B request file W_{d_B}. The recovery sets are specified by means of the recovery set bipartite graph shown in Fig. 1, e.g., $\mathcal{P}_{\mathcal{S}_1}$ corresponds to $\mathcal{S}_1 = \{0, 1, 3\}$. The outgoing edges from each parallel class are labeled arbitrarily with numbers $0, 1$ and 2. Our delivery scheme is such that each user recovers missing subfiles with a specific superscript from each recovery set that its corresponding parallel class participates in. For instance, a user in parallel class $\mathcal{P}_1$ recovers missing subfiles with superscript 0 from $\mathcal{P}_{\mathcal{S}_0}$, superscript 1 from $\mathcal{P}_{\mathcal{S}_1}$ and superscript 2 from $\mathcal{P}_{\mathcal{S}_3}$; these superscripts are the labels of outgoing edges from $\mathcal{P}_1$ in the bipartite graph.

It can be checked, e.g., that user U_{012} which lies in $\mathcal{P}_0$ recovers all missing subfiles with superscript 1 from the equations below.

$$W^1_{d_{012},3} \oplus W^1_{d_{036},2} \oplus W^0_{d_{237},0}, \quad W^1_{d_{012},6} \oplus W^1_{d_{036},1} \oplus W^0_{d_{156},0},$$
$$W^1_{d_{012},4} \oplus W^1_{d_{147},0} \oplus W^0_{d_{048},1}, \quad W^1_{d_{012},7} \oplus W^1_{d_{147},2} \oplus W^0_{d_{237},1},$$
$$W^1_{d_{012},8} \oplus W^1_{d_{258},0} \oplus W^0_{d_{048},2}, \quad W^1_{d_{012},5} \oplus W^1_{d_{258},1} \oplus W^0_{d_{156},2};$$

Each of the equations above benefits three users. They are generated simply by choosing U_{012} from $\mathcal{P}_0$, *any* block from $\mathcal{P}_1$ and the last block from $\mathcal{P}_3$ so that the *intersection of all these blocks is empty*. The fact that these equations are useful for the problem at hand is a consequence of the CCP.

The process of generating these equations can be applied to all possible recovery sets. It can be shown that this allows all users to be satisfied at the end of the procedure.

In what follows we first show that for the recovery set $\mathcal{P}_{\mathcal{S}_a}$ it is possible to generate equations that benefit $k + 1$ users simultaneously.

Claim 1. Consider the resolvable design $(X, \mathcal{A})$ constructed as described in Section III.A by an (n, k) linear block code that satisfies the CCP. Let $\mathcal{P}_{\mathcal{S}_a} = \{\mathcal{P}_i \mid i \in \mathcal{S}_a\}$ for $0 \leq a \leq \frac{zn}{k+1} - 1$, i.e., it is the subset of parallel classes corresponding to $\mathcal{S}_a$. We emphasize that $|\mathcal{P}_{\mathcal{S}_a}| = k + 1$. Consider blocks $B_{i_1,l_{i_1}}, \ldots, B_{i_k,l_{i_k}}$ (where $l_{i_j} \in \{0, \ldots, q-1\}$) that are picked from any k distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a}$. Then, $|\cap_{j=1}^k B_{i_j,l_{i_j}}| = 1$.

Proof. Following the construction in Section III.A, we note that a block $B_{i,l} \in \mathcal{P}_i$ is specified by

$$B_{i,l} = \{j : \mathbf{T}_{i,j} = l\}.$$

Let $\mathbf{G} = [g_{ab}]$, for $0 \leq a \leq k-1$, $0 \leq b \leq n-1$.

Now consider $B_{i_1,l_{i_1}}, \ldots, B_{i_k,l_{i_k}}$ (where $i_j \in \mathcal{S}_a, l_{i_j} \in \{0, \ldots, q-1\}$) that are picked from k distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a}$. W.l.o.g. we assume that $i_1 < i_2 < \cdots < i_k$. Let $\mathcal{I} = \{i_1, \ldots, i_k\}$ and $\mathbf{T}_{\mathcal{I}}$ denote the submatrix of $\mathbf{T}$ obtained by retaining the rows in $\mathcal{I}$. We will show that the vector $[l_{i_1}\ l_{i_2}\ \ldots\ l_{i_k}]^T$ is a column in $\mathbf{T}_{\mathcal{I}}$.

To see this consider the system of equations in variables $\mathbf{u}_0, \ldots, \mathbf{u}_{k-1}$.

$$\sum_{b=0}^{k-1} \mathbf{u}_b g_{bi_1} = l_{i_1},$$
$$\vdots$$
$$\sum_{b=0}^{k-1} \mathbf{u}_b g_{bi_k} = l_{i_k}.$$

By the CCP, the vectors $\mathbf{g}_{i_1}, \mathbf{g}_{i_2}, \ldots, \mathbf{g}_{i_k}$ are linearly independent. Therefore this system of k equations in k variables has a unique solution over $GF(q)$. The result follows. □

Algorithm 1: Signal Generation Algorithm

Input : Indices of recovery set $\mathcal{S}_a$, superscript function $E_a(B)$ for user U_B, signal set $Sig = \emptyset$.

while *any user* $U_B \in \mathcal{P}_j, j \in \mathcal{S}_a$ *does not recover all its missing subfiles with superscript* $E_a(B)$ **do**

  Pick users $U_{B_{i_1,l_{i_1}}}, \ldots, U_{B_{i_{k+1},l_{i_{k+1}}}}$ where $l_j \in \{0, \ldots, q-1\}, B_{j,l_j} \in \mathcal{P}_j$ and $j \in \mathcal{S}_a = \{i_1, \cdots, i_{k+1}\}$, such that $\cap_{j \in \mathcal{S}_a} B_{j,l_j} = \phi$;

  Let $\hat{l}_s = \cap_{j \in \mathcal{S}_a \setminus \{s\}} B_{j,l_j}$ for $s \in \mathcal{S}_a$;

  Add $\oplus_{s \in S_a} W^{E_a(B_{s,l_s})}_{d_{B_{s,l_s}}, \hat{l}_s}$ to Sig

end

Output: Signal set Sig.

Claim 1 implies that any k users from k different parallel classes of $\mathcal{P}_{\mathcal{S}_a}$ have a unique subfile in common. This can be used in turn to generate an equation that simultaneously benefits $k+1$ users.

The formal argument is made in Algorithm 1 which operates as follows. It takes as input a recovery set $\mathcal{P}_{\mathcal{S}_a} = \{\mathcal{P}_i \mid i \in \mathcal{S}_a\}$ and superscript function $E_a(B)$ for each user U_B in $\mathcal{P}_{\mathcal{S}_a}$, which indicates the superscript of subfiles of U_B that will be recovered. For any $k+1$ users U_B from $k+1$ distinct parallel classes, if their corresponding blocks have no point in common, they can generate a signal, each of which can recover one missing subfile of U_B with superscript $E_a(B)$.

Claim 2. For each user U_B belonging to a parallel class in $\mathcal{P}_{\mathcal{S}_a}$ with superscript function $E_a(B)$, the signals generated by Algorithm 1 can recover all the missing subfiles needed by U_B with superscript $E_a(B)$.

For each recovery set $\mathcal{P}_{\mathcal{S}_a}$, the superscript function $E_a(B)$ of a user U_B in parallel class $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a}$ is obtained by the bipartite recovery set graph (see Fig. 1 for an example).

The overall delivery scheme repeatedly applies Algorithm 1 to each of the recovery sets.

Claim 3. The proposed delivery scheme terminates and allows each user's demand to be satisfied. Furthermore the transmission rate of the server is $\frac{(q-1)n}{k+1}$ and the subpacketization level is $q^k z$.

The main requirement for Claim 3 to hold is that the recovery set bipartite graph be biregular, where multiple edges between the same pair of nodes is disallowed and the degree of each parallel class is z. This follows from the definition of the recovery sets (see [6] for details).

The construction above works for a system where $M/N = 1/q$. It turns out that this can be converted into a scheme for $\frac{M}{N} = 1 - \frac{k+1}{nq}$. Thus, any convex combination of these two points can be obtained by memory-sharing. We say that an equation is of the all-but-one type if it is of the form $W_{d_{i_1},\mathcal{A}_{i_1}} \oplus W_{d_{i_2},\mathcal{A}_{i_2}} \oplus \cdots \oplus W_{d_{i_g},\mathcal{A}_{i_g}}$ where for each $\ell \in [g]$, we have $i_\ell \notin \mathcal{A}_{i_\ell}$ and $i_\ell \in \cap_{i_j : j \in [g], j \neq \ell} \mathcal{A}_{i_j}$. Furthermore, we say that such an equation benefits g users.

Claim 4. Consider a caching system with K users, cache fraction $\frac{M}{N}$ and subpacketization level F_s, where the placement is specified by the incidence matrix of a design. Suppose that in the delivery phase, this system transmits Δ equations where each equation is of the all-but-one type and benefits g users.

Then there exists another coded caching system with $K' = K$ users, cache fraction $\frac{M'}{N'} = \frac{K-g}{K}$ and subpacketization level $F'_s = \Delta$. Furthermore, this system requires the transmission of Δ' equations, each of which allows $g' = K(1 - \frac{M}{N})$ users to simultaneously recover missing subfiles. The overall transmission rate is thus F_s/Δ.

Applying Claim 4 in our context implies the existence of a system with $K' = nq$, $\frac{M'}{N'} = 1 - \frac{k+1}{nq}$, $F'_s = (q-1)q^k \frac{zn}{k+1}$, and transmission rate $R' = \frac{F_s}{\Delta} = \frac{k+1}{(q-1)n}$.

III. Some Classes of Linear Code Satisfies the CCP

Maximum-distance-separable (MDS) codes are clearly a class of codes that satisfy the CCP. In fact, for these codes any k columns of the generator matrix can be shown to be full rank. Note however, that MDS codes typically need large field size, e.g., $q \geq n$. In our construction, the value of $M/N = 1/q$ and the number of users is $K = nq$. Thus, for large n, we will only obtain systems with small values of M/N, or equivalently large values of M/N (by Claim 4 above). This may be restrictive in practice.

However, there are other classes of codes that do not suffer from the field size issue. As shown in our previous work [5], a $(k+1, k)$ single parity check (SPC) code can be defined over the additive group $\mathbb{Z} \mod q$ (where q is not necessarily prime) and such systems with $K = (k+1)q, M/N = 1/q$ and $F = q^k$ can be defined for any integer value of q. More generally, we can find several classes of cyclic codes that satisfy the CCP.

A. Cyclic Codes

A cyclic code is a linear block code, where the circular shift of each codeword is also a codeword [8]. The generator matrix of (n, k) cyclic code over $GF(q)$ is obtained as below.

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & 0 & \cdot & 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} \end{bmatrix}$$

The following claim shows that for verifying the CCP for a cyclic code it suffices to pick *any* set of $k+1$ consecutive columns.

Claim 5. Consider a (n, k) cyclic code with generator matrix $\mathbf{G}$. Let $\mathbf{G}_{\mathcal{S}}$ denote a set of $k+1$ consecutive columns of $\mathbf{G}$. If each $k \times k$ submatrix of $\mathbf{G}_S$ is full rank, then $\mathbf{G}$ satisfies the $(k, k+1)$-CCP.

Claim 5 implies a low complexity search algorithm to determine if a cyclic code satisfies the CCP. Instead of checking all $\mathbf{G}_{\mathcal{S}_a}$, $0 \leq a \leq \frac{zn}{k+1} - 1$, in Definition 2, we only need to check an arbitrary $\mathbf{G}_{\mathcal{S}} = [\mathbf{g}_{(i)_n}, \mathbf{g}_{(i+1)_n}, \cdots, \mathbf{g}_{(i+k)_n}]$, for $0 \leq i < n$. To further simplify the search, we choose $i = n - \lfloor \frac{k}{2} \rfloor - 1$.

For this choice of i, Claim 6 shows that $\mathbf{G}_{\mathcal{S}}$ is such that we only need to check the rank of a list of small size of matrices to determine if each $k \times k$ submatrix of $\mathbf{G}_{\mathcal{S}}$ is full rank.

Claim 6. A cyclic code with generator matrix $\mathbf{G}$ satisfies the CCP if the following conditions hold.

- For $0 < j \leq \lfloor \frac{k}{2} \rfloor$, the submatrices

$$\mathbf{C}_j = \begin{bmatrix} g_{n-k-1} & g_{n-k} & 0 & \cdot & \cdot & 0 \\ g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & & \vdots \\ g_{n-k-j+1} & \cdot & \cdot & \cdot & \cdot & g_{n-k} \\ g_{n-k-j} & \cdot & \cdot & \cdot & \cdot & g_{n-k-1} \end{bmatrix}$$

  have full rank. In the above expression, $g_i = 0$ if $i < 0$.
- For $\lfloor \frac{k}{2} \rfloor < j < k$, the submatrices

$$\mathbf{C}_j = \begin{bmatrix} g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & g_{k-j} \\ g_0 & g_1 & \cdot & \cdot & \cdot & \cdot & g_{k-j-1} \\ \vdots & & & & & & \vdots \\ 0 & \cdot & \cdot & 0 & g_0 & g_1 & g_2 \\ 0 & \cdot & & \cdot\cdot & 0 & g_0 & g_1 \end{bmatrix}$$

  have full rank.

Example 5. Consider the polynomial $\mathbf{g}(X) = X^4 + X^3 + X + 2$ over $GF(3)$. Since it divides $X^8 - 1$, it is the generator polynomial of a $(8, 4)$ cyclic code over $GF(3)$. The generator matrix of this code is given below.

$$\mathbf{G} = \begin{bmatrix} 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

It can be verified that the 4×5 submatrix which consists of the two leftmost columns and three rightmost columns of $\mathbf{G}$ is such that all 4×4 submatrices of it are full rank. Thus, by Claim 5 the (4,5)-CCP is satisfied for $\mathbf{G}$.

IV. Comparison with Existing Schemes

Let R^* and F_s^* denote the rate and the subpacketization level of our proposed scheme. Let R^{MN}, F_s^{MN} be the corresponding quantities for the scheme in [1]. We note that the result in [2], [5] is a special case of our work when the linear block code is chosen as a single parity check code over $\mathbb{Z} \mod q$. In this specific case, q does not need to be a prime power (see [5] for more details). Thus, our results subsume the results of [2].

For comparison with [1], we note that for $\frac{M}{N} = \frac{1}{q}$

$$\frac{R^*}{R^{MN}} = \frac{1+n}{1+k}, \text{ and } \frac{F_s^*}{F_s^{MN}} \approx q^{k-n} z \left(\frac{q-1}{q} \right)^{n(q-1)}.$$

It is not too hard to see that F_s^* is exponentially lower than F_s^{MN}. Thus, our rate is higher, but the subpacketization level is exponentially lower. A similar result can be shown in the case when $M/N = 1 - \frac{k+1}{nq}$ (see [6] for more details).

An alternate way of comparing the results is to keep the transmission rates of both schemes roughly the same and then determine the subpacketization levels. This can be achieved by using memory-sharing in the scheme of [1]. It is hard in general to match the rates exactly. Accordingly, we show this by means of an example below.

Example 6. Consider a $(9, 5)$ linear block code over $GF(2)$ with generator matrix specified below,

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

It can be checked that $\mathbf{G}$ satisfies the $(5, 6)$-CCP. Thus, it corresponds to a coded caching scheme with $K = 9 \times 2 = 18$, $\frac{M}{N} = \frac{1}{2}$, $R^* = \frac{3}{2}$ and $F_s^* = 64$.

We can achieve the almost the same rate by performing memory-sharing using scheme of [1] in a caching scheme with $K = 18$. We divide each file into two smaller subfiles W_n^1 and W_n^2 of equal size. The scheme of [1] is then applied separately on W_n^1 and W_n^2 with $\frac{M_1}{N_1} = \frac{2}{9}$ (corresponding to W_n^1) and $\frac{M_2}{N_2} = \frac{7}{9}$ (corresponding to W_n^2). Thus, the overall cache fraction is still $1/2$. The rate of this scheme is $R^{MN} \approx 3/2$. However, the subpacketization level is $F_s^{MN} = \binom{K}{KM_1/N_1} + \binom{K}{KM_2/N_2} = 6120$.

V. Conclusions and Future Work

In this work we have demonstrated a link between specific classes of linear block codes and the subpacketization problem in coded caching. Leveraging this approach allows us to construct families of coded caching schemes where the subpacketization level is exponentially smaller compared to the approach of [1].

There are several opportunities for future work. Even though our subpacketization level is significantly lower than [1], it still scales exponentially with the number of users. Of course, the rate of growth with the number of users is much smaller. Investigating schemes with subpacketization levels that grow sub-exponentially with K and schemes that work with general values of M/N are interesting directions for future work.

References

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Info. Theory*, vol. 60, pp. 2856–2867, May 2014.

[2] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design in centralized coded caching scheme," preprint, 2016, [Online] Available: http://arxiv.org/abs/1510.05064.

[3] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," preprint, 2016, [Online] Available: https://arxiv.org/abs/1608.03989.

[4] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite length analysis of caching-aided coded multicasting," in *52nd Annual Allerton Conference on Communication, Control, and Computing*, Sept 2014, pp. 914–920.

[5] L. Tang and A. Ramamoorthy, "Coded Caching with Low Subpacketization Levels," in *Workshop on Network Coding (NetCod)*, 2016.

[6] ——, "Low subpacketization schemes for coded caching," 2017, [Online], Available:https://www.ece.iastate.edu/adityar/publications/.

[7] D. R. Stinson, *Combinatorial Designs: Construction and Analysis*. Springer, 2003.

[8] S. Lin and D. J. Costello, *Error Control Coding, 2nd Ed.* Prentice Hall, 2004.