Cache-aided Interference Management Using Hypercube Combinatorial Cache Designs

Xiang Zhang, Nicholas Woolsey, Mingyue Ji
Department of Electrical and Computer Engineering, University of Utah
Salt Lake City, UT, USA
Email: {xiang.zhang, nicholas.woolsey, mingyue.ji}@utah.edu

***Abstract*—We consider a cache-aided interference network which consists of a library of N files, K_T transmitters and K_R receivers (users), each equipped with a local cache of size M_T and M_R files respectively, and connected via a discrete-time additive white Gaussian noise channel. Each receiver requests an arbitrary file from the library. The objective is to design a cache placement without knowing the receivers' requests and a communication scheme such that the sum Degrees of Freedom (sum-DoF) of the delivery is maximized. This network model has been investigated by Naderializadeh *et al.*, who proposed a prefetching and a delivery scheme that achieve a sum-DoF of $\min\{\frac{M_TK_T+K_RM_R}{N}, K_R\}$. One of the biggest limitations of this scheme is the requirement of high subpacketization level. This paper attempts to design new algorithms to reduce the file subpacketization in such a network. In particular, we propose a new approach for both prefetching and linear delivery based on a combinatorial design called *hypercube*. We show that the required number of packets per file can be exponentially reduced compared to the state-of-the-art scheme proposed by Naderializadeh *et al.*, or the NMA scheme. When $\frac{M_TK_T+K_RM_R}{N} \leq K_R$, the achievable one-shot sum-DoF using this approach is $\frac{M_TK_T+K_RM_R}{N}$, which shows that 1) the one-shot sum-DoF scales linearly with the aggregate cache size in the network and 2) it is within a factor of 2 to the information-theoretic optimum. Surprisingly, the identical and near optimal sum-DoF performance can be achieved using the hypercube approach with a much less file subpacketization.**

I. Introduction

Wireless traffic has grown dramatically in recent years due to the increasing mobile data demand, especially due to video delivery services [1]. One promising approach to handle this traffic bottleneck is to exploit local cache memories at end user devices or network edge nodes (e.g., small cell base station) to pre-store part of the contents (e.g, movies) which might be requested in the near future. With the help of these cache nodes, the system can serve users with a much higher rate and lower latency [2]–[10]. Among all schemes based on caching approaches, coded caching, introduced in [2], has attracted significant attention. In particular, Maddah-Ali and Niesen considered a *shared link network* and studied the problem of minimizing the worst-case traffic load or *rate*. It was shown that prefetching packets of the library files in a uniform manner during the placement phase, and employing coded scheme based on linear index code during the delivery phase, is sufficient to provide optimal rate under uncoded cache placement [3], [11]. Later, the idea of coded caching has been extended to Device-to-Device (D2D) caching networks [4], multi-server caching networks [12] and combination caching networks [13], where the channels between the transmitters and receivers are either wireline channel or broadcast noiseless channel.

The concept of coded caching was also extended to the wireless channels with the consideration of superpositions (e.g., interference) of transmitted signals [5]–[10]. In [5], the authors considered a three-user interference channel where only transmitters are equipped with cache memories (no cache memories at the receivers) and showed that via a specific cache prefetching strategy, an efficient delivery scheme can be designed by exploiting the gains based on interference cancellation and interference alignment. In [8], the additive Gaussian channel in a broadcast setting with cache-aided receivers was studied. Later, the study was extended to the case where both transmitters and receivers are equipped with cache. Moreover, cache-aided fog radio access network was also investigated in [9].

As shown in above works, the remarkable multiplicative gain of coded caching in terms of network aggregate memory has been established in the asymptotic regime when the number of packets per file F scales to infinity. It has been shown that in most of the cases, to achieve the desired caching gain, F has to increase exponentially as a function of the number of nodes in the network. The finite length analysis of coded caching of shared link network was initiated in [14], and was later characterized via combinatorial designs [15]. The finite length analysis of coded caching in other network topologies other than shared link and MIMO broadcast channel is very limited. In [16], the authors considered a MISO broadcast channel with L transmitting antennas and showed that reduced subpacketization can be achieved. In addition, they extended the achievable scheme to the cache-aided interference networks. In [17], we considered a D2D caching network over noiseless broadcast channel model and introduced a combinatorial design called *hypercube*, and the corresponding placement and coded delivery schemes with a substantially lower subpacketization level while still achieving order optimal throughput.

In this paper, we consider the general wireless interference network with cache memories equipped at both the transmitter and receiver sides. In particular, we consider a wireless interference network with K_T transmitters and K_R receivers, each equipped with a local cache memory of size M_T and M_R

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

files, from a library of N files. We restrict the communication scheme to one-shot linear schemes due to its practicality. This network model was considered by Naderializadeh, Maddah-Ali and Avestimehr in [6]. We will demonstrate how to reduce the subpacketizaiton level (i.e., F) according to a deterministic combinatorial design called the hypercube approach.

Our main contribution in this paper is two-fold. First, based on the hypercube cache placement introduced in [17], we designed a cache placement scheme at both transmitters and receivers, and proposed a linear one-shot delivery scheme by exploiting zero-forcing opportunities via transmitter collaboration and cache-induced interference cancellation opportunities at receivers' side. The proposed scheme achieves an order-wise subpacketization level reduction compared to that achieved in [6]. Second, when $\frac{K_T M_T + K_R M_R}{N} \leq K_R$, the proposed scheme achieves a one-shot sum-DoF of $\frac{K_T M_T + K_R M_R}{N}$, which is within a factor of 2 to the optimum as shown in [6]. More importantly and surprisingly, it achieves the same sum-DoF as in [6]. This implies that there is no any loss in terms of one-shot sum-DoF by using the proposed scheme while requiring a much less file subpacketization. In the rest of the paper, we will refer the scheme in [6] as NMA scheme.

II. Network Model and Problem Formulation

We use the following notation convention. Calligraphic symbols denote sets. We use $|\cdot|$ to represent the cardinality of a set or the length of a vector or the norm of a random variable; $\mathbb{Z}^+$ denotes the integer set, "$a \mod b$" denotes the module operation of a modulo b, and $[n] := \{0, 1, \cdots, n-1\}$. We also define the *commutative product set* of two sets $\mathcal{A}$ and $\mathcal{B}$ as $\mathcal{A} \times \mathcal{B} = \{\{a, b\} : a \in \mathcal{A}, b \in \mathcal{B}\}$.

A. General Problem Formulation

Consider a wireless interference network, as illustrated in Fig. 1, which consists of K_T transmitters and K_R receivers, denoted by $\{\text{Tx}_i : i \in [K_T]\}$ and $\{\text{Rx}_j : j \in [K_R]\}$ respectively. The system contains a library of N files denoted by $\{\mathcal{W}_n : n \in [N]\}$, where file n contains F packets $\{w_{n,p} : p \in [F]\}$ with size of L bits each, i.e., $w_{n,p} \in \mathbb{F}_2^L$.[1] Transmitters and receivers are equipped with cache memories to store part of the file library. In particular, each transmitter and receiver are equipped with a local cache of size M_T and M_R files, respectively. The communication channel between transmitters and receivers is modeled as discrete-time additive white Gaussian noise channel, which can be written as

$$Y_j(t) = \sum_{i=0}^{K_T - 1} h_{ji} S_i(t) + N_j(t) \tag{1}$$

where t is the index of the time slot. $S_i(t) \in \mathbb{C}$ is the complex transmit signal of Tx_i at time slot t, satisfying the power constraint $\mathbb{E}[|S_i(t)|^2] \leq P$. $Y_j(t)$ is the received signal of Rx_j and $N_j(t) \sim \mathcal{N}(0, 1)$ is the complex additive white Gaussian noise (AWGN) at receiver Rx_j. Moreover, $h_{ji} \in \mathbb{C}$ denotes the complex channel gain from Tx_i to Rx_j, which is assumed to stay unchanged during the entire transmission process and is known to all transmitters and receivers. The system operates in two phases: the *prefetching phase* and the *delivery phase* as described in [6]. In the prefetching phase, each transmitter and receiver can store up to $M_T F$ and $M_R F$ arbitrary packets from the file library, respectively. This phase is done without the prior knowledge of the receivers' requests. In the following delivery phase, the receivers' demands are revealed, i.e., each receiver Rx_j will request a specific file $\mathcal{W}_{d_j}, d_j \in [N]$ from the library. These requests are represented by a *demand vector* defined as $\mathbf{d} \triangleq [d_0,\ d_1,\ \cdots,\ d_{K_R - 1}]$. For a specific demand vector, since the receivers have already cached some packets of their requested files, the transmitters only need to deliver the remaining packets to those receivers. The task in this phase is to design the corresponding transmission procedure based on the cache placement in the prefetching phase so that the receivers' demands can be satisfied. In order to make sure that any possible demands can be satisfied, we require that the entire file library should be cached over all transmitters, i.e., $K_T M_T \geq N$.

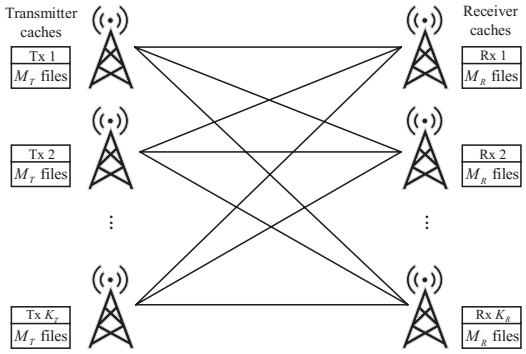


Fig. 1. Wireless interference network consisting of K_T transmitters, each equipped with a cache of size M_T files and K_R receivers, each equipped with a cache of size M_R files. The system also contains a library of N files.

For each cached packet $w_{n,p} \in \mathbb{F}_2^L$, the transmitter performs random Gaussian coding $\psi : \mathbb{F}_2^L \mapsto \mathbb{C}^{\hat{L}}$ to obtain the coded packet $\hat{w}_{n,p} \triangleq \psi(w_{n,p})$ which consists of $\hat{L}$ complex symbols. Assume that the communication will take place in H blocks, each of which consists of $\hat{L}$ time slots. We also assume that one-shot linear scheme is employed in each block $m \in [1 : H]$ to deliver a set of requested (coded) packets $\mathcal{P}_m$ to a subset of the receivers, denoted by $\mathcal{R}_m$. That is, each transmitter $\text{Tx}_i, i \in [K_T]$ will send a coded message

$$s_i^m = \sum_{(n,p): w_{n,p} \in \mathcal{C}_i^{\text{T}} \cap \mathcal{P}_m} \alpha_{i,n,p}^m \hat{w}_{n,p} \tag{2}$$

where $\mathcal{C}_i^{\text{T}}$ denotes the cached contents of Tx_i and $\alpha_{i,n,p}^m$ is the linear combination coefficients used by Tx_i at the m-th block. Accordingly, the received signal of the intended receivers $\text{Rx}_j, j \in \mathcal{R}_m$ in the m-th block is

$$y_j^m = \sum_{i=0}^{K_T - 1} h_{ji} s_i^m + n_j^m \tag{3}$$

where $n_j^m \in \mathbb{C}^{\hat{L}}$ is the random noise at Rx_j in block m.

[1] In this paper, we consider L is a designed variable and equals to $|\mathcal{W}_n|/F$.

Each receiver will utilize its cached contents, consisting of packets stored in the prefetching phase, to subtract some of the interference caused by undesired packets. In particular, each receiver will perform a linear combination $\mathcal{L}_j^m(.)$ (if exists) to recover its requested packets from the overall received signals, as follows

$$\mathcal{L}_j^m(y_j^m, \hat{\mathcal{C}}_j^{\mathrm{R}}) = \hat{w}_{d_j,p} + n_j^m \tag{4}$$

where $\hat{w}_{d_j,p} \in \mathcal{P}_m$ is the desired coded packet of Rx_j and $\hat{\mathcal{C}}_j^{\mathrm{R}}$ denotes the Gaussian coded version of the packets cached by Rx_j.

The one-shot linear sum-DoF is defined as the maximum achievable one-shot linear sum-DoF for the worst case demands under a given caching realization [6], i.e.,

$$\mathsf{DoF}_{\mathrm{L,sum}}^{\left(\{\mathcal{C}_i^{\mathrm{T}}\}_{i=0}^{K_T-1},\{\mathcal{C}_j^{\mathrm{R}}\}_{j=0}^{K_R-1}\right)} = \inf_{\mathbf{d}} \sup_{H,\{\mathcal{P}_m\}_{m=1}^H} \frac{\left|\bigcup_{m=1}^H \mathcal{P}_m\right|}{H} \tag{5}$$

Then the one-shot linear sum-DoF of the network is correspondingly defined as the maximum achievable one-shot linear sum-DoF over all possible caching realizations, i.e.,

$$\mathsf{DoF}_{\mathrm{L,sum}}^*(N, M_T, M_R, K_T, K_R) = \sup_{\{\mathcal{C}_i^{\mathrm{T}}\}_{i=0}^{K_T-1},\{\mathcal{C}_j^{\mathrm{R}}\}_{j=0}^{K_R-1}} \mathsf{DoF}_{\mathrm{L,sum}}^{\left(\{\mathcal{C}_i^{\mathrm{T}}\}_{i=0}^{K_T-1},\{\mathcal{C}_j^{\mathrm{R}}\}_{j=0}^{K_R-1}\right)} \tag{6}$$

in which the cached contents of all transmitter and receivers satisfy the memory constraints, i.e., $|\mathcal{C}_i^{\mathrm{T}}| \le M_T F, \forall i \in [K_T]$ and $|\mathcal{C}_j^{\mathrm{R}}| \le M_R F, \forall j \in [K_R]$.

B. Combinatorial Cache Placement Design

In this paper, the combinatorial cache placement design based on *hypercube*, proposed in [17] to reduce the subpacketization level in wireless D2D networks, is adopted in the prefetching phase. We will show that this scheme can also significantly reduce the required number of packets per file in cache-aided interference networks while using a very different delivery scheme. The details of hypercube cache placement [17] is described as follows.

1) Hypercube cache placement design for wireless D2D caching networks: Consider a wireless D2D network comprising a library of N files, each with F packets, and K users, each of which is equipped with a local cache of size M files. Denote $t \triangleq \frac{KM}{N} \in \mathbb{Z}^+$ as the number of times that each file is cached among all users. In the hypercube cache placement, each file $\mathcal{W}_n$ is partitioned into $\left(\frac{N}{M}\right)^t$ subfiles, i.e., $\mathcal{W}_n = \left\{\mathcal{W}_{n,(\ell_0,\ell_1,\cdots,\ell_{t-1})} : \ell_j \in [\frac{N}{M}], j \in [t]\right\}$. In the prefetching phase, each user $u \in [K]$ caches a set of subfiles $\left\{\mathcal{W}_{n,(\ell_0,\ell_1,\cdots,\ell_{t-1})} : n \in [N]\right\}$, where $\ell_j = u \mod \frac{N}{M}$, for $j = \lfloor u/\left(\frac{N}{M}\right)\rfloor$, and $\ell_i \in [\frac{N}{M}]$ for any $i \neq j$. As a result, each user will cache $(\frac{N}{M})^{t-1}$ packets from each file $\mathcal{W}_n$. It can be verified that the total number of packets cached by any user is equal to $N(\frac{N}{M})^{t-1} = N\frac{F}{N/M} = MF$, satisfying the memory constraint. Under the hypercube file partition method, each packet will represent a lattice point with coordinate $(\ell_0, \ell_1, \cdots, \ell_{t-1})$ in the t-dimensional hypercube, and $\frac{N}{M}$ is the number of lattice points on each edge. The hypercube cache placement is illustrated via the following example.

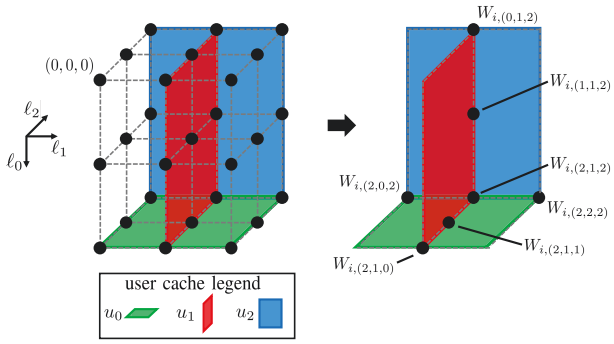


Fig. 2. A 3-dimensional example of the hypercube cache placement. Each subfile is represented by a unique lattice point in the 3-dimensional hypercube (cube). Each of the 9 users caches a set of packets represented by plane of lattice points. As a result, each user caches $9 \times 9 = 81$ subfiles in total.

Example 1: Consider a set of 9 users labelled as $\{0, 1, \cdots, 8\}$ and a set of files $\mathcal{W}_n, n \in \{0, 1, \cdots, 8\}$. We partition the users into $t \triangleq \frac{KM}{N} = 3$ disjoint groups as $\mathcal{U}_0 = \{0, 1, 2\}$, $\mathcal{U}_1 = \{3, 4, 5\}$ and $\mathcal{U}_2 = \{6, 7, 8\}$. We also split each file $\mathcal{W}_i$ into $3^3 = 27$ packets $\mathcal{W}_i = \{\mathcal{W}_{i,(\ell_0,\ell_1,\ell_2)} : \ell_0, \ell_1, \ell_2 \in [3]\}$, each of which can be represented by a unique lattice point in the 3-dimensional cube (Fig. 2). As a result, each lattice point will represent a set of $N = 9$ packets, each from a distinct file. For the cache placement, each user caches all packets represented by a plane of lattice points of the cube. For example, user $u_0 = 2, u_1 = 4$ and $u_2 = 8$ will cache subfiles represented by the green, red and blue planes respectively. We can see that the set of packets $\{\mathcal{W}_{i,(2,1,2)} : i \in [9]\}$ represented by the lattice point $(2, 1, 2)$, which is intersection of the three orthogonal planes of different colors, is cached exclusively by users u_0, u_1 and u_2. Similarly, each subfile is cached by three distinct users. △

2) Hypercube cache placement design for cache-aided interference networks: Different from the D2D settings in [17], in cache-aided interference networks, we have explicit transmitters and receivers instead of D2D users. However, we can still group the transmitters and receivers into different dimensions in order to implement the hypercube placement approach. Let $D_T \triangleq \frac{N}{M_T} \in \mathbb{Z}^+$ and $D_R \triangleq \frac{N}{M_R} \in \mathbb{Z}^+$. For the set of $K_T = D_T t_T$ transmitters $\{\mathrm{Tx}_k : k \in [K_T]\}$, we denote the t_T dimensions of the transmitters as $\mathcal{U}_i^{\mathrm{T}} = \{k : \lfloor \frac{k}{D_T} \rfloor = i\}, i \in [t_T]$. Similarly, for the set of $K_R = D_R t_R$ receivers $\{\mathrm{Rx}_k : k \in [K_R]\}$, we denote the t_R dimensions of the receivers as $\mathcal{U}_j^{\mathrm{R}} = \{k : \lfloor \frac{k}{D_R} \rfloor = j\}, j \in [t_R]$. Here we require that $|\mathcal{U}_i^{\mathrm{T}}| = D_T, \forall i \in [t_T]$ and $|\mathcal{U}_i^{\mathrm{R}}| = D_R, \forall i \in [t_R]$.

The prefetching phase is described as follows.

Prefetching Phase: The hypercube cache placement is employed at both the transmitters' and receivers' sides. That is, each file $\mathcal{W}_n$ is partitioned into ${D_T}^{t_T} {D_R}^{t_R} = (\frac{N}{M_T})^{t_T} (\frac{N}{M_R})^{t_R}$ disjoint equal-size subfiles, denoted by

$$\mathcal{W}_n = \{\mathcal{W}_{n,\mathcal{T},\mathcal{R}}\}_{\substack{\mathcal{T} \in \mathcal{U}_0^{\mathrm{T}} \times \mathcal{U}_1^{\mathrm{T}} \times \cdots \times \mathcal{U}_{t_T-1}^{\mathrm{T}} \\ \mathcal{R} \in \mathcal{U}_0^{\mathrm{R}} \times \mathcal{U}_1^{\mathrm{R}} \times \cdots \times \mathcal{U}_{t_R-1}^{\mathrm{R}}}} \tag{7}$$

where $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{t_T-1}\}$ such that $\tau_i \in \mathcal{U}_i^{\mathrm{T}}, i \in [t_T]$ and $\mathcal{R} = \{r_0, r_1, \cdots, r_{t_R-1}\}$ such that $r_j \in \mathcal{U}_j^{\mathrm{R}}, j \in [t_R]$. Under this file partitioning strategy, each transmitter Tx_i caches a set of subfiles $\{\mathcal{W}_{n,\mathcal{T},\mathcal{R}} : i \in \mathcal{T}\}$ and each receiver Tx_j caches a set of subfiles $\{\mathcal{W}_{n,\mathcal{T},\mathcal{R}} : j \in \mathcal{R}\}$. As a result, the number of packets cached by Tx_i is equal to

$$N{D_T}^{t_T-1}{D_R}^{t_R}\frac{F}{{D_T}^{t_T}{D_R}^{t_R}} = M_T F \text{ packets} \tag{8}$$

where $\frac{F}{{D_T}^{t_T}{D_R}^{t_R}}$ is the number of packets of each subfile and ${D_T}^{t_T-1}{D_R}^{t_R}$ is the number of subfiles cached by Tx_i. It can be easily verified that the memory constraint of transmitters is satisfied. Similarly, the number of packets cached by Rx_j is equal to

$$N{D_T}^{t_T}{D_R}^{t_R-1}\frac{F}{{D_T}^{t_T}{D_R}^{t_R}} = M_R F \text{ packets} \tag{9}$$

which also satisfies the memory constraint. The hypercube cache placement approach in cache-aided interference networks is illustrated as follows.

Example 2: Consider a wireless network with $K_T = 4$ transmitters and $K_R = 4$ receivers. Each transmitter and receiver is equipped with a cache of size $M_T = 2$ and $M_R = 2$ files respectively. The file library contains $N = 4$ files denoted by $\mathcal{W}_0 = A, \mathcal{W}_1 = B, \mathcal{W}_2 = C$ and $\mathcal{W}_3 = D$.

In the prefetching phase, each file $\mathcal{W}_n$ is split into $2^2 \times 2^2 = 16$ disjoint subfiles $\mathcal{W}_{n,\mathcal{T},\mathcal{R}}$ of equal sizes for any $\mathcal{T} \in \{\{0,2\},\{0,3\},\{1,2\},\{1,3\}\}$ and $\mathcal{R} \in \{\{0,2\},\{0,3\},\{1,2\},\{1,3\}\}$. Each subfile is then cached by the two transmitters in $\mathcal{T}$ and the two receivers in $\mathcal{R}$, respectively. For example, file A is split into 16 subfiles:[2]

$$\begin{aligned} &A_{02,02},\ A_{02,03},\ A_{02,12},\ A_{02,13},\\ &A_{03,02},\ A_{03,03},\ A_{03,12},\ A_{03,13},\\ &A_{12,02},\ A_{12,03},\ A_{12,12},\ A_{12,13},\\ &A_{13,02},\ A_{13,03},\ A_{13,12},\ A_{13,13} \end{aligned}$$

where for example, $A_{02,02}$ is cached by transmitters Tx_0 and Tx_2 as well as receivers Rx_0 and Rx_2. The same file partitioning is done for files B, C and D.

It can be seen that each transmitter will cache 8 subfiles of each file. Since each subfile contains $\frac{F}{16}$ packets, the total number of packets cached by each transmitter is $4 \times 8 \times \frac{F}{16} = 2F$, which satisfies the memory constraint of the transmitters. Similarly, the memory constraint of the receivers is also satisfied. △

III. Main Results

The main results on the one-shot linear sum-DoF using the hypercube cache placement approach are presented in this section. Since for the case where $K_R < \frac{K_T M_T + K_R M_R}{N}$, it is argued in [6] that the one-shot linear sum-DoF of K_R is always achievable by utilizing only a fraction of the Tx and Rx cache memories, we focus on the case where $K_R \geq \frac{K_T M_T + K_R M_R}{N}$.

[2] With a slight abuse of notation, we write $A_{\{0,2\},\{0,2\}}$ as $A_{02,02}$ for simplicity and same for other symbols.

Theorem 1: For a $K_T \times K_R$ wireless interference network with a library of N files, each consisting of F packets, and with transmitter and receiver cache sizes of $M_T F$ and $M_R F$ packets respectively, given the hypercube cache placement approach employed in the prefetching phase, and for any $\delta \triangleq \frac{t_T}{t_R} \in \mathbb{Z}^+$, $D_R = \frac{K_R}{t_R} \geq \delta + 1$, where $t_T \in [K_T], t_R \in [K_R]$, the one-shot linear sum-DoF of $\frac{K_T M_T + K_R M_R}{N}$ is achievable when $K_R \geq \frac{K_T M_T + K_R M_R}{N}$ with $F = \left(\frac{N}{M_T}\right)^{t_T}\left(\frac{N}{M_R}\right)^{t_R}\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$. □

IV. Achievable Delivery Scheme

We present the achievable delivery scheme under the hypercube cache placement via the following example.

Example 3: We consider the same network settings as in Example 2. Let each receiver Rx_j request a specific file $\mathcal{W}_{d_j}$ from the library. Without loss of generality, we assume $\mathcal{W}_{d_0} = A, \mathcal{W}_{d_1} = B, \mathcal{W}_{d_2} = C$ and $\mathcal{W}_{d_3} = D$, respectively. In the prefetching phase, each receiver has already cached 8 subfiles of its requested file. Therefore, the transmitters only need to transmit the $16 - 8 = 8$ remaining subfiles to each receiver. More specifically, the following 32 subfiles should be transmitted:

$$\left.\begin{aligned} &A_{02,12},\ A_{03,12},\ A_{12,12},\ A_{13,12},\\ &A_{02,13},\ A_{03,13},\ A_{12,13},\ A_{13,13} \end{aligned}\right\} \text{to } \mathrm{Rx}_0$$

$$\left.\begin{aligned} &B_{02,02},\ B_{03,02},\ B_{12,02},\ B_{13,02},\\ &B_{02,03},\ B_{03,03},\ B_{12,03},\ B_{13,03} \end{aligned}\right\} \text{to } \mathrm{Rx}_1$$

$$\left.\begin{aligned} &C_{02,03},\ C_{03,03},\ C_{12,03},\ C_{13,03},\\ &C_{02,13},\ C_{03,13},\ C_{12,13},\ C_{13,13} \end{aligned}\right\} \text{to } \mathrm{Rx}_2$$

$$\left.\begin{aligned} &D_{02,02},\ D_{03,02},\ D_{12,02},\ D_{13,02},\\ &D_{02,12},\ D_{03,12},\ D_{12,12},\ D_{13,12} \end{aligned}\right\} \text{to } \mathrm{Rx}_3$$

In this example, since for $\delta = 1$, $D_R = 2$, and $t_R = 2$, $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1} = 1$, no further file partitioning is needed. Hence, there are 32 packets to be delivered to the receivers.

We now show how the above 32 packets can be grouped in 8 subsets, each of which contains 4 packets, such that the packets within the same subset can be transmitted simultaneously to the receivers without interference. Fig. 3 shows how the packet grouping and the corresponding transmissions are done. In each communication step, 4 packets are delivered to the receivers simultaneously, and the interference between different users can be effectively eliminated by choosing proper linear combination coefficients at all transmitters. For example, in step 1 shown Fig. 3, 4 packets $A_{02,12}, B_{13,03}, C_{12,13}$ and $D_{03,02}$ are transmitted to receivers Rx_0, Rx_1, Rx_2 and Rx_3 respectively. We write the transmit signals of each transmitter as a linear combination of some of these 4 packets as follows:

$$\begin{aligned} S_0 &= h_{32}\hat{A}_{02,12} - h_{13}\hat{D}_{03,02}\\ S_1 &= h_{23}\hat{B}_{13,03} - h_{02}\hat{C}_{12,13}\\ S_2 &= h_{01}\hat{C}_{12,13} - h_{30}\hat{A}_{02,12}\\ S_3 &= h_{10}\hat{D}_{03,02} - h_{21}\hat{B}_{13,03} \end{aligned}$$

where for each packet $\mathcal{W}_{n,\mathcal{T},\mathcal{R}}$, $\hat{W}_{n,\mathcal{T},\mathcal{R}}$ denotes its physical layer coded version. As a result, after the interference cancellation over the air, the corresponding received signals by Rx_0, Rx_1, Rx_2 and Rx_3 are given by

$$\begin{aligned}
Y_0 =&(h_{32}h_{00}-h_{30}h_{12})\hat{A}_{02,12}+(h_{23}h_{01}-h_{21}h_{03})\hat{B}_{13,03}\\
&+(h_{10}h_{03}-h_{13}h_{00})\hat{D}_{03,02}+N_0\\
Y_1 =&(h_{23}h_{11}-h_{21}h_{13})\hat{B}_{13,03}+(h_{32}h_{10}-h_{30}h_{12})\hat{A}_{02,12}\\
&+(h_{02}h_{11}-h_{01}h_{12})\hat{C}_{12,13}+N_1\\
Y_2 =&(h_{01}h_{22}-h_{02}h_{21})\hat{C}_{12,13}+(h_{32}h_{20}-h_{30}h_{22})\hat{A}_{02,12}\\
&+(h_{10}h_{23}-h_{13}h_{20})\hat{D}_{03,02}+N_2\\
Y_3 =&(h_{10}h_{33}-h_{13}h_{30})\hat{D}_{03,02}+(h_{23}h_{31}-h_{21}h_{33})\hat{B}_{13,03}\\
&+(h_{01}h_{32}-h_{02}h_{31})\hat{C}_{12,13}+N_3,
\end{aligned}$$

where $N_i, i \in [4]$ is the random noise.

We can see that receiver Rx_0 can cancel the interference caused by $B_{13,03}$ and $D_{03,02}$ since they have already been cached by Rx_0. Similarly, Rx_1, Rx_2 and Rx_3 can also cancel the interference caused by undesired packets by utilizing their cached contents. Therefore, all the interference including inter-user interference and interference caused by cached packets can be eliminated so that all receivers can decode their desired subfiles. It can be verified that there exists such linear combinations and all receivers can decode their desired packets in all remaining 7 communication steps. Hence, the 32 packets, each consisting of $\frac{|\mathcal{W}_n|}{16}$ bits, can be delivered to the receivers in 8 communication steps, each containing $\frac{F}{16}=1$ resource block. As a result, a sum-DoF of $\frac{32}{8}=4=\frac{K_TM_T+K_RM_R}{N}$ can be achieved. Hence, the proposed file subpacketization, placement, precoding and scheduling strategy in the delivery phase allow transmitters to collaboratively zero-force some of the outgoing interference and allow receivers to cancel the leftover interference using cached contents for any receivers' demands. $\triangle$

A. Sketch of General Delivery Schemes

In this section, we will sketch the general delivery scheme. Let the receivers' demand vector be $\mathbf{d}=[d_0\ d_1\ \cdots\ d_{K_R-1}]$, i.e., receiver $\mathrm{Rx}_j, j\in[K_R]$ requests the file $\mathcal{W}_{d_j}$. Since some subfiles of the requested file have already been cached by the receiver in the prefetching phase, the transmitters only need to send those subfiles which have not been cached by Rx_j, i.e., $\{\mathcal{W}_{d_j,\mathcal{T},\mathcal{R}}: j\notin\mathcal{R}\}$.

Following similar approach in [6], we need to further partition the set of subfiles which need to be delivered to the receivers into smaller subfiles (or packets) so that they can be scheduled in groups of packets with size t_T+t_R and delivered to the corresponding t_T+t_R receivers simultaneously without interference. In particular, for each packet in a group of size t_T+t_R, it is desired by one particular receiver and can be cancelled by another t_R receivers using their cached contents. In addition, some of the t_T out of the t_T+t_R transmitters can collaboratively zero-force the interference to another t_T-1 unintended receivers. To illustrate the further subpacketization of the subfiles, we need the following definition.

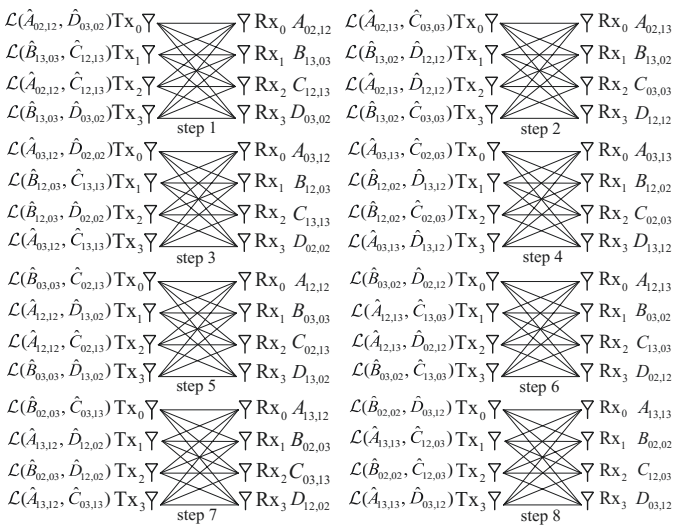


Fig. 3. Delivery phase for the Example 3, in which four receivers $\mathrm{Rx}_j, j\in[4]$ request four different files A, B, C and D respectively. $\mathcal{L}(x,y)$ denotes some linear combination of x and y, i.e., $\mathcal{L}(x,y)=\alpha x+\beta y$, where α and β are some constants.

Definition 1: **(Hypercube Permutation)** Given a set of $D\times t$ points, denoted by $\mathcal{Q}$, we label each of these points by a unique number $u_{i,j}\in[Dt]$, where $i\in[t], j\in[D]$. Assume that these points can be divided into t disjoint groups, which we refer to as *dimensions*. Each dimension consists of D points, denoted by $\mathcal{U}_i=\left\{u_{i,j}:\lfloor\frac{u_{i,j}}{D}\rfloor=i, j=0,1,\cdots,D-1\right\}$, $i\in[t]$. Define the *hypercube permutation* of the set $\mathcal{Q}$, denoted by $\pi^{\mathrm{HCB}}=[\,\pi(0)\ \pi(1)\ \cdots\ \pi(Dt-1)\,]$, as such a permutation of the $D\times t$ points that satisfies the following condition: For the set of users in $\mathcal{U}_i, i\in[t]$, the positions in the permutation (denoted by $pos(\cdot)$, meaning that $pos(u)=i$ if $\pi(i)=u$) of any two of them, u_{i,j_1} and u_{i,j_2} $(j_1\neq j_2)$, should satisfy $|pos(u_{i,j_1})-pos(u_{i,j_2})|=kt, 1\leq k\leq D-1, k\in\mathbb{Z}^+$ and $j_1,j_2\in[D]$. We also define the *circular hypercube permutation* of a set $\mathcal{Q}$ as a way of arranging the elements of $\mathcal{Q}$ around a fixed table, and meanwhile, the corresponding arrangement should be a hypercube permutation. $\Diamond$

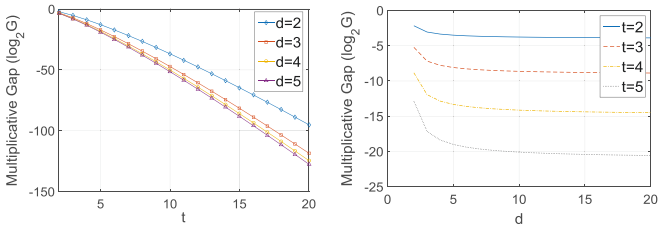
We illustrate the concept of *hypercube permutation* and *circular hypercube permutation* by the following example.

Example 4: For $\mathcal{Q}=\{0,1,2,3\}$ with $t=2$ dimensions and $D=2$ points in each dimension, i.e., $\mathcal{U}_0=\{0,1\}, \mathcal{U}_1=\{2,3\}$, we have

$$\begin{aligned}
\Pi_{\mathcal{Q}}^{\mathrm{HCB}}=\Big\{&[\,0\,2\,1\,3\,],[\,0\,3\,1\,2\,],[\,1\,2\,0\,3\,],[\,1\,3\,0\,2\,],\\
&[\,2\,1\,3\,0\,],[\,2\,0\,3\,1\,],[\,3\,1\,2\,0\,],[\,3\,0\,2\,1\,]\Big\}
\end{aligned}$$

It is clear that, for any two points within one dimension, $0,1\in\mathcal{U}_0$ or $2,3\in\mathcal{U}_1$, we have $|pos(0)-pos(1)|=|pos(2)-pos(3)|=2$, which satisfies the condition $|pos(u_{i,j_1})-pos(u_{i,j_2})|=t$ (note that $k=1$). Furthermore, we have $\Pi_{\mathcal{Q}}^{\mathrm{HCB,circ}}=\left\{[\,0\,2\,1\,3\,],[\,0\,3\,1\,2\,]\right\}$. $\triangle$

Based on the concept of circular hypercube permutations, we can illustrate the further subpacketization of the subfiles as follows. For any $j\in[K_R]$, $\mathcal{T}\in\mathcal{U}_0^{\mathrm{T}}\times\mathcal{U}_1^{\mathrm{T}}\times\cdots\times\mathcal{U}_{t_T-1}^{\mathrm{T}}$,



(a) $G(d,t,1)$ versus t (b) $G(d,t,1)$ versus d

Fig. 4. The multiplicative gap G between the hypercube scheme and the NMA scheme. The comparison is down under the setting $t_T = t_R = t$, $N/M_T = N/M_R = d$, which implies $K_T = K_R = dt$. It can be seen that: (a) For a fixed d, G decreases quickly as t increases and approaches zero as t goes to infinity, and (b) For a fixed t, G converges to some specific non-zero value as d goes to infinity.

and $\mathcal{R} \in \mathcal{U}_0^{\mathrm{R}} \times \mathcal{U}_1^{\mathrm{R}} \times \cdots \times \mathcal{U}_{\lfloor \frac{j}{D_R} \rfloor}^{\mathrm{R}} \setminus \{j\} \times \cdots \times \mathcal{U}_{t_R-1}^{\mathrm{R}}$, where $|\mathcal{T}| = t_T$ and $|\mathcal{R}| = t_R$, we partition $\mathcal{W}_{d_j,\mathcal{T},\mathcal{R}}$ into $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$ packets. Using such a further partition,, the delivery phase can be carried out such that any receiver demand vector $\mathbf{d}$ can be satisfied with a sum-DoF of $tT + tR$.

V. Discussion

In this section , we provide a comprehensive performance comparison of the hypercube cache placement based interference cancellation scheme and the NMA scheme.

Each file in the library is partitioned into $(\frac{N}{M_T})^{t_T}(\frac{N}{M_R})^{t_R}$ subfiles in the hypercube based scheme while this number is $\binom{K_T}{t_T}\binom{K_R}{t_R}$ in the NMA scheme. In the delivery phase, to implement interference cancellation, each subfile which is requested by some receivers is further partitioned into $\Delta_{\mathrm{HCB}} \triangleq \binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$ packets in the proposed hypercube based scheme and $\Delta_{\mathrm{NMA}} \triangleq t_R!\binom{K_R-t_R-1}{t_T-1}(t_T-1)!$ packets in the NMA scheme. Thus, the overall number of packets for a specific requested file required for these two schemes are $F_{\mathrm{HCB}} = {D_T}^{t_T}{D_R}^{t_R-1}(D_R-1)\Delta_{\mathrm{HCB}}$ and $F_{\mathrm{NMA}} = \binom{K_T}{t_T}\binom{K_R-1}{t_R}\Delta_{\mathrm{NMA}}$ respectively. We define the *multiplicative gap* of the subpacketization levels between these two schemes, denoted by G, as $G \triangleq \frac{F_{\mathrm{HCB}}}{F_{\mathrm{NMA}}}$. We next show that for any system parameters, the hypercube scheme has a strictly less subpacketization level than the NMA scheme.

Theorem 2: For any system parameters K_T, K_R, M_T, M_R and N satisfying $\delta \triangleq \frac{t_T}{t_R} \in \mathbb{Z}^+$, the multiplicative gap G is less than 1. *Moreover, under the setting $t_T = \delta t_R = \delta t$, $D_T = D_R = d$,*

$$\lim_{t\to\infty} G(d,t,\delta) = 0 \tag{10}$$

$$\lim_{d\to\infty} G(d,t,\delta) = \frac{(t-1)!(\delta t)!}{\delta^{\delta t}t^{(2\delta+1)t-1}} \tag{11}$$

More specifically, when $\delta \geq 2$ and for sufficiently large t,

$$G(d,t,\delta) \leq \frac{\mathcal{K}_0}{t^{(\delta-1)t-1}} \tag{12}$$

where $\mathcal{K}_0 = \frac{2\pi}{d}\sqrt{\frac{\delta(d-\delta-1)}{d-1}}$. □

One important implication of Theorem 2 is the subpacketzation level reduction of the hypercube scheme over the NMA scheme, which yields a significant advantage since it holds for any possible system parameters while preserving the same one-shot linear sum-DoF gain. Fig. 4 shows the multiplicative gain $G(d,t,1)$ under logarithmic scale for the case when $\delta \triangleq \frac{t_T}{t_R} = 1, t_T = t_R = t$ and $D_R = D_R = d$. It can be seen that the gap decreases exponentially as t increases and goes to zero as t goes to infinity (see Fig. 4(a)) and G converges to some specific value as d goes to infinity (see Fig. 4(b)).

Acknowledgments

This work is supported by NSF 1817154 and NSF 1824558.

References

[1] Visual Networking Index Cisco, "Global mobile data traffic forecast update, 2015–2020 white paper," *Document ID*, vol. 958959758, 2016.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *Information Theory, IEEE Transactions on*, vol. 60, no. 5, pp. 2856–2867, 2014.

[3] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 135–139.

[4] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 849–869, Feb 2016.

[5] M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 809–813.

[6] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 3092–3107, May 2017.

[7] J. Hachem, U. Niesen, and S. N. Diggavi, "Degrees of freedom of cache-aided wireless interference networks," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 5359–5380, July 2018.

[8] S. S. Bidokhti, M. Wigger, and A. Yener, "Gaussian broadcast channels with receiver cache assignment," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[9] A. Sengupta, R. Tandon, and O. Simeone, "Fog-aided wireless networks for content delivery: Fundamental latency tradeoffs," *IEEE Transactions on Information Theory*, vol. 63, no. 10, pp. 6650–6678, Oct 2017.

[10] S. P. Shariatpanahi, G. Caire, and B. H. Khalaj, "Physical-layer schemes for wireless coded caching," *arXiv preprint arXiv:1711.05969*, 2017.

[11] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, Feb 2018.

[12] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 7253–7271, Dec 2016.

[13] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "State-of-the-art in cache-aided combination networks," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct 2017, pp. 641–645.

[14] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., "Finite-length analysis of caching-aided coded multicasting," *IEEE Transactions on Information Theory*, vol. 62, no. 10, pp. 5524–5537, Oct 2016.

[15] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using ruzsa-szemeredi graphs," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 1237–1241.

[16] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1176–1188, 2018.

[17] N. Woolsey, R.-R. Chen, and M. Ji, "Towards practical file packetizations in wireless device-to-device caching networks," *arXiv preprint arXiv:1712.07221*, 2017.