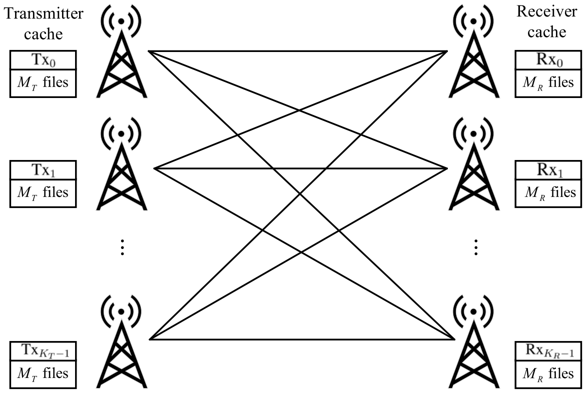


Fig. 1. Wireless interference network consisting of K_T transmitters, each equipped with a cache of size M_T files and K_R receivers, each equipped with a cache of size M_R files. The system also contains a library of N files.

and is known to all transmitters and receivers. The system operates in two phases: the *prefetching phase* and the *delivery phase* as described in [7]. In the prefetching phase, each transmitter and receiver can store up to M_TF and M_RF arbitrary packets from the file library, respectively. This phase is done without the prior knowledge of the receivers' future requests. In the following delivery phase, each receiver Rx_j randomly requests a file $\mathcal{W}_{d_j}, d_j \in [N]$ from the library. These requests are represented by a *demand vector* denoted 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 an efficient transmission procedure based on the cache placement in the prefetching phase so that the receivers' demands can be satisfied. In order to guarantee that any possible demands can be satisfied, we require that the entire file library is cached among all transmitters, i.e., $K_TM_T \geq N$.

For each cached packet $w_{n,p} \in \mathbb{F}_2^L$, the transmitter performs a random Gaussian coding scheme $\psi : \mathbb{F}_2^L \mapsto \mathbb{C}^{\hat{L}}$ with rate $\log P + o(\log P)$ to obtain the coded packet $\hat{w}_{n,p} \triangleq \psi(w_{n,p})$ consisting of $\hat{L}$ complex symbols, so that each coded packet carries one DoF. Assume that the communication will take place in H blocks, each of which consists of $\hat{L}$ time slots. In addition, we allow only one-shot linear transmission schemes 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 $\mathrm{Tx}_i, i \in [K_T]$ will send a linearly coded message

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

where $\mathcal{C}_i^{\mathrm{T}}$ 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 $\mathrm{Rx}_j, j \in \mathcal{R}_m$ 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. 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 operation $\mathcal{L}_j^m(.)$ if possible in block m to recover its requested packets from all 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 channel created by (4) is a point-to-point channel with capacity $\log P + o(\log P)$. Since each coded packet $\hat{w}_{d_j,p}$ is encoded with rate $\log P + o(\log P)$, it can be decoded with vanishing error probability as L increases.

Since each coded packet carries exactly one DoF, a sum-DoF of $|\mathcal{P}_m|$ can be achieved in block m. Therefore, the one-shot linear sum-DoF of $\left|\cup_{m=1}^H \mathcal{P}_m\right|/H$ can be achieved throughout the delivery phase. As a result, 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 [7], i.e.,

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

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.,

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

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

B. Combinatorial Cache Placement Design

In this paper, the combinatorial cache placement design based on *hypercube*, proposed in [28], [29] to reduce the subpacketization level in wireless D2D networks is adopted in the prefetching phase. The hypercube cache placement has a nice geometric interpretation: each packet of the file can be represented by a lattice point in a high-dimensional hypercube and the cached content of each D2D node is represented by a hyperplane in that hypercube (see Fig. 2). Based on the hypercube cache placement and the corresponding communication scheme, order-optimal rate can be achieved with exponentially less number of packets compared to the Ji-Caire-Molisch (JCM) scheme [5]. It turns out that by a non-trivial extension, the hypercube scheme can also significantly reduce the required subpacketization in cache-aided interference networks. The details of hypercube cache placement [28], [29] is described as follows.

1) Hypercube Cache Placement Design for Wireless D2D Caching Networks: Consider a wireless D2D network consisting of a library of N files, each with F packets, and K users,

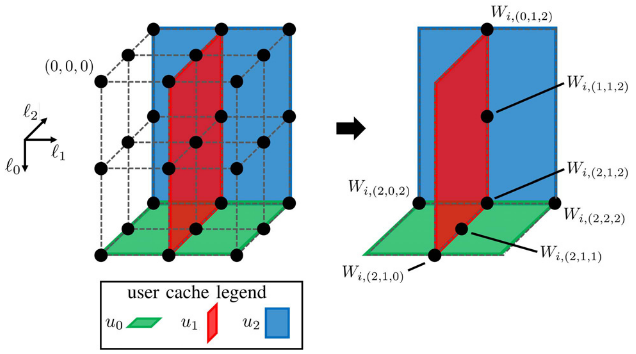


Fig. 2. A 3-dimensional example of the hypercube cache placement. Each subfile is represented by a unique lattice point in the 3-dimensional hypercube (cube). Each of the 9 users caches a set of packets represented by plane of lattice points. As a result, each user caches $9 \times 9 = 81$ subfiles in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

each of which is equipped with a local cache memory of size M files, or equivalently, MF packets. The *caching parameter*, defined as $t \triangleq KM/N \in [1 : K]$, represents the average number of times that each file is cached among all users. In the hypercube cache placement, each file $\mathcal{W}_n$ is split into $(N/M)^t$ subfiles[4] (assuming that N/M and t are both positive integers), i.e., $\mathcal{W}_n = \left\{\mathcal{W}_{n,(\ell_0,\ell_1,\cdots,\ell_{t-1})} : \ell_j \in [N/M],\ j \in [t]\right\}$. It can be seen that each subfile of a file $\mathcal{W}_n$ is uniquely marked by a t-tuple $(\ell_0, \ell_1, \cdots, \ell_{t-1})$ where $\ell_j, j \in [t]$ represents the index of the lattice point along the j-th dimension. In the prefetching phase, each user $u \in [K]$ caches a set of subfiles $\left\{\mathcal{W}_{n,(\ell_0,\ell_1,\cdots,\ell_{t-1})} : \forall n \in [N]\right\}$, where $\ell_j = u \mod (N/M)$, for $j = \lfloor u/\left(N/M\right)\rfloor$, and $\ell_i \in [N/M]$ for any $i \neq j$. As a result, each user will cache $(N/M)^{t-1}$ subfiles from each file $\mathcal{W}_n$. It can be verified that the total number of subfiles cached by any user is equal to $N(\frac{N}{M})^{t-1} = N\frac{(N/M)^t}{N/M} = N\frac{F}{N/M} = MF$, satisfying the memory constraint. The hypercube cache placement has a nice geometric interpretation. Under the hypercube file splitting method, each subfile will represent a lattice point with coordinate $(\ell_0, \ell_1, \cdots, \ell_{t-1})$ in a t-dimensional hypercube, and $N/M \in \mathbb{Z}^+$ is the number of lattice points along each dimension. We will further illustrate the details of the hypercube cache placement via the following example.

Example 1 (Hypercube Cache Placement): Consider a set of $K = 9$ users labeled as $0, 1, \cdots, 8$ and a set of $N = 9$ files $\{\mathcal{W}_n, n \in [9]\}$. Each user has a cache memory of size $M = 3$ files. We first partition the users into $t \triangleq KM/N = 3$ groups denoted by $\mathcal{U}_0 = \{0, 1, 2\}$, $\mathcal{U}_1 = \{3, 4, 5\}$ and $\mathcal{U}_2 = \{6, 7, 8\}$. Each file $\mathcal{W}_n$ is split into $(N/M)^t = 27$ subfiles, i.e., $\mathcal{W}_n = \{\mathcal{W}_{n,(\ell_0,\ell_1,\ell_2)} : \ell_0, \ell_1, \ell_2 \in [3]\}$, each of which can be represented by a unique lattice point in a 3-dimensional cube (see Fig. 2). As a result, each lattice point will represent a set of $N = 9$ subfiles, each from a distinct file. For the cache placement, each user caches all subfiles 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 in Fig. 2. We can see that the set of subfiles $\{\mathcal{W}_{n,(2,1,2)} : \forall n \in \ [9]\}$ represented by the lattice point $(2, 1, 2)$, which is the 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. $\triangle$

2) Hypercube Cache Placement Design for cache-aided interference networks: Different from the D2D setting in [28], in cache-aided interference networks, we have a set of explicit transmitters and receivers instead of D2D users. However, the hypercube approach can still be applied to design the cache placement in the case illustrated as follows.

File Splitting: let $D_T \triangleq N/M_T \in \mathbb{Z}^+$ and $D_R \triangleq N/M_R \in \mathbb{Z}^+$ denote the number of transmitters and receivers on each edge of the hypercube associated with the transmitters' cache and receivers' cache respectively.[5] For the set of $K_T = D_T t_T$ transmitters $\{\mathrm{Tx}_k : k \in [K_T]\}$, we denote the $t_T \triangleq K_T M_T/N$ dimensions of the transmitters as $\mathcal{U}_i^{\mathrm{T}} = \{k : \lfloor k/D_T \rfloor = i\}, \forall i \in [t_T]$.[6] Similarly, for the set of $K_R = D_R t_R$ receivers $\{\mathrm{Rx}_k : k \in [K_R]\}$, we denote the $t_R \triangleq K_R M_R/N$ dimensions of the receivers as $\mathcal{U}_j^{\mathrm{R}} = \{k : \lfloor k/D_R \rfloor = j\}, \forall j \in [t_R]$. It can be seen that $|\mathcal{U}_i^{\mathrm{T}}| = D_T$, $\forall i \in [t_T]$ and $|\mathcal{U}_j^{\mathrm{R}}| = D_R$, $\forall j \in [t_R]$, i.e., for both the transmitter and the receiver hypercubes, all distinct dimensions (edges) contain the same number of lattice points. With this file splitting, the prefetching phase is then 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 split into ${D_T}^{t_T}{D_R}^{t_R} = \left(\frac{N}{M_T}\right)^{t_T}\left(\frac{N}{M_R}\right)^{t_R}$ disjoint equal-size subfiles, denoted by

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

in which the definition of the operator $\bigotimes$ is as follows. For $m \in \mathbb{Z}^+$ sets $\mathcal{A}_0, \mathcal{A}_1, \cdots, \mathcal{A}_{m-1}$, we define $\mathcal{A}_0 \bigotimes \mathcal{A}_1 \bigotimes \cdots \bigotimes \mathcal{A}_{m-1}$ as the set of all un-ordered elements in $\mathcal{A}_0 \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_{m-1}$, where $\times$ denotes the Cartesian product. We use $\{\cdot\}$ to convert a m-tuple to a set. For example, for a tuple $(1, 2, 3)$, we have $\{(1, 2, 3)\} = \{1, 2, 3\}$. Hence, $\mathcal{A}_0 \bigotimes \mathcal{A}_1 \bigotimes \cdots \bigotimes \mathcal{A}_{m-1} \triangleq \{\{A\} : A \in \mathcal{A}_0 \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_{m-1}\}$. The subfile $\mathcal{W}_{n,\mathcal{T},\mathcal{R}}$ is exclusively cached by a set of transmitters in $\mathcal{T}$ and a set of receivers in $\mathcal{R}$. Under this file splitting strategy, each transmitter Tx_i caches a set of subfiles $\{\mathcal{W}_{n,\mathcal{T},\mathcal{R}} : \forall\mathcal{T} : i \in \mathcal{T}, \forall\mathcal{R}, \forall n \in [N]\}$ and each receiver Rx_j caches a set of subfiles $\{\mathcal{W}_{n,\mathcal{T},\mathcal{R}} : \forall\mathcal{T}, \forall\mathcal{R} : j \in \mathcal{R}, \forall n \in [N]\}$. As a result,

[4] In the prefetching phase, each file is split into multiple smaller files and each of these smaller subfiles is then spread across the user caches. We use "subfile" to refer to these smaller subfiles. In the delivery phase, in order to perform interference cancellation, each subfile needs to be further split into multiple even smaller ones. We use "packet" to refer to such smaller files resulting from splitting the subfiles. So a packet is the smallest unit that a file is split into. To transmit these requested packets to the target receivers, random Gaussian encoding must be applied to these packets and the output is called *coded packets*. However, in the description of the general schemes, we just refer the coded packet to as packet for simplicity when there is no confusion.

[5] Since we apply the hypercube cache placement at both the transmitters' and receivers' sides, there are two hypercubes associated with the cache-aided interference network, including the *transmitter hypercube* which is a t_T-dimnesional hypercube with each edge containing N/M_T lattice points (transmitters), and the *receiver hypercube* which is a t_R-dimnesional hypercube with each edge containing N/M_R lattice points (receivers).

[6] The superscript "T" means "Transmitter". Readers should not confuse this with the transpose operator.

the number of subfiles cached by $\text{Tx}_i, i \in [K_T]$ is equal to $N{D_T}^{t_T-1}{D_R}^{t_R}$ and hence the number of packets cached by $\text{Tx}_i, i \in [K_T]$ is equal to

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

where $\frac{F}{{D_T}^{t_T}{D_R}^{t_R}}$ is the number of packets of each subfile (note that in the following delivery phase, each subfile needs to be further split into multiple packets). Similarly, the number of subfiles cached by $\text{Rx}_j, j \in [K_R]$ is equal to $N{D_T}^{t_T}{D_R}^{t_R-1}$ and hence the number of packets cached by $\text{Rx}_j, \forall j \in [K_R]$ is equal to

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

which also satisfies the memory constraint. The application of the hypercube cache placement method to cache-aided interference networks is illustrated via the following example.

Example 2 (Hypercube Cache Placement for Interference Networks): Consider a wireless network with $K_T = 4$ transmitters and $K_R = 4$ receivers. Each transmitter and receiver is equipped with a cache memory of size $M_T = 2$ and $M_R = 2$ files, respectively. The file library contains $N = 4$ files denoted by A, B, C and D. Hence, we have the parameters $D_T = N/M_T = 2, D_R = N/M_R = 2,\ t_T = K_T/D_T = 2$ and $t_R = K_R/D_R = 2$. In this case, both the transmitter and receiver hypercubes are two-dimensional hypercubes (i.e., squares) with each edge containing two transmitters/receivers.

In the prefetching phase, each file $\mathcal{W}_n$ is split into ${D_T}^{t_T}{D_R}^{t_R} = 16$ subfiles $\{\mathcal{W}_{n,\mathcal{T},\mathcal{R}}\}$ of equal sizes for any $\mathcal{T}, \mathcal{R} \in \{\{0,2\},\{0,3\},\{1,2\},\{1,3\}\}$ and. 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:[7]

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

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 splitting is done for files B, C and D. It can be seen that each transmitter caches 8 subfiles of each file. Since each subfile contains $F/16$ packets, the total number of packets cached by each transmitter is $4 \times 8 \times F/16 = 2F$, which satisfies the memory constraint of the transmitters. Similarly, the memory constraint of the receivers is also satisfied. △

III. Main Result

The main result on the one-shot linear sum-DoF using the hypercube cache placement approach is presented in this section. Note that when $\frac{K_T M_T + K_R M_R}{N} > K_R$, the sum-DoF K_R is always achievable by only utilizing a fraction of the Tx/Rx cache memories such that for the updated system with Tx/Rx cache memories $M'_T \le M_T$ and $M'_R \le M_R$, we have $\frac{K_T M'_T + K_R M'_R}{N} = K_R$. Therefore, by applying the proposed scheme on the updated system, the sum-DoF of K_R can be achieved. As a result, we focus on the case where $\frac{K_T M_T + K_R M_R}{N} \le K_R$.

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 $\delta \triangleq t_T/t_R \in \mathbb{Z}^+,\ D_R = N/M_R \ge \delta + 1$, in which $t_T \in [1 : K_T], t_R \in [K_R], D_R \in \mathbb{Z}^+$, the one-shot linear sum-DoF of $\frac{K_T M_T + K_R M_R}{N}$ is achievable when $K_R \ge \frac{K_T M_T + K_R M_R}{N}$ with

$$F = \left(\frac{N}{M_T}\right)^{t_T} \left(\frac{N}{M_R}\right)^{t_R} \binom{D_R - 2}{\delta - 1} \binom{D_R - 1}{\delta}^{t_R - 1} \frac{(\delta!)^{t_R}}{\delta} (t_R - 1)! \tag{10}$$

Proof: The achievability of Theorem 1 is proved by the general achievable scheme described in Section IV-C, which focuses on the case $K_R \ge \frac{M_T K_T + M_R K_R}{N}$. The converse results follows directly from [7] which will not be presented in this paper. ■

The implications of Theorem 1 are two-fold, which includes the optimality of the achievable one-shot linear DoF and the reduced subpacketization level. Note that if either t_T or t_R is not an integer, or both of them are not integers, we can still achieve the sum-DoF of $t_T + t_R$ for any values of t_T and t_R using the *memory-sharing* method in [3] which will be briefly introduced later. The following observations are ready.

A. Sum-DoF Optimality

As shown in [7], when $K_R \ge \frac{K_T M_T + K_R M_R}{N}$, the optimal one-shot linear sum-DoF of the interference network studied in this paper, $\mathsf{DoF}^*_{\text{L,sum}}$, over any possible cache placement realizations, is bounded by $\frac{K_T M_T + K_R M_R}{N} \le \mathsf{DoF}^*_{\text{L,sum}} \le \frac{2(K_T M_T + K_R M_R)}{N}$, which implies that when $\frac{K_T M_T + K_R M_R}{N} \le K_R$, the achievable one-shot linear sum-DoF under the hypercube cache placement is equal to the achievable one-shot linear DoF in [7] and is within a factor of 2 to the optimal one-shot linear sum-DoF of the network. This result indeed shows that the DoF of $\frac{M_T K_T + M_R K_R}{N}$ can be achieved by different cache placement methods, which provides the potential to reduce the total number of packets required. In addition, to see how much DoF gain can be obtained going beyond one-shot linear transmissions, we refer the readers to Section VI of [7] where the scaling law of the optimal sum-DoF is analyzed. In particular, for the cases of large number of transmitters and receivers ($K_T = K_R = K \to \infty$ and other parameters are fixed) and constant number of transmitters ($K_T = C, K_R = K \to \infty$), the one-shot linear scheme achieves the same DoF scaling as the interference alignment alike schemes. This implies that interference alignment alike multi-shot schemes can only provide constant DoF gain over the one-shot linear schemes which has much lower complexity.

B. Subpacketization Level Reduction

Under the hypercube cache placement strategy, the number of packets per file, i.e., F, required for implementing the

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

interference cancellation in the delivery phase is significantly reduced compared to the NMA scheme. In particular, the NMA scheme requires to split each file into $\binom{K_T}{t_T}\binom{K_R}{t_R}$ subfiles in the prefetching phase and further split each subfile into $\frac{t_R![K_R-(t_R+1)]!}{[K_R-(t_T+t_R)]!}$ packets in the delivery phase. However, if we employ the hypercube cache placement strategy, each file is going to be split into $(\frac{N}{M_T})^{t_T}(\frac{N}{M_R})^{t_R}$ subfiles in the prefetching phase, and is further split into $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$ packets[8] in the delivery phase. In Section IV-D, we will show that for any system parameters, the hypercube scheme requires less number of packets than the NMA scheme and the gain of subpacketization can be unbounded with the increase of the cache sizes of transmitters and receivers. Together with the sum-DoF optimality, the hypercube based scheme can achieve the same one-shot linear DoF as in [7] while requiring a significantly smaller F.

C. Non-Integer Caching Parameters t_T and t_R

When the caching parameters $t_T = \frac{K_T M_T}{N}$ and/or $t_R = \frac{K_R M_R}{N}$ are not integers, we can still achieve the one-shot linear sum-DoF of $t_T + t_R$ using the *memory-sharing* method of [3]. More specifically, we can split the Tx/Rx memories and files proportionally so that for each of the new partitions, our proposed scheme can be applied for the updated parameters ${t_T}'$ and ${t_R}'$ which are integers. That is, for each new partition of memories and files, it can be treated as a new interference network with updated Tx/Rx cache memories M_T', M_R', file size F' and the corresponding caching parameters $t_T' = \frac{K_T M_T'}{N} \in \mathbb{Z}^+, t_R' = \frac{K_R M_R'}{N} \in \mathbb{Z}^+$, where the proposed scheme can be directly applied.

D. Non-Integer Values of δ

Although in Theorem 1 we have assumed that $\delta = t_T/t_R \in \mathbb{Z}^+$, the sum-DoF of t_T+t_R can also be achieved even when δ is not an integer. This can be done following a similar method to memory sharing. Note that $\delta \geq 1/t_R$ due to $t_T \geq 1$ since the file library has to be stored at least once by all the transmitters otherwise the receivers' demands can not be satisfied. Now we consider the case when both t_T and t_R are positive integers but $\delta = t_T/t_R$ is not an integer. The scheme to achieve the sum-DoF $t_T + t_R$ is described as follows.

We can split the Tx/Rx cache memories and the files proportionally such that the updated caching parameters t_T' and t_T'' of each partition correspond to δ' and δ'' both of which are integers. More specifically, each Tx memory is split into two parts $M_T' = \alpha M_T$ and $M_T'' = (1-\alpha)M_T$ for some $0 < \alpha < 1$, each Rx memory is split into two parts $M_R' = \beta M_R$ and $M_R'' = (1-\beta)M_R$ for some $0 < \beta < 1$, and each file $\mathcal{W}_n$ is split into two parts $\mathcal{W}_n = (\mathcal{W}_n', \mathcal{W}_n'')$ where $|\mathcal{W}_n'| = \gamma|\mathcal{W}_n|$ and $|\mathcal{W}_n''| = (1-\gamma)|\mathcal{W}_n|$ for some $0 < \gamma < 1$. We then apply the proposed scheme on the two Tx/Rx memory and file partitions $\left(M_T', M_R', \{\mathcal{W}_n'\}_{n\in[N]}, t_T', t_R'\right)$ and $\left(M_T'', M_R'', \{\mathcal{W}_n''\}_{n\in[N]}, t_T'', t_R''\right)$ where $\delta' = t_T'/t_R'$ and $\delta'' = t_T''/t_R''$ are both integers. WLOG, we let $\beta = \gamma$. Therefore, we have $t_T' = \frac{K_T M_T'}{N} = \frac{\alpha}{\gamma} t_T, t_R' = \frac{K_R M_R'}{N} = t_R$ and $t_T'' = \frac{K_T M_T''}{N} = \frac{1-\alpha}{1-\gamma} t_T, t_R'' = \frac{K_R M_R''}{N} = t_R$ and $t_T = p t_T' + (1-p) t_T'', \delta = p\delta' + (1-p)\delta''$ where $p = \frac{t_T''-t_T}{t_T''-t_T'}$. Next we consider two different cases: *1)* $\delta \in [1/t_R, 1)$, and *2)* $\delta \in (q, q+1)$ for some $q \in \mathbb{Z}^+$.

- *Case 1*: $\delta \in [1/t_R, 1)$. Let $t_T' = 1, t_T'' = t_R$. For the first memory and file partition, the coded caching scheme [3] can be applied to achieve the sum-DoF of $t_R' + 1 = t_R + 1$; For the second memory and file partition, we can apply the proposed scheme with $\delta'' = 1$ to achieve the sum-DoF $t_T'' + t_R'' = 2t_R$. As a result, the overall sum-DoF of $p(t_R'+1) + (1-p)(t_T''+t_R'') = t_T + t_R$ can be achieved.
- *Case 2*: $\delta \in (q, q+1)$ for some $q \in \mathbb{Z}^+$. Let $t_T' = q t_R = \lfloor\delta\rfloor t_R$ and $t_T'' = (q+1)t_R = \lceil\delta\rceil t_R$. For the first and second memory and file partitions with $\delta' = \frac{t_T'}{t_R'} = \lfloor\delta\rfloor$ and $\delta'' = \frac{t_T''}{t_R''} = \lceil\delta\rceil$, the proposed scheme can be directly applied to achieve the sum-DoF of $t_T' + t_R' = (\lfloor\delta\rfloor + 1)t_R$ and $t_T'' + t_R'' = (\lceil\delta\rceil + 1)t_R$ respectively. Therefore, the overall sum-DoF $p(t_T'+t_R') + (1-p)(t_T''+t_R'') = t_T + t_R$ can be achieved. In both cases, let F' and F'' be the required number of packets per file over the two memory and file partitions which can be calculated by Eq. (21). Then the number of packets per file is determined as $F = F' + F''$.

IV. Achievable Delivery Scheme

A. An Example

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

Example 3 (Achievable Delivery Scheme): We consider the same network setting as Example 2. Let receiver Rx_j request the file $\mathcal{W}_{d_j}$. Without loss of generality, we assume that $\mathcal{W}_{d_0} = A, \mathcal{W}_{d_1} = B, \mathcal{W}_{d_2} = C$ and $\mathcal{W}_{d_3} = D$. In the prefetching phase, each receiver has already cached 8 subfiles of its requested file. Therefore, the transmitters only need to deliver the $16 - 8 = 8$ remaining subfiles to each receiver. In particular, the following 32 subfiles need to be delivered to the corresponding receivers:

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

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

[8] Here we have implicitly assumed that $D_R - 2 \geq \delta - 1$, i.e., $D_R \geq \delta + 1$. This assumption can be justified as follows. In real-world wireless networks, the number of receivers (users) K_R can be larger than the number of transmitters (base stations, BS) K_T, since each BS can be associated with multiple users. However, each BS can have much larger cache memory than the users, i.e., $M_T \gg M_R$. Due to the large per-BS cache memory M_T but relatively small K_T at the transmitter's side and the smaller per-user cache memory M_R but larger K_R at the users' sides, it is reasonable to assume that the caching parameters t_T and t_R are close to each other. Also, since each user's cache memory is very small compared to the file library, i.e., $M_R/N \ll 1$, then $D_R = N/M_R \gg 1$. For example, consider a network with $N = 500$ files each having size 5 GB (e.g., Netflix movies), $K_T = 5$ BSs each capable of caching $M_T = 200$ files (i.e., 1000 GB memory per BS), and $K_R = 50$ receivers each capable of caching $M_R = 10$ files (50 GB memory per receiver). In this case, we have $t_T = K_T M_T/N = 5 \times 200/500 = 2, t_R = K_R M_R/N = 50 \times 10/500 = 1$ and therefore $\delta = 2$. Moreover, we have $D_R = N/M_R = 500/10 = 50$ which is much larger than $\delta + 1 = 3$. As a result, it can be seen that the assumption $D_R \geq \delta + 1$ is valid in practice.

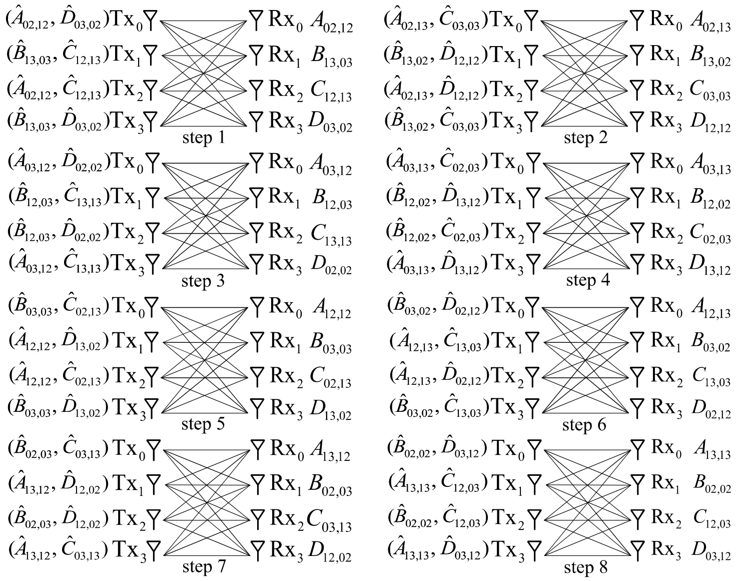

$(\hat{A}_{02,12},\hat{D}_{03,02})\mathrm{Tx}_0$
$(\hat{B}_{13,03},\hat{C}_{12,13})\mathrm{Tx}_1$
$(\hat{A}_{02,12},\hat{C}_{12,13})\mathrm{Tx}_2$
$(\hat{B}_{13,03},\hat{D}_{03,02})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{02,12}$
$\mathrm{Rx}_1\ B_{13,03}$
$\mathrm{Rx}_2\ C_{12,13}$
$\mathrm{Rx}_3\ D_{03,02}$
step 1
$(\hat{A}_{02,13},\hat{C}_{03,03})\mathrm{Tx}_0$
$(\hat{B}_{13,02},\hat{D}_{12,12})\mathrm{Tx}_1$
$(\hat{A}_{02,13},\hat{D}_{12,12})\mathrm{Tx}_2$
$(\hat{B}_{13,02},\hat{C}_{03,03})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{02,13}$
$\mathrm{Rx}_1\ B_{13,02}$
$\mathrm{Rx}_2\ C_{03,03}$
$\mathrm{Rx}_3\ D_{12,12}$
step 2
$(\hat{A}_{03,12},\hat{D}_{02,02})\mathrm{Tx}_0$
$(\hat{B}_{12,03},\hat{C}_{13,13})\mathrm{Tx}_1$
$(\hat{B}_{12,03},\hat{D}_{02,02})\mathrm{Tx}_2$
$(\hat{A}_{03,12},\hat{C}_{13,13})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{03,12}$
$\mathrm{Rx}_1\ B_{12,03}$
$\mathrm{Rx}_2\ C_{13,13}$
$\mathrm{Rx}_3\ D_{02,02}$
step 3
$(\hat{A}_{03,13},\hat{C}_{02,03})\mathrm{Tx}_0$
$(\hat{B}_{12,02},\hat{D}_{13,12})\mathrm{Tx}_1$
$(\hat{B}_{12,02},\hat{C}_{02,03})\mathrm{Tx}_2$
$(\hat{A}_{03,13},\hat{D}_{13,12})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{03,13}$
$\mathrm{Rx}_1\ B_{12,02}$
$\mathrm{Rx}_2\ C_{02,03}$
$\mathrm{Rx}_3\ D_{13,12}$
step 4
$(\hat{B}_{03,03},\hat{C}_{02,13})\mathrm{Tx}_0$
$(\hat{A}_{12,12},\hat{D}_{13,02})\mathrm{Tx}_1$
$(\hat{A}_{12,12},\hat{C}_{02,13})\mathrm{Tx}_2$
$(\hat{B}_{03,03},\hat{D}_{13,02})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{12,12}$
$\mathrm{Rx}_1\ B_{03,03}$
$\mathrm{Rx}_2\ C_{02,13}$
$\mathrm{Rx}_3\ D_{13,02}$
step 5
$(\hat{B}_{03,02},\hat{D}_{02,12})\mathrm{Tx}_0$
$(\hat{A}_{12,13},\hat{C}_{13,03})\mathrm{Tx}_1$
$(\hat{A}_{12,13},\hat{D}_{02,12})\mathrm{Tx}_2$
$(\hat{B}_{03,02},\hat{C}_{13,03})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{12,13}$
$\mathrm{Rx}_1\ B_{03,02}$
$\mathrm{Rx}_2\ C_{13,03}$
$\mathrm{Rx}_3\ D_{02,12}$
step 6
$(\hat{B}_{02,03},\hat{C}_{03,13})\mathrm{Tx}_0$
$(\hat{A}_{13,12},\hat{D}_{12,02})\mathrm{Tx}_1$
$(\hat{B}_{02,03},\hat{D}_{12,02})\mathrm{Tx}_2$
$(\hat{A}_{13,12},\hat{C}_{03,13})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{13,12}$
$\mathrm{Rx}_1\ B_{02,03}$
$\mathrm{Rx}_2\ C_{03,13}$
$\mathrm{Rx}_3\ D_{12,02}$
step 7
$(\hat{B}_{02,02},\hat{D}_{03,12})\mathrm{Tx}_0$
$(\hat{A}_{13,13},\hat{C}_{12,03})\mathrm{Tx}_1$
$(\hat{B}_{02,02},\hat{C}_{12,03})\mathrm{Tx}_2$
$(\hat{A}_{13,13},\hat{D}_{03,12})\mathrm{Tx}_3$
$\mathrm{Rx}_0\ A_{13,13}$
$\mathrm{Rx}_1\ B_{02,02}$
$\mathrm{Rx}_2\ C_{12,03}$
$\mathrm{Rx}_3\ D_{03,12}$
step 8

Fig. 3. Delivery phase for 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. There are in total 8 communication steps and in each of which 4 different packets are delivered to the receivers interference-free.

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

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

Note that in the hypercube-based delivery scheme, each subfile needs to be further split into $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$ packets. In this example, since $\delta = \frac{t_T}{t_R} = 1$, $D_T = t_T = D_R = t_R = 2$, $\delta = 1$, we have $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)! = \binom{0}{0}\binom{1}{1}(2-1)! = 1$, implying that no further file splitting is needed and thus 32 packets will be delivered.

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 delivered simultaneously to the receivers without interference. Fig. 3 shows how the 32 packets to be delivered are grouped and transmitted. In each communication step, $t_T + t_R = 4$ packets are delivered to the receivers simultaneously, and the interference among different users can be effectively eliminated by choosing proper linear combination coefficients at the $t_T + t_R = 4$ transmitters. For example, in step 1 of Fig. 3, four packets $A_{02,12}, B_{13,03}, C_{12,13}$ and $D_{03,02}$ are delivered to receivers Rx_0, Rx_1, Rx_2 and Rx_3 respectively. We write the transmitted signals $S_i, i \in [4]$ of each transmitter Tx_i as a linear combination of a subset of these four 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, due to the careful choice of the linear coefficients, some interference terms are canceled over the air by zero forcing (e.g., $\hat{C}_{12,13}$ is canceled at Rx_0). The corresponding received signals by Rx_0, Rx_1, Rx_2 and Rx_3 after zero forcing are given by

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

where $N_i, i \in [4]$ represents the Gaussian noise.

We can see that receiver Rx_0 can cancel the interference caused by $B_{13,03}$ and $D_{03,02}$ since these two packets have already been cached by Rx_0 and the desired packet $A_{02,12}$ can be successfully decoded by subtracting the undesired but prefetched packets. 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 that can be nulled out by cached packets can be eliminated so that all receivers can decode their desired packets. It can be verified that there exist such linear combinations and all receivers can decode their desired packets in all remaining 7 communication steps. Hence, the 32 packets, each consisting of $|\mathcal{W}_n|/16$ bits, can be delivered to the receivers in 8 communication steps, each containing $F/16 = 1$ resource block. As a result, a sum-DoF of $\frac{K_T M_T + K_R M_R}{N} = 4$ can be achieved. Hence, the proposed file subpacketization, cache 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. △

B. Hypercube Permutation

Before we proceed to the description of the general achievable scheme, we introduce two definitions of special permutations on a given set of points, i.e., the *hypercube permutation* and *circular hypercube permutation*, which are essential to the description of the general delivery phase.

Definition 1 (Hypercube Permutation): Given a set of $D \times t$ points, denoted by $\mathcal{Q}$, i.e., $|\mathcal{Q}| = Dt$, 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 are partitioned 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 a *hypercube permutation* of the set $\mathcal{Q}$, denoted by $\pi^{\text{HCB}} = [\,\pi(0) \ \ \pi(1) \ \ \cdots \ \ \pi(Dt-1)\,]$, as such a permutation of the Dt points that satisfies the following condition: For any set of points $\mathcal{U}_i, i \in [t]$, 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 $j_1, j_2 \in [D]$. ♢

Definition 2 (Circular Hypercube Permutation): A *circular permutation* of a set $\mathcal{Q}$ is a way of arranging the elements of $\mathcal{Q}$ such that these arrangements are invariant of circular shifts. Denote the set of circular permutations of $\mathcal{Q}$ as $\Pi_{\mathcal{Q}}^{\text{circ}}$. For example, if $\mathcal{Q} = \{1, 2, 3\}$, then $\Pi_{\mathcal{Q}}^{\text{circ}} = \{[1\ 2\ 3\ ], [1\ 3\ 2\ ]\}$. A *circular hypercube permutation* of a set $\mathcal{Q}$ is a way of arranging the elements of $\mathcal{Q}$ which are invariant of circular shifts, and meanwhile, the corresponding arrangement should be a hypercube permutation. ♢

We illustrate the concept of hypercube permutation and circular hypercube permutation via 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

$$\Pi_{\mathcal{Q}}^{\text{HCB}} = \big\{[\,0\ 2\ 1\ 3\,], [\,0\ 3\ 1\ 2\,], [\,1\ 2\ 0\ 3\,], [\,1\ 3\ 0\ 2\,], [\,2\ 1\ 3\ 0\,], [\,2\ 0\ 3\ 1\,], [\,3\ 1\ 2\ 0\,], [\,3\ 0\ 2\ 1\,]\big\}. \tag{11}$$

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 $\Pi_{\mathcal{Q}}^{\text{HCB,circ}} = \big\{[\,0\ 2\ 1\ 3\,], [\,0\ 3\ 1\ 2\,]\big\}$. △

Lemma 1: For a set of points (users) $\mathcal{Q}$ of dimension t and D points (users) in each dimension, denote the set of all hypercube permutations as $\Pi_{\mathcal{Q}}^{\text{HCB}}$, then $\left|\Pi_{\mathcal{Q}}^{\text{HCB}}\right| = (D!)^t(t)!$. The set of circular hypercube permutations of $\mathcal{Q}$, denoted by $\Pi_{\mathcal{Q}}^{\text{HCB,circ}}$, has size $\left|\Pi_{\mathcal{Q}}^{\text{HCB,circ}}\right| = \frac{(D!)^t(t-1)!}{D}$.

Proof: See Appendix A. ■

C. General Achievable Scheme

In this section, we present the general achievable scheme which is formally described in Algorithm 1. Recall that $t_T = \frac{K_T M_T}{N}$ and $t_R = \frac{K_R M_R}{N}$, and we assume $t_T, t_R \in \mathbb{Z}^+$, $\frac{M_T K_T + M_R K_R}{N} \leq K_R$. In this paper, we focus on the case $\delta \triangleq \frac{t_T}{t_R} \in \mathbb{Z}^+$, implying that $t_T \geq 1$.

Algorithm 1 General Hypercube-Based Achievable Scheme

```
*Prefetching Phase:*
for $i = 0, 1, \cdots, K_T - 1$ do
    Group Tx$_i$ into the transmitter dimension $\mathcal{U}_j^{\text{T}}$, where $j = \lfloor \frac{i}{D_T} \rfloor$.
end for
for $i = 0, 1, \cdots, K_R - 1$ do
    Group Rx$_i$ into the receiver label set $\mathcal{U}_j^{\text{R}}$, where $j = \lfloor \frac{i}{D_R} \rfloor$.
end for
for $n = 0, 1, \cdots, N - 1$ do
    Split $\mathcal{W}_n$ into $\left(\frac{N}{M_T}\right)^{t_T} \left(\frac{N}{M_R}\right)^{t_R}$ disjoint equal-size subfiles:
        $$\mathcal{W}_n = \{\mathcal{W}_{n,\mathcal{T},\mathcal{R}}\}_{\substack{\mathcal{T} \in \mathcal{U}_0^{\text{T}} \bigotimes \mathcal{U}_1^{\text{T}} \bigotimes \cdots \bigotimes \mathcal{U}_{t_T-1}^{\text{T}} \\ \mathcal{R} \in \mathcal{U}_0^{\text{R}} \bigotimes \mathcal{U}_1^{\text{R}} \bigotimes \cdots \bigotimes \mathcal{U}_{t_R-1}^{\text{R}}}}.$$
end for
for $i = 0, 1, \cdots, K_T - 1$ do
    Tx$_i$ caches $\{\mathcal{W}_{n,\mathcal{T},\mathcal{R}} : i \in \mathcal{T}\}$ for all $n \in [N]$.
end for
for $j = 0, 1, \cdots, K_R - 1$ do
    Rx$_j$ caches $\{\mathcal{W}_{n,\mathcal{T},\mathcal{R}} : j \in \mathcal{R}\}$ for all $n \in [N]$.
end for

*Delivery Phase:*
for $j = 0, 1, \cdots, K_R - 1$ do
    for $\mathcal{T} \in \mathcal{U}_0^{\text{T}} \bigotimes \mathcal{U}_1^{\text{T}} \bigotimes \cdots \bigotimes \mathcal{U}_{t_T-1}^{\text{T}}$ do
        for $\mathcal{R} \in \mathcal{U}_0^{\text{R}} \bigotimes \mathcal{U}_1^{\text{R}} \bigotimes \cdots \bigotimes \mathcal{U}_{\lfloor \frac{j}{D_R} \rfloor}^{\text{R}} \setminus \{j\} \bigotimes \cdots \bigotimes \mathcal{U}_{t_R-1}^{\text{R}}$ do
            Split the subfile $\mathcal{W}_{d_j,\mathcal{T},\mathcal{R}}$ into $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1} \frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$ disjoint packets of eqaul-sizes:
                $$\left\{\mathcal{W}_{d_j,\mathcal{T},\dot{\pi},\ddot{\pi}}\right\}_{\substack{\dot{\pi}=\pi[1:t_R] \\ \ddot{\pi}=\pi[t_R+1:t_T+t_R-1] \\ \pi \in \Pi_{\mathcal{Q}^{\text{U}}}^{\text{HCB}},\, \pi(0)=j,\, \pi(t_R)=r_{\lfloor \frac{j}{D_R} \rfloor} \\ \{\pi(1),\pi(2),\cdots,\pi(t_R-1)\}=\mathcal{R}\setminus\{r_{\lfloor \frac{j}{D_R} \rfloor}\}}}$$
            where $\mathcal{Q} \in \Gamma_{\mathcal{U}_0^{\text{R}},\delta+1} \bigotimes \cdots \bigotimes \Gamma_{\mathcal{U}_{\lfloor \frac{j}{D_R} \rfloor}^{\text{R}},\delta+1} \bigotimes \cdots \bigotimes \Gamma_{\mathcal{U}_{t_R-1}^{\text{R}},\delta+1}$.
        end for
    end for
end for
for $\mathcal{T} \in \mathcal{U}_0^{\text{T}} \bigotimes \mathcal{U}_1^{\text{T}} \bigotimes \cdots \bigotimes \mathcal{U}_{t_T-1}^{\text{T}}$ do
    for $\mathcal{R} \in \Gamma_{\mathcal{U}_0^{\text{R}},\delta+1} \bigotimes \Gamma_{\mathcal{U}_1^{\text{R}},\delta+1} \bigotimes \cdots \bigotimes \Gamma_{\mathcal{U}_{t_R-1}^{\text{R}},\delta+1}$ do
        for $\pi \in \Pi_{\mathcal{R}^{\text{U}}}^{\text{HCB,circ}}$ do
            Each transmitter sends a linear combination (Lemma 3) of the coded packets:
                $$S_i = \mathcal{L}_{i,\mathcal{T},\pi}\left(\left\{\hat{W}_{d_{\pi(\ell)},\mathcal{T}(\ell),\pi[\ell+1:\ell+t_R],\pi[\ell+t_R+1:\ell+t_R+t_T-1]} : \ell \in [t_T + t_R], i \in \mathcal{T}(\ell)\right\}\right)$$
        end for
    end for
end for
```

The corresponding prefetching and delivery phases are described as follows.

1) Prefetching Phase: The hypercube cache placement is employed at both the transmitters' and receivers' sides in the prefetching phase. Refer to Section II-B.2 for detailed descriptions.

2) Delivery Phase: In the delivery phase, the receivers' demand vector $\mathbf{d} = [d_0, d_1, \cdots, d_{K_R-1}]$ is revealed, i.e., each receiver $\text{Rx}_j, j \in [K_R]$ requests a 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., $\{\mathcal{W}_{d_j,\mathcal{T}}, \forall \mathcal{T}, \forall \mathcal{R} : j \notin \mathcal{R}\}$.

Following a similar methodology of [7], we need to further split the set of subfiles to be delivered to the receivers into packets so that they can be scheduled in subsets of size $t_T + t_R$ and delivered to the receivers simultaneously without interference. In particular, for any packet in the subset of $t_T + t_R$ packets, it is requested by one particular receiver and can be cancelled by another t_R receivers by utilizing their cached packets. Also, the transmitters can collaborate to zero-force the the interference to another $t_T - 1$ unintended receivers. We describe how to do such a further splitting based on the hypercube cache placement in the following.

For any $j \in [K_R]$, $\mathcal{T} = \{(\tau_0, \tau_1, \cdots, \tau_{t_T-1})\}$ with $(\tau_0, \tau_1, \cdots, \tau_{t_T-1}) \in \mathcal{U}_0^{\text{T}} \times \mathcal{U}_1^{\text{T}} \times \cdots \times \mathcal{U}_{t_T-1}^{\text{T}}$, and $\mathcal{R} = \{(r_0, r_1, \cdots, r_{t_T-1})\}$ with $(r_0, r_1, \cdots, r_{t_T-1}) \in \mathcal{U}_0^{\text{R}} \times \mathcal{U}_1^{\text{R}} \times \cdots \times \mathcal{U}^{\text{R}}_{\lfloor \frac{j}{D_R} \rfloor} \backslash \{j\} \times \cdots \times \mathcal{U}_{t_R-1}^{\text{R}}$ (note that $|\mathcal{T}| = t_T$ and $|\mathcal{R}| = t_R$), we split $\mathcal{W}_{d_j,\mathcal{T},\mathcal{R}}$ into $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$ disjoint packets of equal-sizes, denoted by

$$\left\{\mathcal{W}_{d_j,\mathcal{T},\dot{\pi},\ddot{\pi}}\right\}_{\substack{\dot{\pi}=\pi[1:t_R] \\ \ddot{\pi}=\pi[t_R+1:t_T+t_R-1] \\ \pi\in\Pi^{\text{HCB}}_{\mathcal{Q}^{\text{U}}},\, \pi(0)=j,\, \pi(t_R)=r_{\lfloor \frac{j}{D_R} \rfloor} \\ \{\pi(1),\pi(2),\cdots,\pi(t_R-1)\}=\mathcal{R}\backslash\{r_{\lfloor \frac{j}{D_R} \rfloor}\}}}, \tag{12}$$

where $\mathcal{Q} \in \Gamma_{\mathcal{U}_0^{\text{R}},\delta+1} \bigotimes \cdots \bigotimes \Gamma_{\mathcal{U}^{\text{R}}_{\lfloor \frac{j}{D_R} \rfloor},\delta+1} \bigotimes \cdots \bigotimes \Gamma_{\mathcal{U}^{\text{R}}_{t_R-1},\delta+1}$ and the notations are defined as follows. For a set $\mathcal{S}$, $\Gamma_{\mathcal{S},s}$ is defined as a set whose elements are all subsets of $\mathcal{S}$ with size s, i.e., $\Gamma_{\mathcal{S},s} = \{\mathcal{A} : \mathcal{A} \subseteq \mathcal{S}, |\mathcal{A}| = s\}, \forall s \in [1 : |\mathcal{S}|]$. For example, for $\mathcal{S} = \{0, 1, 2\}$, we have $\Gamma_{\mathcal{S},2} = \{\{0, 1\}, \{1, 2\}, \{0, 2\}\}$. For a set $\mathcal{Q}$ whose elements are sets, $\mathcal{Q}^{\text{U}}$ denotes the union of the elements in $\mathcal{Q}$. For example, if $\mathcal{Q} = \{\{0, 1\}, \{2, 3\}\}$, we have $\mathcal{Q}^{\text{U}} = \{0, 1\} \cup \{2, 3\} = \{0, 1, 2, 3\}$. Moreover, for a set $\mathcal{S}$, and a hypercube permutation $\pi \in \Pi^{\text{HCB}}_{\mathcal{S}}$ and two integers i, j, where $i \leq j$, $\pi[i : j]$ is defined as $\pi[i : j] = [\pi(i \oplus_{|\mathcal{S}|} 0),\ \pi(i \oplus_{|\mathcal{S}|} 1),\ \cdots,\ \pi(i \oplus_{|\mathcal{S}|} (j - i))]$, in which for two integers m, n, $m \oplus_{|\mathcal{S}|} n$ is defined as

$$m \oplus_{|\mathcal{S}|} n = 1 + (m + n - 1 \mod |\mathcal{S}|). \tag{13}$$

After such a further splitting, for a specific set of $t_T + t_R$ receivers and a corresponding hypercube permutation π, the packet $\mathcal{W}_{d_j,\mathcal{T},\dot{\pi},\ddot{\pi}}$, which is desired by Rx_j, can be cancelled at receivers in $\dot{\pi}$ by utilizing their individual cached contents and can be zero-forced at receivers in $\ddot{\pi}$ through the collaboration of some transmitters. Lemma 2 shows how this further splitting is done. For a set $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{t_T-1}\}$ whose elements are from the t_T different transmitter dimensions, i.e., $\tau_i \in \mathcal{U}_i^{\text{T}}, i \in [t_T]$, we define the corresponding sets $\mathcal{T}(\ell) \triangleq \left\{\tau_0^{(\ell)}, \tau_1^{(\ell)}, \cdots, \tau_{t_T-1}^{(\ell)}\right\}, \ell \in [t_T + t_R]$, where $\mathcal{T}(0) = \mathcal{T}$, i.e., $\tau_i^{(0)} = \tau_i, \forall i \in [t_T]$ and

- when $1 \leq \ell \leq t_T$,

$$\tau_i^{(\ell)} = \begin{cases} \tau_i^{(0)} + 1 \mod D_T & 0 \leq i \leq \ell - 1, \\ \tau_i^{(0)} & \ell \leq i \leq t_T - 1. \end{cases} \tag{14}$$

- when $t_T + 1 \leq \ell \leq t_T + t_R - 1$,

$$\tau_i^{(\ell)} = \begin{cases} \tau_i^{(0)} & 0 \leq i \leq \ell - t_T - 1, \\ \tau_i^{(0)} + 1 \mod D_T & \ell - t_T \leq i \leq t_T - 1. \end{cases} \tag{15}$$

Lemma 2: Based on the hypercube cache placement, for any receivers' demand vector $\mathbf{d}$*, the set of packets needed to be sent to the receivers can be grouped into disjoint subsets of size* $t_T + t_R$ *as*

$$\bigcup_{\substack{\mathcal{T}\in\mathcal{U}_0^{\text{T}} \bigotimes \mathcal{U}_1^{\text{T}} \bigotimes \cdots \bigotimes \mathcal{U}_{t_T-1}^{\text{T}} \\ \mathcal{R}\in\Gamma_{\mathcal{U}_0^{\text{R}},\delta+1} \bigotimes \Gamma_{\mathcal{U}_1^{\text{R}},\delta+1} \bigotimes \cdots \bigotimes \Gamma_{\mathcal{U}_{t_R-1}^{\text{R}},\delta+1} \\ \pi\in\Pi^{\text{HCB,circ}}_{\mathcal{R}^{\text{U}}}}} \times \Big\{\mathcal{W}_{d_{\pi(\ell)},\mathcal{T}(\ell),\pi[\ell+1:\ell+t_R],\pi[\ell+t_R+1:\ell+t_R+t_T-1]} : \ell \in [t_T + t_R]\Big\}. \tag{16}$$

Proof: See Appendix B. ■

Given the grouping method of the packets in Lemma 2, we will have $D_T^{t_T}\binom{D_R}{\delta+1}^{t_R}\frac{[(\delta+1)!]^{t_R}(t_R-1)!}{\delta+1}$ (using Lemma 1) steps of communications. More specifically, the term ${D_T}^{t_T}$ corresponds to the number of possible choices of $\mathcal{T}$, $\binom{D_R}{\delta+1}^{t_R}$ corresponds to the number of choices of $\mathcal{R}$. We also have $\left|\Pi^{\text{HCB,circ}}_{\mathcal{R}^{\text{U}}}\right| = \frac{[(\delta+1)!]^{t_R}(t_R-1)!}{\delta+1}$ which is a direct result of Lemma 1, i.e., the number of different hypercube permutations of the set $\mathcal{R}^{\text{U}}$ partitioned into $t = t_R$ dimensions and $D = \delta+1$ points in each dimension. In each of these communication steps, specific sets $\mathcal{T}$ and $\mathcal{R}$ and a hypercube permutation are fixed, and each transmitter $\text{Tx}_i, i \in \mathcal{T}(\ell)$ transmits a linear combination of the coded packets, i.e.,

$$S_i = \mathcal{L}_{i,\mathcal{T},\pi}\Big(\Big\{\hat{W}_{d_{\pi(\ell)},\mathcal{T}(\ell),\pi[\ell+1:\ell+t_R],\pi[\ell+t_R+1:\ell+t_R+t_T-1]} : \ell \in [t_T + t_R], i \in \mathcal{T}(\ell)\Big\}\Big), \tag{17}$$

in which for any packet $\mathcal{W}_{d_j,\mathcal{T},\dot{\pi},\ddot{\pi}}$, $\hat{W}_{d_j,\mathcal{T},\dot{\pi},\ddot{\pi}}$ denotes its Gaussian coded version, and $\mathcal{L}_{i,\mathcal{T},\pi}(.)$ represents the linear combination that Tx_i chooses to transmit set of packets in (17).

The following lemma shows the existence of the linear combination coefficients.

Lemma 3: For any subset of t_T *transmitters* $\mathcal{T} \in \mathcal{U}_0^{\text{T}} \bigotimes \mathcal{U}_1^{\text{T}} \bigotimes \cdots \bigotimes \mathcal{U}_{t_T-1}^{\text{T}}$*, any set of* $t_T + t_R$ *receivers* $\mathcal{R}^{\text{U}}$ *for which* $\mathcal{R} \in \Gamma_{\mathcal{U}_0^{\text{R}},\delta+1} \bigotimes \Gamma_{\mathcal{U}_1^{\text{R}},\delta+1} \bigotimes \cdots \bigotimes \Gamma_{\mathcal{U}_{t_R-1}^{\text{R}},\delta+1}$*, and any circular hypercube permutation* $\pi \in \Pi^{\text{HCB,circ}}_{\mathcal{R}^{\text{U}}}$*, there exists a choice of the linear combinations* $\{\mathcal{L}_{i,\mathcal{T},\pi}(.)\}_{i=1}^{K_T}$ *in (17) such that the set of* $t_T + t_R$ *packets in*

$$\Big\{\mathcal{W}_{d_{\pi(\ell)},\mathcal{T}(\ell),\pi[\ell+1:\ell+t_R],\pi[\ell+t_R+1:\ell+t_R+t_T-1]} : \ell \in [t_T+t_R]\Big\} \tag{18}$$

can be delivered simultaneously without interference by the transmitters in $\bigcup_{\ell \in [t_T + t_R]} \mathcal{T}(\ell)$ to the receivers in $\mathcal{R}^{\mathrm{U}}$.

Proof: The proof of Lemma 3 follows exactly the same steps given in [7]. To show the existence of such linear combinations, we require the linear coefficients to be designed such that for any receiver in $\mathcal{R}^{\mathrm{U}}$, its desired packets must be received with non-zero coefficients, and the undesired subfiles which can not be cancelled by utilizing its cached content, must be zero-forced. Then we can show the existence of such linear combinations simply by observing the fact that the number of variables (coefficients) equals the number of equations (received signal requirements). The details of the proof are omitted here. ■

D. Subpacketization Complexity Analysis

In this section, we provide a comprehensive performance comparison between the proposed hypercube-based based scheme and the NMA scheme.

In the hypercube-based scheme, each file in the library is split into $\left(\frac{N}{M_T}\right)^{t_T}\left(\frac{N}{M_R}\right)^{t_R}$ subfiles while in the NMA scheme each file is split into $\binom{K_T}{t_T}\binom{K_R}{t_R}$ subfiles. In the delivery phase, to implement interference cancellation, each requested subfile is further split into

$$\Delta_{\mathrm{HCB}}(K_T, M_T, K_R, M_R, N) \triangleq \binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)! \quad (19)$$

packets in the proposed hypercube-based scheme and

$$\Delta_{\mathrm{NMA}}(K_T, M_T, K_R, M_R, N) \triangleq \binom{K_R-t_R-1}{t_T-1}(t_T-1)!t_R! \quad (20)$$

packets in the NMA scheme. To measure the subpacketization complexity, we count the total number of packets that a specific file needs to be split into which equals the number of subfiles per file times the number of packets per subfile. When counting the number of subfiles per file, both the pre-stored and requested subfiles by any receiver should be included since the total number of packets per file should reflect the size of the smallest units (i.e., packets) that a file is split into. Therefore, the total number of packets per file required for these two schemes are

$$\begin{aligned} &F_{\mathrm{HCB}}(K_T, M_T, K_R, M_R, N) \\ &\quad = {D_T}^{t_T}{D_R}^{t_R}\Delta_{\mathrm{HCB}}(K_T, M_T, K_R, M_R, N), \quad (21) \\ &\quad F_{\mathrm{NMA}}(K_T, M_T, K_R, M_R, N) \\ &\quad = \binom{K_T}{t_T}\binom{K_R}{t_R}\Delta_{\mathrm{NMA}}(K_T, M_T, K_R, M_R, N). \quad (22) \end{aligned}$$

Since the comparison of subpacketization levels is always done under the same set of system parameters, we ignore the these parameters in the expressions of $\Delta_{\mathrm{HCB}}, \Delta_{\mathrm{NMA}}, F_{\mathrm{HCB}}$ and F_{NMA} for brevity. To compare the subpacketization level between our scheme and the NMA scheme, we define the multiplicative gap of the subpacketization levels between these two schemes as follows.

Definition 3 (Multiplicative Gap of Subpacketization Levels): For the system parameters K_T, M_T, K_R, M_R and N, the *multiplicative gap* G of the subpacketization levels between the hypercube-based scheme and the NMA scheme, is defined as

$$G(K_T, M_T, K_R, M_R, N) \triangleq \frac{F_{\mathrm{HCB}}(K_T, M_T, K_R, M_R, N)}{F_{\mathrm{NMA}}(K_T, M_T, K_R, M_R, N)}. \quad (23)$$

For ease of notation, we ignore the parameters and simply denote $G = F_{\mathrm{HCB}}/F_{\mathrm{NMA}}$. ◊

We next show that for any system parameters, the hypercube scheme has a strictly lower subpacketization level than that of the NMA scheme. Moreover, we show that there is an order gain compared to the NMA scheme when $t \to \infty$ if d and δ are fixed.

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

$$G(d, t, \delta) \leq C_0\left(\frac{C_1}{t}\right)^t \frac{1}{t^{(\delta-1)t-1}}, \quad (24)$$

where the constants (independent of t) are

$$C_0 = \frac{(d-2)!e^6}{(d-\delta-1)!}\left(\sqrt{\frac{d-1}{\delta}}\frac{(d-1)!}{(d-\delta-1)!}\left(\frac{2\pi}{d-1}\right)^{3/2}\right)^{-1}$$

and

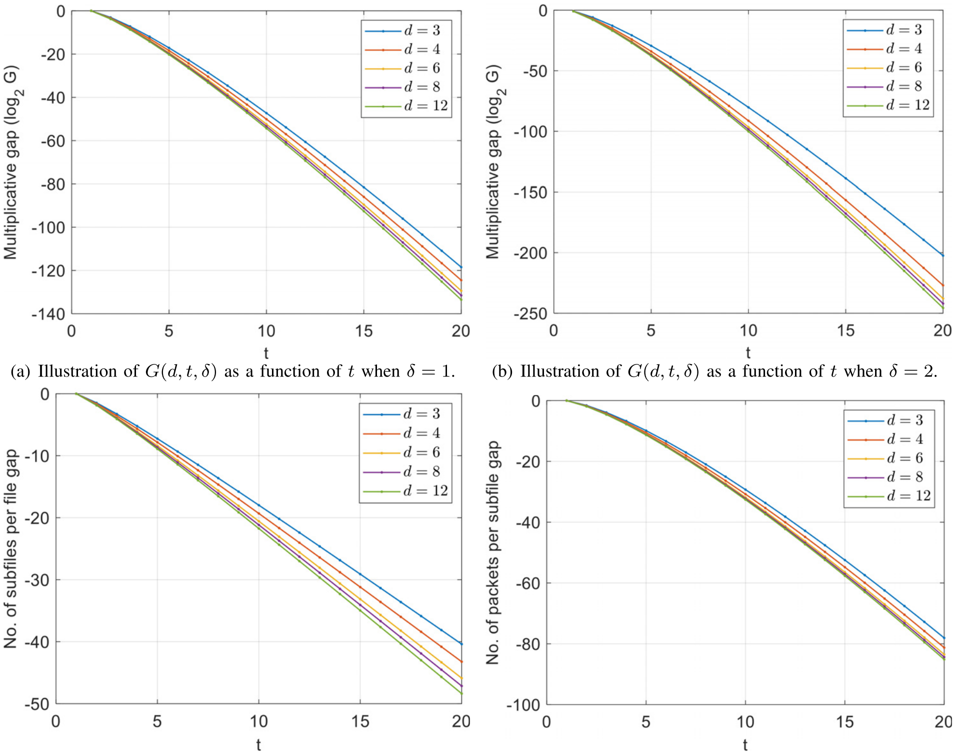
$$C_1 = \frac{(d-1)!}{(d-\delta-1)!}\left(\frac{e}{d-1}\right)^{\delta}\left(\frac{d}{d-1}\right)^{-(\delta+1)(d-1)}\left(\frac{d-1}{d-\delta-1}\right)^{-(d-\delta-1)}.$$

Proof: See Appendix C. ■

Theorem 2 shows that the proposed hypercube-based scheme strictly outperforms the NMA scheme in terms of subpacketizaiton while achieving the same one-shot liner sum-DoF. In fact, the proposed scheme requires not only a smaller number of subfiles per file but also a smaller number of packets per subfile than the NMA scheme, demonstrating the advantage of the hypercube-based design. From (24) we see that if $t \geq C_1$, $G(d, t, \delta) \leq C_0/t^{(\delta-1)t-1}$. Therefore, for fixed d and δ, we have the scaling $G(d, t, \delta) = O(1/t^{(\delta-1)t-1})$ as $t \to \infty$, implying that there is an order gain in subpacketization of the hypercube-based scheme compared to the NMA scheme. Fig. 4 shows the multiplicative gain $G(d, t, \delta)$ under logarithmic scale for the case when $\delta \triangleq t_T/t_R = 1, 2$ under the setting $t_T = \delta t_R = \delta t, D_R = D_R = d$. 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), (b)), which demonstrates an order gain in subpacketization reduction of the proposed scheme compared to the NMA scheme. Moreover, from Fig. 4 (c), (d), it can be seen that the proposed scheme also requires exponentially smaller number of subfiles per file and packets per subfile.

V. Discussion

In this section, we will first provide two possible extensions of the proposed scheme, which are cache-aided



(a) Illustration of $G(d, t, \delta)$ as a function of t when $\delta = 1$.

(b) Illustration of $G(d, t, \delta)$ as a function of t when $\delta = 2$.

(c) No. of subfiles per-file gap as a function of t when $\delta = 1$.

(d) No. of packets per-subfile gap as a function of t when $\delta = 1$.

Fig. 4. The multiplicative gap G between the hypercube scheme and the NMA scheme. The comparison is down under the setting $t_T = \delta t_R = \delta t$, $N/M_T = N/M_R = d$, which implies $K_T = \delta K_R = \delta dt$. It can be seen that: (a) $\delta = 1$. For a fixed d, G decreases (exponentially) quickly as t increases and approaches zero as t goes to infinity, and (b) $\delta = 2$. In this case, the number of transmitters increases and G decreases faster. (c) $\delta = 1$. Comparison of the No. of subfiles per file of the hypercube design to the NMA scheme (logarithmic scale). (d) $\delta = 1$. Comparison of the No. of packets per subfile of the two schemes (logarithmic scale).

Device-to-Device (D2D) interference networks and wireless coded distributed computing networks. Second, we will discuss the connection to and differences from some related existing works.

A. Extension to Cache-Aided D2D Interference Networks and Wireless Distributed Computing Systems

In the setting of a typical cache-aided D2D interference networks, all the nodes (or devices) are expected to have homogeneous cache memory sizes. The proposed hypercube-based scheme can be directly extended to such D2D interference networks to achieve an order-optimal one-shot linear sum-DoF while maintaining the promised subpacketization levels compared to the direct translation of the NMA scheme. There are multiple approaches to apply the hypercube-based approach to cache-aided D2D interference networks. In the following, we will illustrate one example of such applications. We consider a D2D interference network with a library of N files and K nodes, each equipped with a cache memory of size M files. We assume K is even and $t = KM/N \le K/2$. We partition the network into two groups with equal number of devices, i.e., each group has $K/2$ devices. Let $t' = \frac{KM}{2N} \in \mathbb{Z}^+$. In the prefetching phase, in each group, we perform the hypercube cache placement such that the two groups have identical cache placement. The delivery phase has two steps, in the first step, one group of nodes will perform as transmitters and the other group will perform as receivers. Note that since $K_T = K_R = K/2$, the proposed delivery scheme based on the hypercube cache placement can be directly used. The achievable sum-DoF is $t = t_T + t_R = KM/N$. In the first phase, the requests from one group of receivers can be served. In the second step, we exchange the groups of transmitters and receivers such that the other group can be served with the same achievable sum-DoF. Therefore, the total achievable sum-DoF is given by $t = KM/N$.

Moreover, due to the similarity between the cache-aided D2D interference network and Coded Distributed Computing (CDC, [30]), the hypercube cache placement can be directly applied to the wireless CDC interference networks. From the wireless D2D caching network example, it can be seen that the proposed hypercube-based scheme can be applied in a more practical half-duplex transmission settings. For example, the hypercube cache placement scheme can be employed in the file assignment phase in the CDC networks. Then we use the same delivery scheme as in the wireless D2D caching networks to achieve an order optimal communication-computation trade-off.

B. Comparison With Existing Works

In this section we discuss the connection to and the differences from the most related works [13] and [27] and thus highlight the uniqueness of the hypercube-based design.

The work of [13] shows that by adding multiple (L) transmit antennas the supacketization level of coded caching can be reduced approximately to its L-th root compared to the shared-link coded caching scheme. It turns out that this scheme can be extended to the cache-aided $K_T \times K_R$ interference networks to achieve the same sum-DoF as the hypercube-based scheme proposed. However, due to the use of user/receiver grouping, the scheme of [13] suffers sum-DoF loss (i.e., can not achieve $t_T + t_R$) when either K_R/t_T or t_R/t_T is not an integer, which means that it can not achieve sum-DoF $t_T + t_R$ when $\delta = t_T/t_R > 1$, putting a major limitation to its applicability. Moreover, [27] considered a similar setting as [13] but a totally different cyclic cache placement based on PDA was proposed to achieve the sum-DoF $K\gamma + L$ with a quadratic (w.r.t. K) subpacketization. However, the proposed scheme works only when $K\gamma \leq L$. The differences of our work from these two works are summarized as follows.

1) Different design methodologies and parameter regimes. The hypercube-based scheme applies the hypercube cache placement with a nice geometric interpretation and does not rely on receiver grouping which requires that $t_R \geq t_T$ and puts a strong limitation in applying to interference networks. In contrast, our work primarily focused on the case $\delta \geq 1$, although by using a memory sharing alike method, the hypercube-based design can be extended to the cases when δ is not an integer or $\delta < 1$ without sum-DoF loss (Note that the scheme of [27] does not work when $\delta < 1$). At the point $\delta = 1$, the scheme of [13] achieves a lower subpacketization level than the hypercube-based scheme which can be shown as follows. Assume $M_T \in \mathbb{Z}^+$ (otherwise the transmitter side cache placement in [13] does not work), then [13] requires subpacketization $\binom{K_T/t_T}{t_R/t_T} = K_T/t_T = D_T$ while the hypercube-based scheme has $F_{\text{HCB}} = D_T^{t_T} D_R^{t_R} \Delta_{\text{HCB}}$ which is larger than D_T.

2) Symmetry in Cache Placement. Different from the hypercube-based design, [13] employs asymmetric cache placement methods at the transmitter and receiver sides. One potential drawback is that the scheme of [13] can not be directly applied to cache-aided Device-to-Device (D2D) interference networks and the wireless CDC systems where each user needs to be both transmitter and receivers in order to fulfill the file requests of all users. However, due to the symmetric cache placements at both the transmitter and receiver sides, the hypercube-based scheme extends naturally to such networks and incurs no extra cost when the users switch their roles from transmitters to receivers or vice versa.

VI. Conclusion

In this paper, we considered the cache-aided interference management problem where the transmitters and receivers are equipped with cache memories of certain sizes to pre-store parts of the contents. We adopt a new cache placement method called hypercube at both the transmitters' and receivers' sides. Based on the hypercube cache placement, we proposed a corresponding delivery scheme where the one-shot linear DoF of $\min\left\{\frac{M_T K_T + M_R K_R}{N}, K_R\right\}$ is achievable with exponentially lower subpacketization compared to the well-known NMA scheme. More specifically, via the design of the cache placement and the communication scheme, a set of $\frac{M_T K_T + M_R K_R}{N}$ packets can be delivered to the receivers simultaneously and interference-free, which is a joint effect of the zero-forcing (collaboration of transmitters via cache placement design at the transmitters' side) and cache cancellation (neutralization of known interference via the cache placement design at the receivers' side). The result shows that our proposed scheme can achieve exactly the same sum-DoF performance as the NMA scheme while requiring significantly lower supacketization levels.

Appendix A
Proof of Lemma 1

First, we show that given a set of $|\mathcal{Q}| = Dt$ points (users) with t dimensions and D points in each dimension, the number of different hypercube permutations is equal to $\left|\Pi_{\mathcal{Q}}^{\text{HCB}}\right| = (D!)^t t!$. According to Definition 1, for a hypercube permutation π^{HCB}, the users belonging to the same dimension $\mathcal{U}_i$ can only appear in positions $p_{i,1}, p_{i,2}, \cdots, p_{i,D-1}$ such that $p_{i,j} \mod t = C_i, \forall j \in [D-1]$, where C_i is a constant irrespective of j and $C_i \in [t-1]$. For two different dimensions $\mathcal{U}_{i_1}$ and $\mathcal{U}_{i_2}$, the corresponding modulo residues $C_{i_1} \neq C_{i_2}$ if $i_1 \neq i_2$. As a result, $\{C_0, C_1, \cdots, C_{t-1}\} = \{0, 1, \cdots, t-1\}$. Thus, given a group of users $\mathcal{U}_i$ and a prescribed modulo residue C_i, there are $D!$ ways to arrange these users to the corresponding set of positions $\{p_{i,j} : p_{i,j} \mod t = C_i, j \in [D-1]\}$. Since we have t such user groups (dimensions), according to the multiplication principle, there are $(D!)^t$ ways to arrange all the users $\mathcal{Q}$ to the positions $\{p_{i,j} : p_{i,j} \mod t = C_i, j \in [D-1], i = 0, 1, \cdots, t-1\}$ under a prescribed modulo residue assignment. Since there are $t!$ different ways to assign the modulo residues $C_0, C_1, \cdots, C_{t-1}$ to the t user groups, we conclude that $\left|\Pi_{\mathcal{Q}}^{\text{HCB}}\right| = (D!)^t t!$.

Now, for any $\pi \in \Pi_{\mathcal{Q}}^{\text{HCB}}$, it is easy to see that there are $Dt - 1$ other permutations in $\Pi_{\mathcal{Q}}^{\text{HCB}}$ which are resulted from circularly shifting the elements of π. Since circular shifting is not allowed in the circular permutation, we have

$$\left|\Pi_{\mathcal{Q}}^{\text{HCB,circ}}\right| = \frac{\left|\Pi_{\mathcal{Q}}^{\text{HCB}}\right|}{Dt} = \frac{(D!)^t (t-1)!}{D}, \tag{25}$$

which completes the proof of Lemma 1.

Appendix B
Proof of Lemma 2

The proof of Lemma 2 can be completed by verifying the following two conditions: 1) For a specific receiver Rx_j, the number of packets it receives in the delivery phase equals the number of packets which are desired but have not been cached by Rx_j; 2) The number of packets received by all K_R receivers equals the number of packets desired by them.

Each set in the union of (16) is composed of $t_T + t_R$ packets. The number of such sets is equal to

$$D_T^{t_T} \binom{D_R}{\delta+1}^{t_R} \frac{((\delta+1)!))^{t_R} (t_R - 1)!}{\delta + 1}. \tag{26}$$

Therefore, the total number of packets in (16) is equal to

$$\begin{aligned} &D_T^{t_T} \binom{D_R}{\delta+1}^{t_R} \frac{((\delta+1)!)^{t_R} (t_R - 1)!}{\delta + 1} (t_T + t_R) \\ &\quad = \frac{D_T^{t_T} K_R (D_R - 1)! (D_R!)^{t_R - 1} (t_R - 1)!}{((D_R - \delta - 1)!)^{t_R}}, \end{aligned} \tag{27}$$

where we used the fact that $\delta = t_T/t_R$ and $t_R = K_R/D_R$.

On the other hand, Rx$_j$, $\forall j \in [K_R-1]$ has cached $D_T^{t_T} D_R^{t_R-1}$ subfiles in the prefetching phase, so the number of subfiles Rx$_j$ needs is equal to $D_T^{t_T} D_R^{t_R-1}(D_R-1)$. Since in the delivery phase, each desired subfile is further split into $\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta}(t_R-1)!$ packets, the total number of packets needed by Rx$_j$ is equal to

$$D_T^{t_T} D_R^{t_R-1}(D_R-1)\binom{D_R-2}{\delta-1}\binom{D_R-1}{\delta}^{t_R-1}\frac{(\delta!)^{t_R}}{\delta} \cdot (t_R-1)! = \frac{D_T^{t_T}(D_R-1)!(D_R!)^{t_R-1}(t_R-1)!}{((D_R-\delta-1)!)^{t_R}}. \quad (28)$$

Therefore, the total number of packets needed by all K_R receivers is equal to

$$K_R D_T^{t_T}\frac{(D_R-1)!(D_R!)^{t_R-1}(t_R-1)!}{((D_R-\delta-1)!)^{t_R}}, \quad (29)$$

which equals the total number of packets in (27), implying that the set of packets needed by the receivers can be grouped into subsets of size t_T+t_R, verifying the second condition. Moreover, the number of packets received by Rx$_j$ in the delivery phase is equal to

$$D_T^{t_T}\binom{D_R-1}{\delta}\binom{D_R}{\delta+1}^{t_R-1}\frac{((\delta+1)!)^{t_R}(t_R-1)!}{\delta+1} = \frac{D_T^{t_T}(D_R-1)!(D_R!)^{t_R-1}(t_R-1)!}{((D_R-\delta-1)!)^{t_R}}, \quad (30)$$

which equals the number of packets calculated in (28), verifying the first condition. As a result, the proof of Lemma 2 is complete.

Appendix C
Proof of Theorem 2

We will first show that for any system parameters K_T, K_R, M_T, M_R and N, which satisfy $K_T = D_T t_T, K_R = D_R t_R$ and $\delta = t_T/t_R \in \mathbb{Z}^+$, we have 1) $D_T^{t_T} < \binom{K_T}{t_T}$, 2) $D_R^{t_R} < \binom{K_R}{t_R}$, and 3) $\Delta_{\mathrm{HCB}} < \Delta_{\mathrm{NMA}}$. As a result, we obtain $G < 1$.

We first prove that $D_T^{t_T} < \binom{K_T}{t_T}$. For ease of notation, we denote D_T as d and t_T as t for the time being. We have

$$\frac{D_T^{t_T}}{\binom{K_T}{t_T}} = \frac{d^t}{\binom{dt}{t}} = \frac{d^t t!}{dt(dt-1)(dt-2)\cdots(dt-(t-1))} = \left(\frac{t}{t}\right)\left(\frac{t-1}{t-\frac{1}{d}}\right)\cdots\left(\frac{t-(t-1)}{t-\frac{t-1}{d}}\right). \quad (31)$$

Since we have assumed that $d \geq \delta+1 \geq 2$ where $\delta \geq 1$, it can be seen that $t-i \leq t-i/d, \forall i \in [t-1]$, implying that each individual term on the RHS of (31) is less than 1. As a result, the product is less than 1, implying $D_T^{t_T} < \binom{K_T}{t_T}$. Similarly, we can prove $D_R^{t_R} < \binom{K_R}{t_R}$.

Next we prove $\Delta_{\mathrm{HCB}} < \Delta_{\mathrm{NMA}}$. Denote t_R as t and D_R as d, we have $t_T = \delta t_R = \delta t$. Thus, Δ_{HCB} and Δ_{NMA} can be written as

$$\Delta_{\mathrm{HCB}} = \binom{d-2}{\delta-1}\binom{d-1}{\delta}^{t-1}\frac{(\delta!)^t}{\delta}(t-1)! = \frac{((d-1)!)^t(t-1)!}{((d-\delta-1)!)^t(d-1)}, \quad (32)$$

$$\Delta_{\mathrm{NMA}} = \binom{dt-t-1}{\delta t-1}(\delta t-1)!t! = \frac{(dt-t-1)!t!}{((d-\delta-1)t)!}. \quad (33)$$

Therefore,

$$\begin{aligned}\frac{\Delta_{\mathrm{NMA}}}{\Delta_{\mathrm{HCB}}} &= \frac{((d-\delta-1)!)^t((d-1)t)!}{((d-\delta-1)t)!((d-1)!)^t} \\ &= \frac{\prod_{i=0}^{\delta t-1}((d-1)t-i)}{\left(\prod_{i=0}^{\delta-1}(d-1-i)\right)^t} \\ &= \lambda_0\lambda_1\cdots\lambda_{t-1}, \end{aligned} \quad (34)$$

in which the parameter λ_k is defined as

$$\lambda_k \triangleq \frac{\prod_{i=k\delta}^{(k+1)\delta-1}((d-1)t-i)}{\prod_{i=0}^{\delta-1}(d-1-i)}, \quad \forall k \in [t-1]. \quad (35)$$

Note that $\lambda_0 > \lambda_1 > \cdots > \lambda_{t-1}$. Next we show that $\lambda_{t-1} \geq 1$. From (35), we have

$$\begin{aligned}\lambda_{t-1} &= \frac{\prod_{i=(t-1)\delta}^{\delta t-1}((d-1)t-i)}{\prod_{i=0}^{\delta-1}(d-1-i)} \\ &= \prod_{i=0}^{\delta-1}\left(t-\frac{(\delta-i)(t-1)}{d-1-i}\right) \\ &\overset{(a)}{\geq} \prod_{i=0}^{\delta-1}\left(t-\frac{(\delta-i)(t-1)}{\delta+1-1-i}\right) \\ &= \prod_{i=0}^{\delta-1}(t-(t-1)) = 1, \end{aligned} \quad (36)$$

where in (a) we used the assumption that $d \geq \delta+1$. Hence, we obtain that $\lambda_{t-1} \geq 1$. Since $\lambda_0 > \lambda_1 > \cdots > \lambda_{t-1} \geq 1$, we have $\frac{\Delta_{\mathrm{NMA}}}{\Delta_{\mathrm{HCB}}} = \lambda_0\lambda_1\cdots\lambda_{t-1} > 1$, implying $\Delta_{\mathrm{HCB}} < \Delta_{\mathrm{NMA}}$. Combining the above results, we conclude that the multiplicative gap G is strictly less than 1 for any system parameters, i.e., $G = \frac{{D_T}^{t_T}{D_R}^{t_R}}{\binom{K_T}{t_T}\binom{K_R}{t_R}} \cdot \frac{\Delta_{\mathrm{HCB}}}{\Delta_{\mathrm{NMA}}} < 1$. This proof also shows that the hypercube based scheme requires less number of subfiles per file in the prefetching phase and and less number of packets per subfile in the delivery phase than the NMA scheme.

Due to space limit, we refer the reader to Appendix C of [31] for the proof of the the upper bound on $G(d,t,\delta) \leq C_0\left(\frac{C_1}{t}\right)^t\frac{1}{t^{(\delta-1)t-1}}$ in Theorem 2. As a result, the proof of Theorem 2 is complete.

References

[1] X. Zhang, N. Woolsey, and M. Ji, "Cache-aided interference management using hypercube combinatorial cache designs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[2] G. M. D. T. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, 2019.

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[4] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 135–139.

[5] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[6] M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 809–813.

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

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

[9] S. S. Bidokhti, M. Wigger, and A. Yener, "Gaussian broadcast channels with receiver cache assignment," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

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

[11] S. P. Shariatpanahi, G. Caire, and B. H. Khalaj, "Physical-layer schemes for wireless coded caching," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2792–2807, May 2019.

[12] F. Xu, M. Tao, and K. Liu, "Fundamental tradeoff between storage and latency in cache-aided wireless interference networks," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7464–7491, Nov. 2017.

[13] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1176–1188, Jun. 2018.

[14] F. Xu and M. Tao, "Fundamental limits of decentralized caching in fog-RANs with wireless fronthaul," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1430–1434.

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

[16] K. Wan, D. Tuninetti, and P. Piantanida, "An index coding approach to caching with uncoded cache placement," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1318–1332, Mar. 2020.

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

[18] M. Ji *et al.*, "On the fundamental limits of caching in combination networks," in *Proc. IEEE 16th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2015, pp. 695–699.

[19] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Caching in combination networks," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 1269–1273.

[20] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "State-of-the-art in cache-aided combination networks," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, Oct. 2017, pp. 641–645.

[21] A. A. Zewail and A. Yener, "Combination networks with or without secrecy constraints: The impact of caching relays," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1140–1152, Jun. 2018.

[22] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "Combination networks with end-user-caches: Novel achievable and converse bounds under uncoded cache placement," 2017, *arXiv:1701.06884*. [Online]. Available: http://arxiv.org/abs/1701.06884

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

[24] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5821–5833, Sep. 2017.

[25] L. Tang and A. Ramamoorthy, "Coded caching schemes with reduced subpacketization from linear block codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3099–3120, Apr. 2018.

[26] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Trans. Inf. Theory*, vol. 64, no. 8, pp. 5755–5766, Aug. 2018.

[27] M. Salehi, E. Parrinello, S. P. Shariatpanahi, P. Elia, and A. Tölli, "Low-complexity high-performance cyclic caching for large MISO systems," 2020, *arXiv:2009.12231*. [Online]. Available: http://arxiv.org/abs/2009.12231

[28] N. Woolsey, R.-R. Chen, and M. Ji, "Towards finite file packetizations in wireless device-to-device caching networks," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5283–5298, Sep. 2020.

[29] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial coded design for heterogeneous distributed computing," 2020, *arXiv:2007.11116*. [Online]. Available: http://arxiv.org/abs/2007.11116

[30] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.

[31] X. Zhang, N. Woolsey, and M. Ji, "Cache-aided interference management using hypercube combinatorial cache design with reduced subpacketizations and order optimal sum-degrees of freedom," 2020, *arXiv:2008.08978*. [Online]. Available: http://arxiv.org/abs/2008.08978

Xiang Zhang (Student Member, IEEE) received the bachelor's degree in electronic and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, The University of Utah, Salt Lake City, UT, USA. His research interests include information theory, coding theory, wireless communication, information retrieval in caching systems, and distributed computing. He was a recipient of the National Endeavor Scholarship of China in 2015.

Nicholas Woolsey (Student Member, IEEE) received the B.S. degree in biomedical engineering from the University of Connecticut in 2012 and the M.Eng. degree in bioengineering from the University of Maryland, College Park, MD, USA, in 2015, with a focus on signal processing, imaging and optics. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, The University of Utah. He is also with Trabus Technologies, San Diego, CA, USA. His research interests include combinatorial designs and algorithms for resource allocation, coding and efficient communications in distributed computing, and private and caching networks. From 2014 to 2017, he was an Electrical Engineer with Northrop Grumman Corporation (NGC), Ogden, UT, USA, developing test and evaluation methods, modernization solutions and signal processing algorithms for the sustainment of aging aircraft and ground communication systems. While at NGC, he received the "Outside the Box" Grant to investigate the design of a modern receiver that interfaces aging technology and the 2016 Brent Scowcroft Team Award for performing exceptional systems engineering work.

Mingyue Ji (Member, IEEE) received the B.E. degree in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2006, the M.Sc. degree in electrical engineering from the Royal Institute of Technology, Sweden, in 2008, the M.Sc. degree in electrical engineering from the University of California, Santa Cruz, in 2010, and the Ph.D. degree from the Ming Hsieh Department of Electrical Engineering, University of Southern California, in 2015. He subsequently was a Staff II System Design Scientist with Broadcom Corporation (Broadcom Limited) from 2015 to 2016. He is currently an Assistant Professor with the Electrical and Computer Engineering Department and an Adjunct Assistant Professor with the School of Computing, The University of Utah. His research interests include the broad area of information theory, coding theory, concentration of measure and statistics with the applications of caching networks, wireless communications, distributed storage and computing systems, federated learning, and (statistical) signal processing. He received the IEEE Communications Society Leonard G. Abraham Prize for the Best IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC) Paper in 2019, the Best Paper Award in IEEE ICC 2015 Conference, the Best Student Paper Award in IEEE European Wireless 2010 Conference, and USC Annenberg Fellowship from 2010 to 2014. He is also an Associate Editor of IEEE TRANSACTIONS ON COMMUNICATIONS.