Towards Finite File Packetizations in Wireless Device-to-Device Caching Networks

Nicholas Woolsey, *Student Member, IEEE*, Rong-Rong Chen, *Member, IEEE*, and Mingyue Ji, *Member, IEEE*

***Abstract*— We consider wireless device-to-device (D2D) caching networks with single-hop transmissions. Previous work has demonstrated that caching and coded multicasting can significantly increase per user throughput. However, the state-of-the-art coded caching schemes for D2D networks are generally impractical because content files are partitioned into an exponential number of packets with respect to the number of users if both library and memory sizes are fixed. In this paper, we present two combinatorial approaches of D2D coded caching network design with reduced packetizations and desired throughput gain compared to the conventional uncoded unicasting. The first approach uses a "hypercube" design, where each user caches a "hyperplane" in this hypercube and the intersections of "hyperplanes" represent coded multicasting codewords. In addition, we extend the hypercube approach to a decentralized design. The second approach uses the Ruzsa-Szeméredi graph to define the cache placement. Disjoint matchings on this graph represent coded multicasting codewords. Both approaches yield an exponential reduction of packetizations while providing a per-user throughput that is comparable to the state-of-the-art designs in the literature. Furthermore, we apply spatial reuse to the new D2D network designs to further reduce the required packetizations and significantly improve per user throughput for some parameter regimes.**

***Index Terms*— Coded caching, device-to-device communications, packetizations, spatial reuse.**

I. Introduction

WIRELESS caching is a promising approach to significantly improve the user throughput and simultaneously accommodate a large number of user demands in future generations of wireless networks [1]–[13]. In this paper, we investigate achievable *coded caching* schemes in device-to-device (D2D) caching networks, where users strategically cache packets of content files to enable coded multicasting which serves distinct content to multiple users with one channel use. Different from the seminal *shared-link* caching networks [1], where one source node (base station) with access to the entire library serves all the users over a multicast channel, in D2D networks, users receive requested packets from other users. For such a network consisting of n users, each caching an equivalent M files out of a library of m files, previous work demonstrates that if $Mn \geq m$ and $m \geq n$ and when spatial reuse is not allowed, meaning that any transmission can be successfully received by any users in the network, the transmission rate (i.e., normalized traffic load) is $\Theta\left(\frac{m}{M}\right)$, which is not a function of n. Hence, the aggregate throughput of the network is scalable [14].[1] This surprising result shows that the transmission rate of the shared-link caching scheme in [1] and D2D caching scheme in [14] are identical for a large number of users.[2]

D2D caching networks have the potential to provide some unique advantages. For example, D2D caching networks have a greater flexibility to implement spatial reuse in comparison to shared-link caching networks. The authors in [14] demonstrate that users in a D2D network can be grouped into clusters based on proximity. These clusters can perform the coded multicasting delivery simultaneously and surprisingly, the order-optimal traffic load in each cluster is identical to the traffic load when no clustering is enabled (e.g., consider the entire network as a single cluster). Nevertheless, clustering may improve per user throughput since the link rate (bits/second/Hz) in each cluster may increase as the size of each cluster decreases. Due to their unique characteristics, the study of the fundamental limits of D2D caching networks has become a popular topic in the past few years [10], [14]–[23].

The promised gain in per user throughput of the state-of-the-art coded D2D caching schemes relies on a large amount of file packetization which makes the networks impractical to implement. Files need to be split into a very large number of packets and therefore the files will be unrealistically large for many caching network implementations. In this paper, we study and propose new achievable coded caching schemes in D2D networks such that the packetization of each file is significantly reduced without sacrificing much throughput of the currently proposed D2D caching schemes.

Manuscript received October 16, 2019; revised March 17, 2020, May 27, 2020, and June 23, 2020; accepted June 23, 2020. Date of publication July 3, 2020; date of current version September 16, 2020. Work supported through the National Science Foundation grants CCF-1817154 and SpecEES-1824558 and the Idaho National Laboratory (INL) Laboratory Directed Research and Development (LDRD) Program under DOE Idaho Operations Office Contract DE-AC07-05ID14517. The associate editor coordinating the review of this article and approving it for publication was L. Ong. *(Corresponding authors: Nicholas Woolsey; Mingyue Ji.)*

The authors are with the Department of Electrical and Computer Engineering, The University of Utah, Salt Lake City, UT 84112 USA (e-mail: nicholas.woolsey@utah.edu; rchen@ece.utah.edu; mingyue.ji@utah.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2020.3006897

[1]Note that when no spatial reuse is allowed, the per user throughput is inversely proportional to the traffic load in the network.

[2]We will use the following standard "order" notation: given two functions f and g, we say that: 1) $f(n) = O(g(n))$ if there exists a constant c and integer N such that $f(n) \leq cg(n)$ for $n > N$. 2) $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. 3) $f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$. 4) $f(n) = \omega(g(n))$ if $g(n) = o(f(n))$. 5) $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

There have been many results studying the large packetization issue in shared-link caching networks [24]–[26] and we discuss some of them here. The work of [24] used a Placement Delivery Array (PDA) to investigate new placement and delivery schemes. One of the scheme in [24] reduces the number of users served in each coded multicast transmission by 1 while significantly reducing the packetization compared to the seminal work of [1]. In this way, the rate only increases slightly while greatly increasing the number of practical parameters regimes. Furthermore, the authors of [25] demonstrated the connection between (m, k) linear block codes over $\mathrm{GF}(q)$ with coded caching network design. The generated codeword matrix defines the cache of mk users and at most $(k+1)q^k$ file packetizations are necessary. While the scheme only works for linear block codes with the $(k, k+1)$-consecutive column property (see [25]), the authors demonstrated the flexibility of this approach and in some cases it can be used to design a caching network given n, m and M while meeting a specific packetization requirement. While the schemes of [24] and [25] significantly reduce the file packetization compared to [1], all of these schemes require an exponential number of packets per file compared to the number of users. A recent result has demonstrated that caching schemes where a linear number of packets per file are necessary by using a Ruzsa-Szeméredi graph [27], [28] to design coded caching scheme in order to have the global caching gain [1], [29]. While this approach requires a large number of users, it has proven the existence of sub-exponential schemes which inspires the search for practical caching schemes with reduced packetizations.

There has been limited work studying file packetization in D2D caching networks. In [14], the authors demonstrate that if no spatial reuse is allowed, let $t = \frac{nM}{m} \in \mathbb{Z}^+$, the required number of packets per file is $K = t\binom{n}{t}$ which grows exponentially as the number of users, n, increases. The authors of [14] also explored the concept of user clustering in order to exploit spatial reuse to increase per user throughput. Moreover, it was found that clustering also has the potential to reduce packetizations. Another approach to study D2D coded caching networks is the use of a D2D Placement Delivery Array (DPDA) [30]. By using the DPDA, the authors first derived a lower bound for the rate of a coded caching network as $R \geq \frac{m}{M} - 1$ and also a lower bound for packetizations when the rate lower bound is met and $t \in \{1, 2, n-2, n-1\}$. The work of [30] also demonstrated that the scheme of [14] meets the lower bound on rate always, meets the lower bound on packetization for $t \in \{1, n-1\}$ but does not meet the lower bound on packetization when $t \in \{2, n-2\}$. The authors of [30] developed a specific scheme for $t = 2$ and $t = n-2$ which meets the lower bound on rate and packetization. An extension of this work can be found in [31], where the authors adopted the approach introduced in [14] to any PDA based designs to yield a D2D coded caching design. However, this approach requires extra packetization for placement. An open question remains as to the existence of D2D coded caching networks which work for a large range of t and are designed specifically for D2D and not simply adapted from a shared-link scheme. Furthermore, only the scheme of [14] has considered spatial reuse which is a potential advantage of D2D networks to further reduce packetizations without reducing per user throughput.

In this paper, we study several approaches to design coded caching networks with reduced packetizations. We propose two combinatorial designs for centralized D2D caching networks which have reduced packetization compared to [14]. The first approach uses a hypercube to define the cache placement and we demonstrate how the geometry of this hypercube relates to coded multicasting opportunities for delivery. The hypercube approach is optimized specifically for D2D caching networks such that, for some cases, it does not require additional packetization beyond the placement phase as seen with previous D2D coded caching designs. In addition, by adopting the idea recently proposed in [32], we extend this approach to a decentralized coded D2D caching scheme, which allows a much more flexible design for given network parameters. Meanwhile, the advantage of the reduced packetization of the hypercube approach still remains in the decentralized D2D caching networks. The second approach is based on an application of the Ruzsa-Szeméredi graph [27], [28], which is first used for shared-link caching in [29]. We extend the use of Ruzsa-Szeméredi graph to D2D caching networks. Both D2D combinatorial designs, sustain the significant throughput gain compared to conventional uncoded unicast [16] and the required packetizations are reduced exponentially compared to [14] with respect to the number of users n while keeping the library size m and memory size M fixed. Finally, we study the impact of enabling spatial reuse in these caching network designs and show this can further reduce the required packetizations, while also improving the per user throughput significantly for some parameter regimes. The work we present here includes and expands on our previous conference submissions [33], [34].

This work makes two key contributions. First, while it is commonly believed that a D2D caching scheme should be converted from a shared-link caching scheme (as shown in [14]) instead of being designed from scratch, this work establishes the existence of the first expandable D2D schemes based on hypercube designs that are not derived from shared-link schemes. The proposed hypercube design can achieve a load-memory-subpacketization tradeoff that has not been achieved in prior works. It has an appealing geometric interpretation that is essential for a better understanding of the differences between shared-link and D2D caching schemes. Furthermore, the hypercube design allows possible extensions to the design of heterogeneous coded distributed computing schemes [35], [36], which are equivalent to heterogeneous D2D coded caching schemes. Second, unlike shared-link D2D caching networks, spatial reuse is an important feature for D2D networks. However, the impact of spatial reuse has largely been ignored for coded D2D caching schemes. In this work, we adopt the Ruzsa-Szeméredi graph approach from shared-link caching networks [29] to D2D networks and show for the first time that spatial reuse can significantly increase the per-user throughput while reducing subpacketization dramatically (about quadratic in terms of the number of users per cluster) of D2D caching networks.

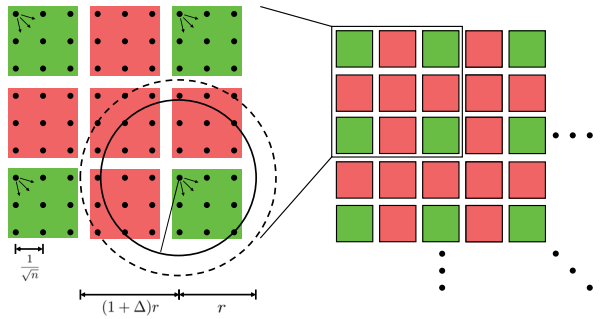


Fig. 1. a) Representation of 81 users on a square grid. Users are divided into 9 equally sized clusters of 9 users based on user proximity. The clusters highlighted in green represent clusters which can be simultaneously active assuming that cluster highlighted in red are not active. b) A larger network which includes the 81 users of Fig. 1(a) where the clusters highlighted in green can be simultaneously active. The reuse factor is $\mathcal{K} = 4$.

The outline of this paper is as follows. In section II, we introduce the D2D network model and problem formulation. In Section III, we describe the proposed centralized hypercube based coded caching approach and analyze its performance. In Section IV, we extend the hypercube design to a decentralized D2D caching network. Section V introduces the Ruzsa-Szeméredi graph based coded caching approach and analyzes its performance. In Section VI, we show how the proposed schemes can take advantage of spatial reuse. Finally, we conclude the paper in Section VII.

Notation Convention: We use $|\cdot|$ to represent the cardinality of a set or the length of a vector. For $n, m \in \mathbb{Z}^+$, let $[n] := \{1, 2, \ldots, n\}$ and define $[n]^m$ as the set of all m-length vectors whose elements are in $[n]$. A bold symbol such as $\boldsymbol{a}$ indicates a vector and a_i denotes the i-th element of $\boldsymbol{a}$.

II. Network Model and Problem Formulation

We consider a wireless D2D network with single-hop transmissions formed by the set of users $\mathcal{U} = \{1, \ldots, n\}$. The users are uniformly distributed on a unit grid with a minimum distance of $1/\sqrt{n}$ as shown in Fig. 1(a). Each user $u \in \mathcal{U}$ makes a request $f_u \in \mathcal{F}$, where $\mathcal{F} = \{1, \cdots, m\}$ is a file library of m independently generated messages $\{W_1, \ldots, W_m\}$ with entropy F bits each. We denote the demand vector as $\boldsymbol{f} = (f_1, \cdots, f_n)$. The file library, $\mathcal{F}$, is generated once and kept unchanged during subsequent network operations. In addition, we assume $m \geq n$ and users request distinct files. Each user locally caches the equivalent of M files, or MF bits. Furthermore, define $t \triangleq nM/m \geq 1$, as the number of times the library is cached collectively among the users.

Users have active links between one another based on the *protocol model* [37] described as follows. A communication link, consisting of user u transmitting to user v, will be successful if and only if the distance between user u and v is less than or equal to r and user v is at least a distance of $(1 + \Delta)r$ from all transmitting users other than user u. The parameters r, $\Delta > 0$ are given by the protocol model. We assume that any $r > 0$ is possible and r dictates a constant data rate, C_r, in the unit of bits/s/Hz.[3]

[3]In practice, C_r can be a function of the transmission range r. However, the protocol model does not capture this relationship.

The protocol model allows for spatial reuse as shown in Fig. 1(a). 4 users, one in each green cluster, are transmitting to 8 local users who are at most a distance of r away. The users receiving the transmission are at least a distance of $(1 + \Delta)r$ from the other three transmitting users. The users of Fig. 1(a) may be part of a larger network of users as shown in Fig. 1(b) which depicts active clusters (involved in a successful communication link) in green and non-active clusters (neither receiving or transmitting) in red. The set of active clusters highlighted in green is one of $\mathcal{K}$ sets. $\mathcal{K}$ is defined as the reuse factor or the number of cluster sets such that, for any given set, each cluster of that set can be active without interference and the $\mathcal{K}$ cluster sets collectively include all clusters.

A D2D caching scheme consists of three phases: the cache placement phase, the coded delivery phase, and the transmission phase. These are defined as follows.

Definition 1: **(Cache Placement Phase)** The cache placement phase maps the file library $\mathcal{F}$ onto the cache of each user. For $u \in \mathcal{U}$, the function $\phi_u : \mathbb{F}_2^{mF} \to \mathbb{F}_2^{MF}$ generates the cache content $Z_u \triangleq \phi_u(W_f : f \in \mathcal{F})$ stored in the cache of user u and kept fixed throughout subsequent operations. $\Diamond$

The cache functions $\{\phi_u : u \in \mathcal{U}\}$ can be centralized or decentralized. In this work, the centralized setting means that the cache of each user is dependent of the cache of the other users; while in the decentralized setting, each user's cache is independent of any other user's cache and independent of the total number of users n.

Definition 2: **(Coded Delivery Phase)** The coded delivery phase is defined by two sets of functions: the node encoding functions, denoted by $\{\psi_u : u \in \mathcal{U}\}$, and the node decoding functions, denoted by $\{\lambda_u : u \in \mathcal{U}\}$. Let R_u^{T} denote the number of *coded bits* transmitted by node u to satisfy the demand vector $\boldsymbol{f}$. The transmission rate of node u is defined by $R_u = \frac{R_u^{\mathrm{T}}}{F}$. The function $\psi_u : \mathbb{F}_2^{MF} \times \mathcal{F}^n \to \mathbb{F}_2^{FR_u}$ generates the transmitted message $X_{u,\boldsymbol{f}} \triangleq \psi_u(Z_u, \boldsymbol{f})$ of node u as a function of its cache content Z_u and the demand vector $\boldsymbol{f}$.[4] Let $\mathcal{D}_u$ denote the set of users whose transmit messages are received by user u (according to some transmission policy in Definition 3). The function $\lambda_u : \mathbb{F}_2^{F\sum_{v\in\mathcal{D}_u} R_v} \times \mathbb{F}_2^{MF} \times \mathcal{F}^n \to \mathbb{F}_2^F$ decodes the request of user u from the received messages and its own cache, i.e., we have

$$\hat{W}_{u,\mathsf{f}} \triangleq \lambda_u(\{X_{v,\mathsf{f}} : v \in \mathcal{D}_u\}, Z_u, \boldsymbol{f}). \qquad \Diamond$$

Definition 3: **(Transmission Phase)** The transmission policy Π is a rule to activate D2D links in the network. Let $\mathcal{L}$ denote the set of all directed links. Let $\mathcal{A} \subseteq 2^{\mathcal{L}}$ denote the set of all possible feasible subsets of links (this is a subset of the power set of $\mathcal{L}$, formed by all sets of links forming independent sets in the network interference graph induced by the protocol model). Let $\mathsf{A}_\mathsf{t} \subset \mathcal{A}$ denote a feasible set of simultaneously active links at time t. A feasible transmission policy Π consists of a sequence of activation sets, i.e., sets of active transmission links, $\{\mathsf{A}_\mathsf{t} : \mathsf{t} = 1, 2, 3, \ldots\}$, such that at each time t the active links in A_t do not violate the protocol model. $\Diamond$

[4]We also refer the transmission rate to traffic load in this paper.

Since users make arbitrary requests, similar to [1], [14], we focus on the worst-case error probability defined as

$$P_e = \max_{\boldsymbol{f}\in\mathcal{F}^n} \max_{u\in\mathcal{U}} \mathbb{P}\left(\hat{W}_{u,\boldsymbol{f}} \neq W_{f_u}\right).$$

For a given number of users n and library size m, letting the transmission rate $R = \sum_{u\in\mathcal{U}} R_u$, we say that the cache-rate pair (M, R) is achievable if $\forall\ \varepsilon > 0$ there exists a sequence indexed by the file size $F \to \infty$ of cache encoding functions $\{\phi_u\}$, delivery functions $\{\psi_u\}$ and decoding functions $\{\lambda_u\}$, with rate $R^{(F)}$ and probability of error $P_e^{(F)}$ such that $\limsup_{F\to\infty} R^{(F)} \leq R$ and $\limsup_{F\to\infty} P_e^{(F)} \leq \varepsilon$. Note that RF gives the achievable total traffic load transmitted in the whole network.

Different from the shared-link network model [1], the performance of D2D caching networks cannot be completely characterized by the transmission rate, R, because of spatial reuse under the protocol model. Hence, we define the per user throughput as follows,

$$T \triangleq \frac{F}{D}, \tag{1}$$

where D is the number of channel uses required to satisfy all user requests. The pair (M, T) is achievable if (M, R) is achievable and there exists a transmission policy Π such that the RF encoded bits are delivered to their destinations in $D \leq \frac{F}{T}$ channel uses. The optimal achievable throughput is defined as

$$T^*(M) \triangleq \sup\{T : (M, T) \text{ is achievable}\}.$$

Remark 1: When considering clustering and assuming that the transmission rate of each cluster is exactly R_{c}, we can obtain $D = \frac{R_{\mathrm{c}} F\mathcal{K}}{C_r}$. Therefore, the per user throughput for a clustering scheme is

$$T_{\mathrm{c}} = \frac{C_r}{R_{\mathrm{c}}\mathcal{K}}. \tag{2}$$

In the following, we will compare the per user throughput of clustering and non-clustering schemes.

III. Centralized Hypercube Coded Caching Approach

In this section, we outline the proposed hypercube scheme which is a new combinatorial design of a centralized D2D caching network. Moreover, we consider the case when the transmission range $r \geq \sqrt{2}$. In other words, a transmission from any user can be received by the rest of the users in the network and there is no user clustering. In the following, we present the achievable rate and packetization of the hypercube scheme and then provide a definition of the scheme and examples.

Theorem 1: Let m, n, M be the library size, number of users and the cache size per user, respectively. For $r \geq \sqrt{2}$ and $t, \frac{m}{M} \in \mathbb{Z}^+$ and $t \geq 2$, the following rate is achievable:

$$R^{\mathrm{hc}}(M) = \frac{t}{t-1}\left(\frac{m}{M} - 1\right) \tag{3}$$

with the requirement of $K^{\mathrm{hc}} = \left(\frac{m}{M}\right)^t$ when $\frac{\frac{m}{M}-1}{t-1} \in \mathbb{Z}^+$, or $K^{\mathrm{hc}} = (t-1)\left(\frac{m}{M}\right)^t$ when $\frac{\frac{m}{M}-1}{t-1} \notin \mathbb{Z}^+$. □

A. Proof of Theorem 1, the Hypercube Scheme

To prove Theorem 1 and explain the hypercube scheme, we use the following definitions.

Definition 4: **(Hypercube)** A hypercube with dimension $d \in \mathbb{Z}^+$ and side length $x \in \mathbb{Z}^+$ is the set of points $[x]^d$. The hypercube contains x^d unique points. ◊

Definition 5: **(Hyperplane)** A hyperplane, $\mathcal{P}_{y,i}$, normal to dimension $i \in [d]$ at position $y \in [x]$ is the set of points where the i-th dimension is fixed such that $\mathcal{P}_{y,i} = \{\boldsymbol{\xi} \in [x]^d : \xi_i = y\}$. The hypercube contains xd unique hyperplanes and each hyperplane contains $|\mathcal{P}_{y,i}| = x^{d-1}$ points. ◊

Definition 6: **(Line)** A line oriented to dimension $i \in [d]$ and passes through the point $\boldsymbol{\nu} \in [x]^d$ is the set of points $\mathcal{L}_{\boldsymbol{\nu},i} = \left\{\boldsymbol{\xi} \in [x]^d : \xi_j = \nu_j,\ \forall j \in [x] \setminus i\right\}$. Alternatively, a line is defined as the intersection of a set of $d-1$ perpendicular hyperplanes, $\mathcal{L}_{\boldsymbol{\nu},i} = \bigcap_{j\in[d]\setminus i} \mathcal{P}_{\nu_j,j}$. Given $i, j \in [d]$ and $\boldsymbol{\xi}, \boldsymbol{\nu} \in [x]^d$, two lines are equivalent, $\mathcal{L}_{\boldsymbol{\xi},i} = \mathcal{L}_{\boldsymbol{\nu},j}$, *if and only if* $i = j$ and $\xi_k = \nu_k$ for all $k \in [d]\setminus i$. The hypercube contains dx^{d-1} unique lines and each line contains $|\mathcal{L}_{\boldsymbol{\nu},i}| = x$ points. ◊

The placement phase of the Hypercube caching scheme is defined as follows.

1) Cache Placement Phase: Given that $t, \frac{m}{M} \in \mathbb{Z}^+$, define a hypercube, of dimension $d = t$ and side length $x = \frac{m}{M}$. Split each file W_f into a set of disjoint and equally-sized packets $\left\{W_{f,\boldsymbol{\nu}} : \boldsymbol{\nu} \in \left[\frac{m}{M}\right]^t\right\}$. Each point, $\boldsymbol{\nu} \in \left[\frac{m}{M}\right]^t$, represents a set of packets $\{W_{f,\boldsymbol{\nu}} : f \in [m]\}$. Each hyperplane, $\mathcal{P}_{y,i}$, represents the cache of a user so that there is a one-to-one mapping between the set of users and the hyperplanes. For all values $y \in \left[\frac{m}{M}\right]$ and dimensions $i \in [t]$, there is one user who caches

$$Z_{y,i} = \{W_{f,\boldsymbol{\xi}} : f \in [m],\ \boldsymbol{\xi} \in \mathcal{P}_{y,i}\}. \tag{4}$$

To validate this cache placement, we find that there are $t \cdot \frac{m}{M} = n$ hyperplanes and therefore a one-to-one mapping exists between the users and the hyperplanes. Next, each hyperplane contains $\left(\frac{m}{M}\right)^{t-1}$ points and, therefore, the number of bits cached at each user is

$$\frac{\left(\frac{m}{M}\right)^{t-1}}{\left(\frac{m}{M}\right)^{t}} \cdot mF = MF. \tag{5}$$

Example 1 Cache Placement: Consider a cache network of $n = 15$ users where each user is capable of storing $\frac{M}{m} = \frac{1}{5}$ of the file library and $t = 3$. We construct a hypercube of dimension $t = 3$ and side length $\frac{m}{M} = 5$. In this case, the hypercube is simply a cube and similarly, the hyperplanes are planes. The cube is the set of points $[5]^3$. We split each file into $5^3 = 125$ disjoint and equal-size packets, mapping each packet to a point of the cube. Each user caches a set of packets from every file represented by a plane. For example, the cache of one particular user is

$$\begin{aligned} Z_{2,1} &= \{W_{f,\boldsymbol{\nu}} : f \in [m],\ \boldsymbol{\nu} \in \mathcal{P}_{2,1}\} \\ &= \{W_{f,\boldsymbol{\nu}} : f \in [m],\ \nu_1 = 2\}. \end{aligned} \tag{6}$$

Then another user caches the packets of $Z_{4,1} = \{W_{f,\boldsymbol{\nu}} : f \in [m],\ \nu_1 = 4\}$ and has no cached packets in common with the user who caches $Z_{2,1}$ as their respectively planes are parallel.

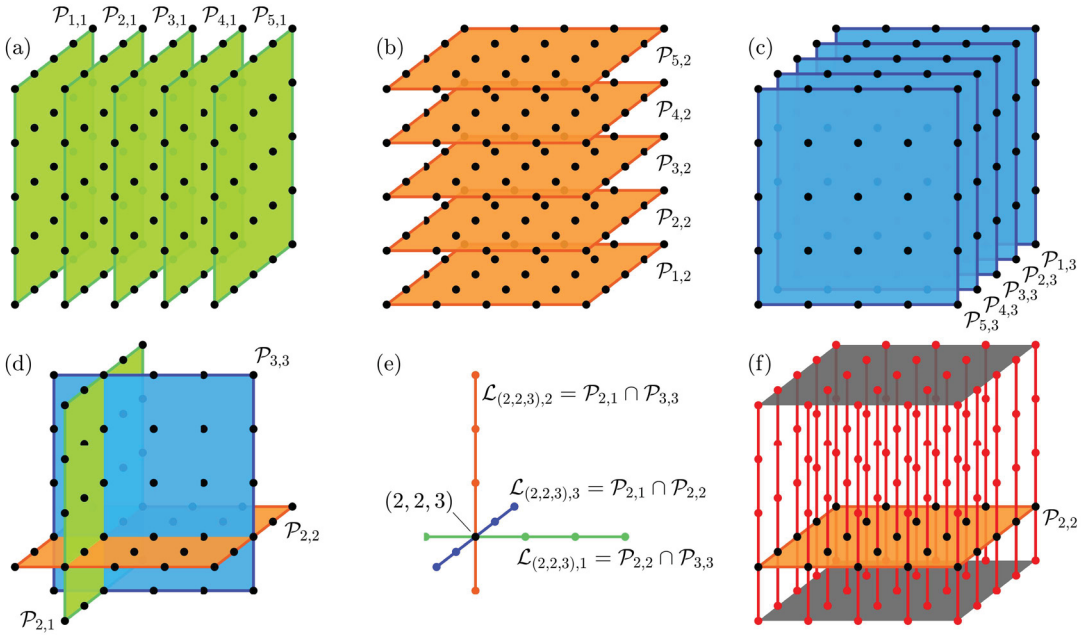


Fig. 2. Six representations of a hypercube of dimension 3 and side length 5. For the caching design, each point represents a set of packets, one from each file of the library. (a), (b) and (c) depict all the hyperplanes of the hypercube normal to the first, second and third dimension, respectively, of the hypercube. The hyperplanes represent user caches of Example 1. (d) Depiction of 3 orthogonal planes, representing the caches of the users in the multicast group of Example 2. (e) The lines formed by intersections of the hyperplanes of (d), representing the transmitted packets of Example 2. (f) All normal lines to the plane $\mathcal{P}_{2,2}$ which includes every point of the hypercube. The red points of each line represent a set of packets, $\mathcal{T}_{\boldsymbol{\nu},2}$, that the user who caches $Z_{2,2}$ will receive from a multicast group in the delivery phase to recover its requested file as described in Example 3.

However, even another user caches $Z_{3,3} = \{W_{f,\boldsymbol{\nu}} : f \in [m],\ \nu_3 = 3\}$ which is represented by a plane perpendicular to the users who cache $Z_{2,1}$ and $Z_{4,1}$ and therefore has some cached packets in common with these users. Each plane contains $\left(\frac{m}{M}\right)^{t-1} = 25$ points and therefore each user caches a $\frac{25}{125} = \frac{1}{5}$ fraction of each file and thusly $\frac{1}{5}$ of the entire library. The planes representing caches of the users defined by planes $\{\mathcal{P}_{y,1} : y \in [5]\}$, $\{\mathcal{P}_{y,2} : y \in [5]\}$ and $\{\mathcal{P}_{y,3} : y \in [5]\}$ are depicted in Fig. 2(a), Fig. 2(b) and Fig. 2(c), respectively. Each point in Fig. 2 represents a set of packets. △

2) Delivery Phase: For each point $\boldsymbol{\nu} \in \left[\frac{m}{M}\right]^t$ of the hypercube, we define a subset of t users

$$\mathcal{G}_{\boldsymbol{\nu}} = \{u_1, u_2, \cdots, u_t\} \tag{7}$$

as a multicast group where user $u_i \in \mathcal{G}_{\boldsymbol{\nu}}$ is the user that caches $Z_{\nu_i,i}$. The hyperplane cached by each user of $\mathcal{G}_{\boldsymbol{\nu}}$ is perpendicular to the hyperplane cached by every other user of $\mathcal{G}_{\boldsymbol{\nu}}$ since $\mathcal{G}_{\boldsymbol{\nu}}$ contains one user from each dimension. For ease of notation,[5] let user u_i request file W'_i and for all $i \in [t]$, define

$$\mathcal{T}_{\boldsymbol{\nu},i} = \left\{W'_{i,\boldsymbol{\xi}} : \boldsymbol{\xi} \in \mathcal{L}_{\boldsymbol{\nu},i} \setminus \boldsymbol{\nu}\right\} \tag{8}$$

as the set of requested packets by user u_i that will be received from the other users of $\mathcal{G}_{\boldsymbol{\nu}}$. The packet set, $\mathcal{T}_{\boldsymbol{\nu},i}$, includes the packets cached along the line $\mathcal{L}_{\boldsymbol{\nu},i}$ except the packet at $\boldsymbol{\nu}$. Also, $\mathcal{T}_{\boldsymbol{\nu},i}$ is not cached by user u_i, but is cached by every other user of $\mathcal{G}_{\boldsymbol{\nu}}$. Split each $\mathcal{T}_{\boldsymbol{\nu},i}$ into $t-1$ disjoint and equal size packet sets labeled as $\left\{\mathcal{T}^j_{\boldsymbol{\nu},i} : j \in [t] \setminus i\right\}$. The packets of $\mathcal{T}^j_{\boldsymbol{\nu},i}$ are included in a coded multicast from user u_j to serve the request of user u_i. Specifically, each user $u_j \in \mathcal{G}_{\boldsymbol{\nu}}$ broadcasts

$$\bigoplus_{i \in [t] \setminus j} \mathcal{T}^j_{\boldsymbol{\nu},i} \tag{9}$$

to the other users of $\mathcal{G}_{\boldsymbol{\nu}}$.

Example 2 Delivery Phase: Assume the cache placement from Example 1 and consider the point $\boldsymbol{\nu} = (2,2,3)$ and the user group $\mathcal{G}_{(2,2,3)} = \{u_1, u_2, u_3\}$. The cache of users u_1, u_2 and u_3 are represented by planes $\mathcal{P}_{2,1}$ (green plane), $\mathcal{P}_{2,2}$ (red plane), and $\mathcal{P}_{3,3}$ (blue plane), respectively, in Fig. 2(d). For multicast opportunities, we are interested in packets requested by one user and cached by the other two. This is analogous to points of the cube that lie on the intersection of two planes, but not the third. These points are highlighted in Fig. 2(e). For example, the packets requested by user u_2 from this multicast group are represented by the red line, $\mathcal{L}_{(2,2,3),2}$, with the exception of the point $(2,2,3)$ which intersects all three planes. Similarly, user u_1 requests the green line, $\mathcal{L}_{(2,2,3),1} \setminus (2,2,3)$, and user u_3 requests the blue line, $\mathcal{L}_{(2,2,3),3} \setminus (2,2,3)$, in Fig. 2(d). Without loss of generality, assume users u_1, u_2 and u_3 request files A, B and C, respectively. Then we define

$$\mathcal{T}_{(2,2,3),1} = \left\{A_{(1,2,3)}, A_{(3,2,3)}, A_{(4,2,3)}, A_{(5,2,3)}\right\} \tag{10}$$
$$\mathcal{T}_{(2,2,3),2} = \left\{B_{(2,1,3)}, B_{(2,3,3)}, B_{(2,4,3)}, B_{(2,5,3)}\right\} \tag{11}$$
$$\mathcal{T}_{(2,2,3),3} = \left\{C_{(2,2,1)}, C_{(2,2,2)}, C_{(2,2,4)}, C_{(2,2,5)}\right\} \tag{12}$$

[5]The file W'_i is a file in library set $\{W_1, \ldots, W_m\}$ defined in Section II. Note that, for different multicast groups, u_i may represent a different user in which case W'_i represents a different file since users make distinct requests.

which are based on the lines of Fig. 2(d). These sets are split up and the users transmit

$$\text{user } u_1: \quad B_{(2,1,3)} \oplus C_{(2,2,1)}, \quad B_{(2,3,3)} \oplus C_{(2,2,2)} \tag{13}$$
$$\text{user } u_2: \quad A_{(1,2,3)} \oplus C_{(2,2,4)}, \quad A_{(3,2,3)} \oplus C_{(2,2,5)} \tag{14}$$
$$\text{user } u_3: \quad A_{(4,2,3)} \oplus B_{(2,4,3)}, \quad A_{(5,2,3)} \oplus B_{(2,5,3)}. \tag{15}$$

△

The delivery phase is valid for the following reasons. Within each multicast group, defined by a point $\boldsymbol{\nu} \in \left[\frac{m}{M}\right]^t$, the users exchange packets represented by lines defined by the intersection of $t-1$ hyperplanes representing the users' caches. In other words, each packet of the set $\mathcal{T}_{\boldsymbol{\nu},i}$ is requested by user $u_i \in \mathcal{G}_{\boldsymbol{\nu}}$ and cached at all other users of $\mathcal{G}_{\boldsymbol{\nu}}$. Specifically,

$$\begin{aligned}\mathcal{T}_{\boldsymbol{\nu},i} &= \left\{W'_{i,\boldsymbol{\xi}} : \boldsymbol{\xi} \in \mathcal{L}_{\boldsymbol{\nu},i} \setminus \boldsymbol{\nu}\right\} \\ &= \left\{W'_{i,\boldsymbol{\xi}} : \boldsymbol{\xi} \in \bigcap_{k\in[t]\setminus i} \mathcal{P}_{\nu_k,k} \setminus \boldsymbol{\nu}\right\} \\ &\subset \left\{W_{f,\boldsymbol{\xi}} : f \in [m],\ \boldsymbol{\xi} \in \mathcal{P}_{\nu_j,j}\right\} = Z_{\nu_j,j} \end{aligned} \tag{16}$$

for all $j \in [t] \setminus i$ and $Z_{\nu_j,j}$ is the cache of user $u_j \in \mathcal{G}_{\boldsymbol{\nu}}$. Furthermore, $\mathcal{T}^j_{\boldsymbol{\nu},i} \subseteq \mathcal{T}_{\boldsymbol{\nu},i} \subset Z_{\nu_j,j}$ and user $u_j \in \mathcal{G}_{\boldsymbol{\nu}}$ is able to broadcast $\bigoplus_{i\in[t]\setminus j} \mathcal{T}^j_{\boldsymbol{\nu},i}$. Also, by a similar argument, user u_j can decode $\mathcal{T}^k_{\boldsymbol{\nu},j}$ from the transmission $\bigoplus_{i\in[t]\setminus k} \mathcal{T}^k_{\boldsymbol{\nu},i}$ from user u_k. This is true because $\bigcup_{i\in[t]\setminus j,k} \mathcal{T}_{\boldsymbol{\nu},i} \subset Z_{\nu_j,j}$ as we have shown in (16).

It remains to show that every user can recover the entirety of their requested file. To do this we use the following lemma.

Lemma 1: For all $i \in [t]$ and $y \in \left[\frac{m}{M}\right]$, the set of all points $\left[\frac{m}{M}\right]^t$ is equivalent to the union of all the lines normal to the plane $\mathcal{P}_{y,i}$,

$$\bigcup_{\boldsymbol{\nu}\in\mathcal{P}_{y,i}} \mathcal{L}_{\boldsymbol{\nu},i} = \left[\frac{m}{M}\right]^t. \tag{17}$$

□

An example of Lemma 1 can observed in Fig. 2(f) and the proof is presented below. *Proof:* The proof follows from the fact that all normal lines to a plane are parallel and each point is passed through by exactly one line. Notice that $\mathcal{L}_{\boldsymbol{\xi},i} = \mathcal{L}_{\boldsymbol{\nu},i}$ if $\xi_k = \nu_k$ for all $k \in [t] \setminus i$. Also, for any line, $\boldsymbol{\xi} \in \mathcal{L}_{\boldsymbol{\xi},i}$. Given any point $\boldsymbol{\xi} \in \left[\frac{m}{M}\right]^t$, define $\boldsymbol{\xi}' \in \left[\frac{m}{M}\right]^t$ such that $\xi'_k = \xi_k$ for all $k \in [t] \setminus i$ and $\xi'_i = y$. We find that $\mathcal{L}_{\boldsymbol{\xi}',i} = \mathcal{L}_{\boldsymbol{\xi},i}$ and $\boldsymbol{\xi} \in \mathcal{L}_{\boldsymbol{\xi}',i}$. Moreover, $\mathcal{L}_{\boldsymbol{\xi}',i} \in \{\mathcal{L}_{\boldsymbol{\nu},i} : \boldsymbol{\nu} \in \mathcal{P}_{y,i}\}$ and $\boldsymbol{\xi} \in \bigcup_{\boldsymbol{\nu}\in\mathcal{P}_{y,i}} \mathcal{L}_{\boldsymbol{\nu},i}$. ■

For the user who caches $Z_{y,i}$ and requests file W'_i, to recover the entirety of their requested file, they must have the set of packets $\left\{W'_{i,\boldsymbol{\nu}} : \boldsymbol{\nu} \in \left[\frac{m}{M}\right]^t\right\}$. Notice that this user participates in every multicast group $\mathcal{G}_{\boldsymbol{\nu}}$ such that $\boldsymbol{\nu} \in \mathcal{P}_{y,i}$. We have shown that this user $u_i \in \mathcal{G}_{\boldsymbol{\nu}}$ will receive and decode $\left\{W'_{i,\boldsymbol{\xi}} : \boldsymbol{\xi} \in \mathcal{L}_{\boldsymbol{\nu},i} \setminus \boldsymbol{\nu}\right\}$ from the multicast group $\mathcal{G}_{\boldsymbol{\nu}}$. Given that $\boldsymbol{\nu} \in P_{y,i}$, user u_i has $W'_{i,\boldsymbol{\nu}}$ locally cached and after the delivery phase has all requested packets from represented by the line $\mathcal{L}_{\boldsymbol{\nu},i}$. Considering all multicast groups that this user is a part of, $\{\mathcal{G}_{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \mathcal{P}_{y,i}\}$, this user has the set of packets

$$\left\{W'_{i,\boldsymbol{\xi}} : \boldsymbol{\xi} \in \bigcup_{\boldsymbol{\nu}\in\mathcal{P}_{y,i}} \mathcal{L}_{\boldsymbol{\nu},i}\right\}. \tag{18}$$

By Lemma 1, this is the set of all requested packets by this user.

Example 3 File Recovery: We build on Examples 1 and 2 and show that the user that caches $Z_{2,2}$ can recover all its requested file, B. For all $\boldsymbol{\nu} \in \mathcal{P}_{2,2}$, this user will participate in multicast group $\mathcal{G}_{\boldsymbol{\nu}}$. From $\mathcal{G}_{\boldsymbol{\nu}}$, this user will receive and decode

$$\begin{aligned}\mathcal{T}_{\boldsymbol{\nu},2} &= \{B_{\boldsymbol{\xi}} : \boldsymbol{\xi} \in \mathcal{L}_{\boldsymbol{\nu},2} \setminus \boldsymbol{\nu}\} \\ &= \left\{B_{(\nu_1,1,\nu_3)}, B_{(\nu_1,3,\nu_3)}, B_{(\nu_1,4,\nu_3)}, B_{(\nu_1,5,\nu_3)}\right\}. \end{aligned} \tag{19}$$

Moreover, this user has $B_{\boldsymbol{\nu}} = B_{(\nu_1,2,\nu_3)}$ locally cached and will have the set of all requested packets cached along the line $\mathcal{L}_{\boldsymbol{\nu},2}$. Therefore, after the delivery phase, this user has the set of requested packets represented by all lines normal to the plane $\mathcal{P}_{2,2}$

$$\{B_{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \mathcal{L}_{\boldsymbol{\nu},2},\ \forall \boldsymbol{\nu} \in \mathcal{P}_{2,2}\} = \left\{B_{\boldsymbol{\nu}} : \boldsymbol{\nu} \in [5]^3\right\} \tag{20}$$

which is the entirety of its request. This is shown in Fig. 2(f) where the red points of each line, $\mathcal{L}_{\boldsymbol{\nu},2}$, represent a different packet set $\mathcal{T}_{\boldsymbol{\nu},2}$. All of the red points of Fig. 2(f) represent the requested received packets and the black points of $\mathcal{P}_{2,2}$ represent the cached packets. Together, the red and black points cover all the points of the hypercube. △

The rate of the hypercube scheme can be resolved by recognizing there are $\left(\frac{m}{M}\right)^t$ multicast groups and within each group t users transmit $\frac{\frac{m}{M}-1}{t-1}$ coded packets of size $\frac{F}{\left(\frac{m}{M}\right)^t}$ bits. Therefore, the rate is

$$R^{\text{hc}} = \left(\frac{m}{M}\right)^t \cdot t \cdot \frac{\frac{m}{M}-1}{t-1} \cdot \frac{1}{\left(\frac{m}{M}\right)^t} = \frac{t}{t-1}\left(\frac{m}{M}-1\right). \tag{21}$$

The packetization from the cache phase is $\left(\frac{m}{M}\right)^t$. In many cases, no further packetization is necessary to execute the delivery phase different from the original D2D caching work [14]. Any additional packetization for the hypercube scheme occurs when splitting the packet set $\mathcal{T}_{\boldsymbol{\nu},i}$ into $t-1$ equal size sets. If $|\mathcal{T}_{\boldsymbol{\nu},i}| = |\mathcal{L}_{\boldsymbol{\nu},i}| - 1 = \frac{m}{M} - 1$ is not divisible by $t-1$, then extra packetization is needed. A simple solution is to further split each packet $t-1$ times. In this case the packetization is $K^{\text{hc}} = (t-1)\left(\frac{m}{M}\right)^t$.

Example 4 Rate and Packetization: Continuing Examples 1 through 3, we compute the rate and packetization of this caching scheme. The rate is computed by counting all multicast groups, the number of users per multicast group and the normalized bits of each transmission $R^{\text{hc}} = 5^3 \cdot 3 \cdot \frac{2}{5^3} = 6$. The packetization is $K^{\text{hc}} = 5^3 = 125$ which results from the cache phase. Note that, no additional packetization is necessary for the delivery phase. Comparing to the scheme of [14] with similar n, and $\frac{M}{m}$, the rate is $R' = 4$ with packetization of $K' = 1365$. The use of the hypercube scheme has significantly reduced the packetization despite a small increase in rate. △

This completes the proof of Theorem 1. ■

B. Comparison to State-of-the-Art D2D Caching Networks

As was proposed in [14], the achievable rate and packetization pair is

$$(R', K') = \left(\frac{m}{M}\left(1-\frac{M}{m}\right), t\binom{n}{t}\right) = \left(\frac{m}{M}-1, t\binom{n}{t}\right). \tag{22}$$

We first compare the packetization for constant m and M and use Stirling's formula to demonstrate that

$$\begin{aligned}\frac{K'}{K^{\mathrm{hc}}} &= \frac{t\binom{n}{t}}{(t-1)\left(\frac{m}{M}\right)^t} = \frac{n!t}{t!(n-t)!(t-1)\left(\frac{m}{M}\right)^t} \\ &= \Theta\left(\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2\pi\sqrt{t(n-t)}\left(\frac{t}{e}\right)^t\left(\frac{n-t}{e}\right)^{(n-t)}\left(\frac{m}{M}\right)^t}\right) \\ &= \Theta\left(\sqrt{\frac{n}{2\pi t(n-t)}}\cdot\frac{n^n}{n^t(n-t)^{(n-t)}}\right) \\ &= \Theta\left(\sqrt{\frac{1}{n}}\cdot\left(\frac{m}{m-M}\right)^{\left(1-\frac{M}{m}\right)n}\right)\end{aligned} \tag{23}$$

where we use the worst case packetization for the hypercube scheme when $\frac{\frac{m}{M}-1}{t-1} \notin \mathbb{Z}^+$. We find that the packetization of the hypercube scheme is exponentially less than the scheme of [14]. Furthermore, we are interested in comparing the rate of the two schemes. Hence, we obtain

$$\frac{R'(M)}{R^{\mathrm{hc}}(M)} = \frac{t-1}{t} = 1-\frac{m}{nM}. \tag{24}$$

We find that, given a constant m and M, the rate of the two schemes are asymptotically equivalent as the number of users, n, becomes large. The hypercube approach demonstrates a significant decrease in the number of packetizations, especially as the number of users becomes large. Furthermore, the rate only increases slightly, and for a large number of users the rate of the two schemes are essentially equivalent.

Another D2D coded caching scheme is the DPDA2 scheme of [31] published later than the current work in [34]. It is worth mentioning that the DPDA2 scheme is equivalent to the coded distributed computing design of [38]. The rate and packetization of this scheme is

$$(R^{\mathrm{DPDA}}, K^{\mathrm{DPDA}}) = \left(\frac{t}{t-1}\left(\frac{m}{M}-1\right), (t-1)\left(\frac{m}{M}\right)^{t-1}\right). \tag{25}$$

In fact, this scheme has the same rate as the hypercube scheme. Moreover, considering the worst case packetization of the hypercube scheme, the DPDA has less packetization by a constant factor of $\frac{m}{M}$.

IV. Decentralized Hypercube Coded Caching Approach

Recent work [32] showed that it is possible to translate a centralized shared-link caching scheme into a decentralized shared-link coded caching scheme. Here, the term "decentralized" implies that users independently and randomly cache a portion of the library. In this section, we expand the concept of [32] to D2D coded caching networks and translate the hypercube scheme from centralized to decentralized. Generally, D2D incurs an additional cost because each multicast serves one less user compared to the shared-link case [14]. However, in the decentralized setting, we find this additional cost can be eliminated when serving certain multicast groups. Moreover, we obtain a rate-packetization trade-off and expand the use of the hypercube scheme to more general parameters, n, m and M.

Our decentralized approach is summarized as follows. We consider the file partition and cache placement of the hypercube scheme designed for n' (dummy) users such that each of them caches a $\frac{M}{m}$ fraction of the content files, where $n' \ll n$.[6] This means that we partition each file into $K' = \left(\frac{m}{M}\right)^{t'}$ packets where $t' = \frac{n'M}{m}$. Then, we define n' sets of packets, $Z_{i,j}$ for $i \in \left[\frac{m}{M}\right]$ and $j \in [t']$. Each of the n users in the decentralized network uniformly at random and independently cache one of the n' packet sets defined by the centralized scheme. The probability of caching any packet set is $\frac{1}{n'}$. For delivery, users form groups of size n' users that collectively cache every packet set. In general, each user group will perform the centralized delivery scheme. However, to form each group, some previously satisfied users may need to be included. In this case, the users perform only some of the transmissions from the centralized design as requests from the previously satisfied users are ignored. Then, we find that for some multicast groups, satisfied users can send transmissions to unsatisfied users with the same efficiency of the shared-link case in [32], closing the gap between the D2D and shared-link settings. The following example demonstrates this process.

A. An Example

In this example, consider a decentralized network of $n = 32$ users and each user caches $\frac{M}{m} = \frac{1}{3}$ of the library. Let $n' = 6$ and $t' = \frac{n'M}{m} = 2$. There are 6 packet sets defined by a 2-dimensional hypercube (plane) where the hyperplanes are lines in this case. The number of packets per file for this decentralized network is $K' = (\frac{m}{M})^{t'} = 9$. The packet sets, corresponding to each row or column of the hypercube, are labeled as $Z_{i,j}$ where $i \in [3]$ and $j \in [2]$. Each of the 32 users independently caches one of these sets at random using a uniform distribution. A possible outcome of the random caching is shown in Fig. 3(a) where 7 users cache $Z_{1,1}$, 6 users cache $Z_{2,1}$, 6 users cache $Z_{1,2}$, 3 users cache $Z_{2,2}$, etc.

To satisfy the 32 distinct user requests, there are multiple delivery phases which are similar to the centralized delivery design. The users form multicast groups depicted in Fig. 3 by points of the hypercube. Different from the centralized scheme, a multicast group may contain a both an unsatisfied user and a previously satisfied user and are outlined with a blue square in Fig. 3. Alternatively, a multicast group may contain only satisfied users which is marked by a red "X" in Fig. 3.

The delivery phases are as follows. First, observe that any set of 6 users, $\mathcal{U}_D$, that collectively cache every packet set,

[6]Note that n' is only a dummy variable to define the packet sets for the n users in the decentralized network to cache.

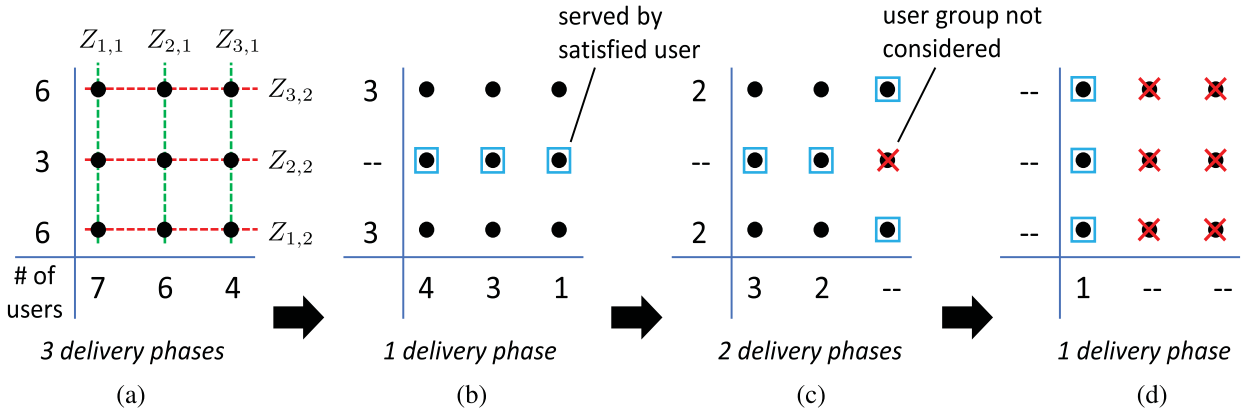


Fig. 3. A hypercube of dimension $t' = 2$ and side length $\frac{m}{M} = 3$. Each point represents a packet for every file. The numbers to the left and bottom are the number of remaining unsatisfied users who cache the rows and columns respectively. The points also represent multicast groups of the delivery phases. Multicast groups that contain at least one satisfied user or contain no unsatisfied users are represented by the blue squares and red "X"s, respectively.

$Z_{i,j}$, can perform the centralized delivery scheme. In this example, we pick 3 disjoint sets of users that cache all packets sets to perform the centralized delivery phase. After this, there are 14 remaining unsatisfied users as shown in Fig. 3(b) and no unsatisifed users that cache $Z_{2,2}$. Of these remaining users, we consider a set of 5 users who collectively cache every packet set except $Z_{2,2}$. These 5 users can perform a centralized delivery scheme by including one previously satisfied user who caches $Z_{2,2}$. The previously satisfied user does not request any packets. Within each multicast group that it belongs to, it only transmits packets to the unsatisfied users. The transmissions of hypercube delivery that serve the previously satisfied users are not necessary, effectively reducing the rate.

In the third phase, as shown in Fig. 3(c), of the remaining 9 unsatisfied users, we consider 2 non-overlapping sets of 4 users who collectively cache the packet sets $Z_{1,1}$, $Z_{2,1}$, $Z_{1,2}$ and $Z_{3,2}$. These user sets can perform the delivery phases by including previously satisfied users that cache $Z_{3,1}$ and $Z_{2,2}$. In this particular example, there is one multicasting group marked by the red "X" in Fig. 3(c) that does not need to be considered since both users in this multicasting group are satisfied. Similarly, after the delivery shown in Fig. 3(c), we are left with a single user (but forming 3 multicast groups) who can obtain the requested packets from the 3 users caching these packets.

The transmission rate of this example is computed by counting the number of multicasting groups in each phase that either include or do not include a previously satisfied user. The equivalent number of files transmitted by a multicast group with only unsatisifed users is $\frac{4}{9}$. For a group that contains both a satisfied and unsatisfied user, the equivalent number of files transmitted is $\frac{2}{9}$. Therefore, the rate is

$$R' = (3 \cdot 9 + 6 + 2 \cdot 4 + 0) \cdot \frac{4}{9} + (0 + 3 + 2 \cdot 4 + 3) \cdot \frac{2}{9} = \frac{64}{3}. \tag{26}$$

This decentralized scheme does not require any additional packetization beyond the placement phase. Interestingly, the transmission rate of this example is exactly that of conventional unicasting with $n\left(1 - \frac{M}{m}\right) = \frac{64}{3}$. This occurs because there is only unicasting for the 2-dimensional hypercube approach, which was chosen for ease of disposition. Next, we demonstrate how this approach applies to coded delivery.

B. General Decentralized Algorithm

The D2D hypercube decentralized caching design is defined as follows.

1) Cache Placement Phase: Let $n' \ll n$ such that $n' \in \mathbb{Z}^+$ and $t' = \frac{n'M}{m} \in \mathbb{Z}^+$. Define a hypercube of dimension t' and side length $\frac{m}{M}$. Split each file into $\left(\frac{m}{M}\right)^{t'}$ packets and define n' packet sets, $\left\{Z_{i,j} : i \in \left[\frac{m}{M}\right], j \in [t']\right\}$, based on the hypercube cache design. Each of the n users independently and randomly cache one of the n' packet sets with equal probability $\frac{1}{n'}$.

2) Delivery Phase: Initialize the set of unsatisfied users, $\bar{\mathcal{U}}_S$, to be the set of all n users, $\mathcal{U}$, and initialize the set of satisfied users, $\mathcal{U}_S$, to be the empty set. While $\bar{\mathcal{U}}_S$ is not empty, do the following. Let $\mathcal{U}_D \subseteq \bar{\mathcal{U}}_S$ be any largest subset of unsatisfied users that cache different packet sets, $Z_{i,j}$. For each $\boldsymbol{\nu} \in \left[\frac{m}{M}\right]^{t'}$, define a multicast group of t' users

$$\mathcal{G}_{\boldsymbol{\nu}} = \{u_1, \cdots, u_{t'}\} \subset (\mathcal{U}_D \cup \mathcal{U}_S) \tag{27}$$

where user u_j caches $Z_{\nu_j,j}$ and $u_j \in \mathcal{U}_S$ if and only if the packet set $Z_{\nu_j,j}$ is not cached by any user of $\mathcal{U}_D$. Then, if $\mathcal{G}_{\boldsymbol{\nu}} \cap \mathcal{U}_S = \emptyset$, the multicast group, $\mathcal{G}_{\boldsymbol{\nu}}$, performs the transmissions by the centralized hypercube design. Otherwise, if $\mathcal{G}_{\boldsymbol{\nu}}$ contains any user from $\mathcal{U}_S$, then any user $u' \in \mathcal{G}_{\boldsymbol{\nu}} \cap \mathcal{U}_S$ transmits

$$\bigoplus_{i:u_i \in (\mathcal{G}_{\boldsymbol{\nu}} \cap \mathcal{U}_D)} \mathcal{T}_{\boldsymbol{\nu},i} \tag{28}$$

where $\mathcal{T}_{\boldsymbol{\nu},i}$ is defined as the set of packets user u_i receives from multicast group $\mathcal{G}_{\boldsymbol{\nu}}$ in the centralized hypercube design. Note that if $\mathcal{G}_{\boldsymbol{\nu}} \cap \mathcal{U}_D = \emptyset$, there is no transmission. After iterating through all multicast groups $\left\{\mathcal{G}_{\boldsymbol{\nu}} : \boldsymbol{\nu} \in \left[\frac{m}{M}\right]^{t'}\right\}$, update the set of unsatisfied users as $\bar{\mathcal{U}}_S \leftarrow \bar{\mathcal{U}}_S \setminus \mathcal{U}_D$ and the set of

Algorithm 1 Decentralized Hypercube Coded Caching Delivery Design

Input: $\mathcal{U}$, $\frac{M}{m}$, t'
$\bar{\mathcal{U}}_S \leftarrow \mathcal{U}$, $\mathcal{U}_S \leftarrow \emptyset$.
while $\bar{\mathcal{U}}_S$ is not empty **do**
  $\mathcal{U}_D \leftarrow$ any largest subset of $\bar{\mathcal{U}}_S$ such that each user $u \in \mathcal{U}_D$ has a unique cache set
  for $\boldsymbol{\nu} \in \left[\frac{m}{M}\right]^{t'}$ **do**
    $\mathcal{G}_{\boldsymbol{\nu}} \leftarrow \{u_1, \cdots, u_{t'}\} \subset (\mathcal{U}_D \cup \mathcal{U}_S)$ such that u_j caches $Z_{\nu_j,j}$ and $u_j \in \mathcal{U}_S$ if and only if the packet set $Z_{\nu_j,j}$ is not cached by any user of $\mathcal{U}_D$
    if $\mathcal{G}_{\boldsymbol{\nu}} \cap \mathcal{U}_S = \emptyset$ **then**
      Users of $\mathcal{G}_{\boldsymbol{\nu}}$ perform the centralized delivery method for this multicasting group
    else
      any user of $\mathcal{G}_{\boldsymbol{\nu}} \cap \mathcal{U}_S$ transmits $\bigoplus_{i:u_i \in (\mathcal{G}_{\boldsymbol{\nu}} \cap \mathcal{U}_D)} \mathcal{T}_{\boldsymbol{\nu},i}$
    end if
  end for
  $\bar{\mathcal{U}}_S \leftarrow \bar{\mathcal{U}}_S \setminus \mathcal{U}_D$, $\mathcal{U}_S \leftarrow \mathcal{U}_S \cup \mathcal{U}_D$.
end while

satisfied users as $\mathcal{U}_S \leftarrow \mathcal{U}_S \cup \mathcal{U}_D$. The delivery scheme ends when $\bar{\mathcal{U}}_S = \emptyset$, otherwise, $\mathcal{U}_D$ is redefined and the process is repeated. The decentralized delivery design is outlined in Algorithm 1.

This decentralized scheme is valid since each user caches $\left(\frac{m}{M}\right)^{t'-1}$ of $\left(\frac{m}{M}\right)^{t'}$ packets from each file and therefore an $\frac{M}{m}$ fraction of the library. For each delivery set of users, $\mathcal{U}_D$, each user receives and decodes all requested packets as they would if they performed the centralized delivery scheme.

Remark 2: During delivery phase, we might not be able to form multicast groups entirely from unsatisfied users of $\mathcal{U}_D \subseteq \bar{\mathcal{U}}_S$. Hence, satisfied users from $\mathcal{U}_S$ are necessary. Note that a single satisfied user of a multicasting group can serve all requests of unsatisfied users from that multicasting group. This is possible since by the centralized caching design, each user requests a packet from a multicasting group such that this packet is locally cached by every other user of the multicasting group.

C. Performance

Performance of the decentralized design is given in the following theorem.

Theorem 2: Let m, n, M be the library size, number of users and the cache size per user, respectively. Assume that $r \geq \sqrt{2}$ and $\frac{m}{M}, t' \in \mathbb{Z}^+$ where $t' = \frac{n'M}{m} \geq 2$, and the required number of packets per file is at most $K' = (t'-1)\left(\frac{m}{M}\right)^{t'}$. Furthermore, if $n \geq \beta n' \log n'$ for some $\beta > 1$, then all the packet sets can be cached in the network with probability $1 - {n'}^{1-\beta}$, and the following transmission rate is achieved with probability $1 - o(1)$ as $n, n' \to \infty$,

$$R_{\mathrm{d}}^{\mathrm{hc}}(M) \leq k_\alpha \frac{t'}{t'-1}\left(\frac{m}{M} - 1\right), \tag{29}$$

where

$$k_\alpha = \begin{cases} (d_\beta - 1 + \alpha)\log n', \\ \quad \text{if } n = \beta n' \log n', \\ \left(\frac{n}{n'} + \alpha\sqrt{\frac{2n}{n'}\log n'}\right), \\ \quad \text{if } \omega\left(n' \log n'\right) = n \leq n \cdot \mathrm{polylog}(n), \\ \left(\frac{n}{n'} + \sqrt{\frac{2n\log n'}{n'}}\left(1 - \frac{1}{\alpha}\frac{\log\log n'}{2\log n'}\right)\right), \\ \quad \text{if } n = \omega\left(n\left(\log n\right)^3\right). \end{cases} \tag{30}$$

Here, $\alpha > 1$ and d_β (a number depending only on β) are constants that define the parameter regimes of (30) and $\mathrm{polylog}(n)$ denotes the class of functions $\bigcup_{k\geq 1} O\left((\log n)^k\right)$ as in [39, Theorem 1]. □

Proof: First, we show that with probability $1 - {n'}^{1-\beta}$ all packet sets are cached in the D2D network. Since each packet set is uniformly selected by each user at random, it reduces to a "bin-ball" problem [39] where the packet sets are the n' bins and users are the n balls. Hence, we need to characterize the probability that no bin is empty. Let Y_k denote the event that the k bin is empty, then we can compute

$$\mathbb{P}\left(Y_k\right) = \left(1 - \frac{1}{n'}\right)^n \leq e^{-n/n'}. \tag{31}$$

Hence, the probability that all bins are non-empty and each packet set is cached at least once is

$$\begin{aligned} \mathbb{P}\left(\cap_k \overline{Y_k}\right) &= \mathbb{P}\left(\overline{\cup_k Y_k}\right) = 1 - \mathbb{P}\left(\cup_k Y_k\right) \\ &\geq 1 - \sum_{k=1}^{n'} \mathbb{P}\left(Y_k\right) \geq 1 - n' e^{-n/n'} \\ &\geq 1 - n' e^{-\beta n' \log n'/n'} = 1 - {n'}^{1-\beta}. \end{aligned} \tag{32}$$

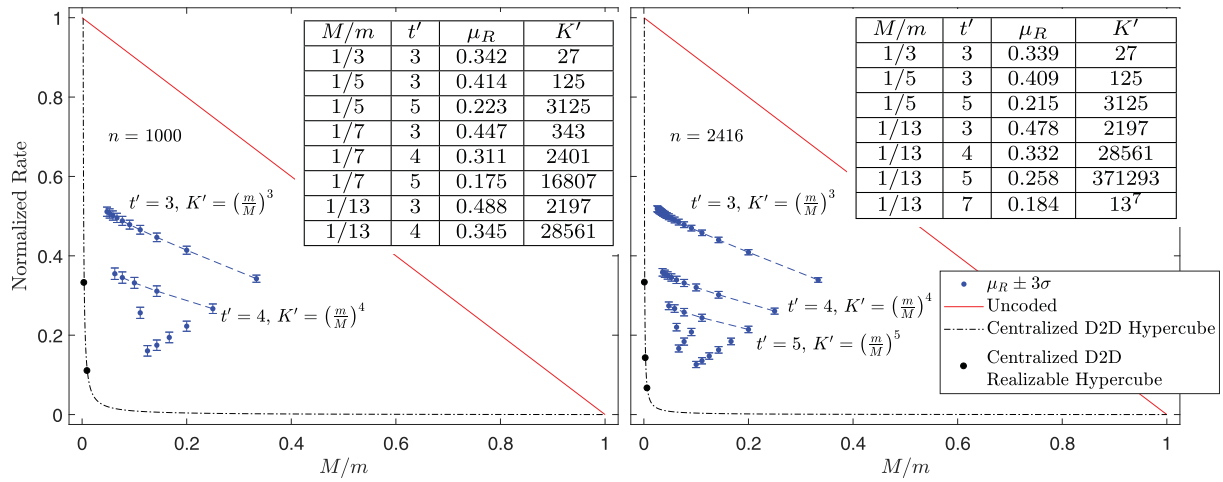
Second, we will show (29) holds with probability $1 - o(1)$. Let L denote the maximum number of users that cache the same packet set, or equivalently, the maximum number of balls in each bin. For instance, $L = 7$ in the example of Section IV-A. An upper bound on the rate can be computed by assuming that users perform L centralized delivery phases. The key is to find k_α such that the tail estimate $\mathbb{P}\left(\{L \leq k_\alpha\} \cap \left\{\cap_k \overline{Y_k}\right\}\right) = 1 - o(1)$. Hence, by using (3), we can obtain $R_{\mathrm{d}}^{\mathrm{hc}}(M) \leq L\frac{t'}{t'-1}\left(\frac{m}{M} - 1\right) \leq k_\alpha \frac{t'}{t'-1}\left(\frac{m}{M} - 1\right)$ with probability $1 - o(1)$. From Theorem 1 in [39], we can show that if k_α, $\alpha > 1$ satisfies (30), then we must have $\mathbb{P}\left(\{L \leq k_\alpha\}\right) = 1 - o(1)$. Furthermore, since

$$\begin{aligned} &\mathbb{P}\left(\{L \leq k_\alpha\}\right) \\ &= \mathbb{P}\left(\{L \leq k_\alpha\} \cap \left\{\cap_k \overline{Y_k}\right\}\right) + \mathbb{P}\left(\{L \leq k_\alpha\} \cap \{\cup_k Y_k\}\right) \\ &= \mathbb{P}\left(\{L \leq k_\alpha\} \cap \left\{\cap_k \overline{Y_k}\right\}\right) \\ &\quad + \mathbb{P}\left(\{\cup_k Y_k\}\right)\mathbb{P}\left(\{L \leq k_\alpha\} \mid \{\cup_k Y_k\}\right) \\ &\leq \mathbb{P}\left(\{L \leq k_\alpha\} \cap \left\{\cap_k \overline{Y_k}\right\}\right) + {n'}^{1-\beta}, \end{aligned} \tag{33}$$

we obtain

$$\begin{aligned} \mathbb{P}\left(\{L \leq k_\alpha\} \cap \left\{\cap_k \overline{Y_k}\right\}\right) &\geq \mathbb{P}\left(\{L \leq k_\alpha\}\right) - {n'}^{1-\beta} \\ &= 1 - o(1). \end{aligned} \tag{34}$$

This completes the proof. ■



M/m	t'	μ_R	K'
1/3	3	0.342	27
1/5	3	0.414	125
1/5	5	0.223	3125
1/7	3	0.447	343
1/7	4	0.311	2401
1/7	5	0.175	16807
1/13	3	0.488	2197
1/13	4	0.345	28561

M/m	t'	μ_R	K'
1/3	3	0.339	27
1/5	3	0.409	125
1/5	5	0.215	3125
1/13	3	0.478	2197
1/13	4	0.332	28561
1/13	5	0.258	371293
1/13	7	0.184	13^7

Fig. 4. Results of the hypercube approach decentralized D2D caching network simulations for $n = 1000$ users (left plot) and $n = 2416$ users (right plot). The mean rate $\mu_R = \mathbb{E}\left[R_{\rm d}^{\rm hc}(M)\right]$ plus or minus 3 times the standard deviation, $\mu_R \pm 3\sigma$, for a variety of hypercube constructions are compared to the D2D uncoded rate and hypercube centralized rate.

Remark 3: Note that, the requirement that each packet set is cached at least once is more restrictive than the general requirement that each packet is cached at least once across the network. As discussed, our decentralized scheme does not operate when a packet set is missing, even though every packet may be present in the network. However, with some changes to the delivery design (not shown here), the scheme could operate under this scenario.

D. Simulation Methods and Results

Decentralized networks are simulated with the number of users, n, equal to 1000 and 2416 users. For these networks we simulate a decentralized cache scheme for all parameter pairs, $n', t' \in \mathbb{Z}^+$, such that $3 \leq t' \leq 8$ and $9 \leq n' \leq 64$ for the network of 1000 users and $3 \leq t' \leq 10$ and $9 \leq n' \leq 120$ for the network of 2416 users, and $\frac{\frac{m}{M}-1}{t'-1} \in \mathbb{Z}^+$ holds, such that the packetization is $K' = \left(\frac{m}{M}\right)^{t'}$. For each parameter pair, (n', t'), 10^4 decentralized cache placement and delivery phases are simulated. Given that $n' \ll n$ in all cases, there were no instances of a simulated decentralized network where a packet set was not cached at least once. The mean and standard deviation of the rate, μ_R and σ, respectively, are depicted in Fig. 4. The results are compared to the uncoded rate $R^{\rm u} = n\left(1 - \frac{M}{m}\right)$ (red line) and the coded rate of the centralized hypercube scheme $R^{\rm hc} = \left(1 - \frac{m}{nM}\right)^{-1} \cdot \left(\frac{m}{M} - 1\right)$ (black dashed line).

From these results, we see that the decentralized hypercube scheme outperforms the uncoded scheme. On both plots, the set of points with a fixed t' is highlighted. As t' increases, the rate comes closer to the centralized coded rate. However, the number of packets per file, K', increases exponentially. This result demonstrates that there is a trade-off in designing decentralized D2D caching networks using the hypercube approach. Specifically, we can increase the number of packets to reduce the rate and vice versa. This may provide the flexibility to yield a practical amount of packetization while having limited impact on the transmission rate. Furthermore, we see that the decentralized approach yields more realizable constructions of the hypercube. In fact, as shown in Fig. 4, the number of realizable hypercube schemes for 1000 and 2416 users are 2 and 3 respectively ($t = 4, 10$ for $n = 1000$ and $t = 4, 8, 16$ for $n = 2416$). In comparison, the decentralized network design provides many more possibilities.

V. Ruzsa-Szeméredi Graph Coded Caching Approach

While the hypercube approach requires significantly (in fact, exponentially) less number of packets per file compared to [14], the hypercube approach still yields an exponential number of packets relative to n if m and M are fixed. While the hypercube approach certainly increases the domain for which a D2D caching network is implementable, it is still an open question as to whether there exists a coded D2D caching network scheme without spatial reuse ($r > \sqrt{2}$) with a sub-exponential number of packets per file. Motivated by [29], in this section we propose a coded D2D caching scheme based on Ruzsa-Szeméredi graphs which requires only a sub-quadratic packetization. While the general, expandable scheme only holds for arbitrarily large n, this work demonstrates that sub-quadratic D2D caching schemes exist.

A. Ruzsa-Szeméredi Graphs

In this section, we focus on a specific Ruzsa-Szeméredi graph design, which was first introduced in [27] and used in a novel manner in [29] to construct a cache placement and coded multicasting scheme for a shared-link caching network. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ is the edge set. We introduce the following definitions [28], [29].

Definition 7: Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, a matching $\mathcal{M}$ in $\mathcal{G}$ is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex. ◊

Definition 8: The set of edges $\mathcal{M} \in \mathcal{E}$ is an induced matching if for the set $\mathcal{S}$ of all the vertices incident on the

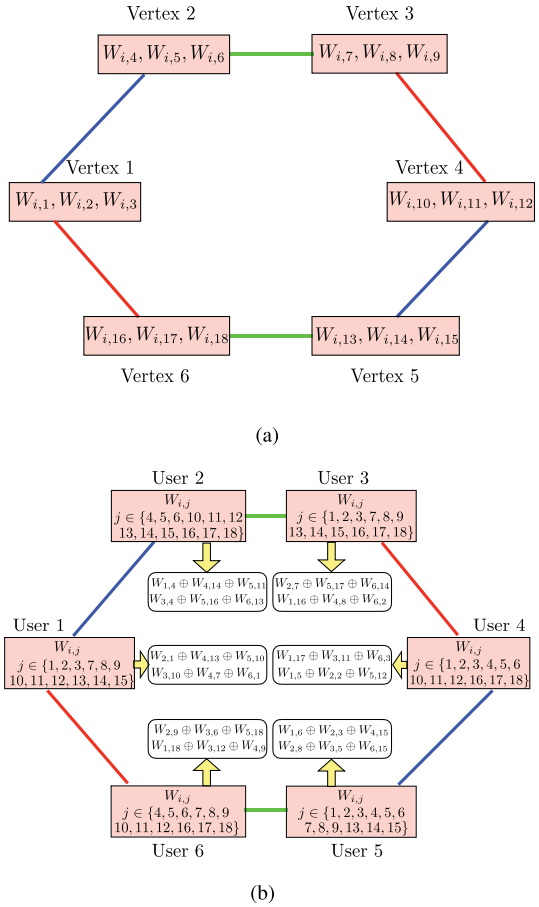


Fig. 5. a) A $(2,3)$-Ruzsa-Sezmeredi graph with 6 vertices which represents a coded caching scheme where $n = 6$, $\frac{M}{m} = \frac{2}{3}$, and $K = n(2\gamma - 1) = 3n = 18$. The graph can be split into $\tau = 3$ induced pairwise disjoint matchings denoted by different colors in the graph, which cover all edges of the original graph. b) An example of the proposed scheme in a D2D network with $n = m = 6, K = 18$ and $M = 4$. Each vertex represents a user. The content of each rectangle represents cached packets. The coded multicast packets are shown in the center of the figure.

edges in $\mathcal{M}$, the induced graph $\mathcal{G}_{\mathcal{S}}$ contains no other edges apart from those in the matching $\mathcal{M}$. $\Diamond$

Definition 9: A disjoint matching $\mathcal{M}$ is a matching such that any pair of the edges in $\mathcal{M}$ are not adjacent to any third edge in $\mathcal{E}$. $\Diamond$

Definition 10: A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is called an $(\gamma, \tau)-$Ruzsa-Szeméredi (RS) Graph if its set of edges consists of τ pairwise disjoint induced matching, each of size γ. $\Diamond$

An example of RS graph is shown in Fig. 5(a), which is also used in [29]. We will apply Ruzsa-Szeméredi graph to construct a cache placement and a coded multicasting scheme in wireless D2D caching networks. In this section, we will focus on the case that a transmission from any user can be received and successfully decoded by all users in the network (e.g., $r \geq \sqrt{2}$). In the remainder of this section, we will first introduce a motivating example and then present the general achievable scheme.

B. An Example

Similar to [29], our proposed scheme is based on RS Graphs. Here, let $m = n = 6$ and $M = 4$. Each file is partitioned into $K = 3n = 18$ packets with F/K bits each. Assume that user u requests file W_u. To find the cache placement and the coded multicasting scheme, we build a $(2,3)$-Ruzsa-Szeméredi graph $\mathcal{G}$ with 6 vertices as shown in Fig. 5(a) where each vertex is represented by 3 packets. In our proposed achievable scheme, we partition each file into $n(2\gamma - 1) = 18$ packets and denote the jth packet from file i by $W_{i,j}, i \in [6], j \in [18]$. The cache placement and coded multicasting scheme are shown in Fig. 5(b), where each vertex represents a user and his cached packets are shown inside each vertex. For instance, the most left vertex, vertex 1, represents user 1 and his cached packets, which are $W_{i,j}$, where $i \in \{1, \cdots, 6\}$ and $j \in \{1,2,3,7,8,9,10,11,12,13,14,15\}$. Note that these exclude the packets represented by its neighboring vertices, vertex 2 and vertex 6, which are shown in Fig. 5(b).

The coded multicasting scheme is shown in the center of Fig. 5(b). For example, user 1 transmits $W_{2,1} \oplus W_{4,13} \oplus W_{5,10}$, which are used by users 2, 4, 5 to decode packets $W_{2,1}, W_{4,13}, W_{5,10}$, respectively. In particular, it can be seen that user 2 has $W_{4,13}$ and $W_{5,10}$ such that he can decode $W_{2,1}$. Similarly, user 4 and 5 can also decode the corresponding packet in a similar manner. In this example, if we use the achievable scheme proposed in [14], each file is split into $t\binom{n}{t} = 4\binom{6}{4} = 60$ packets and the transmission rate is given by $\frac{m}{M} - 1 = \frac{6}{4} - 1 = \frac{1}{2}$. While using the proposed scheme, the transmission rate is $\frac{12}{18} = \frac{2}{3}$. Therefore, the proposed scheme requires $K = 18$ instead of $K = 60$ as the scheme in [14], and the achievable transmission rate of the proposed scheme is $\frac{2}{3}$ instead of $\frac{1}{2}$ achived by the scheme in [14]. Hence, we observe that while sacrificing the transmission rate by $(\frac{2}{3} - \frac{1}{2})/\frac{1}{2} = \frac{1}{3} = 33\%$, the proposed scheme reduces the number of packets needed per file by $\frac{60-18}{60} = \frac{7}{10} = 70\%$.

C. General Achievable Scheme

In this section, we generalize the deterministic caching and coded delivery scheme illustrated in Section V-B to the general case of m, n and M.

1) Building the Ruzsa-Szeméredi graph: We build a Ruzsa-Szeméredi graph with n vertices, each with a degree of $n - t$. This graph consists of τ pairwise disjoint induced matchings and each has γ edges.

2) Cache Placement Phase: The cache placement scheme is closely related to the scheme in [29] and is designed according to the Ruzsa-Szeméredi graph. Each file is divided into $K = n(2\gamma - 1)$ packets, labeled by $\{W_{i,j} : i \in [m], j \in [K]\}$. Each vertex in the RS graph represents $2\gamma - 1$ distinct packets. User u caches packets corresponding to those vertices that are not adjacent to itself and packets corresponding to the vertex itself.

3) Delivery Phase: As a consequence of the caching scheme described above, any subset of $2\gamma - 1$ users belonging to a disjoint matching of size γ in $\mathcal{U} = \{1, \ldots, n\}$ has the property that they share $2\gamma - 1$ packets from each file. Consider one such subset. For any file requested by the remaining (2γ)-th user, by construction, there are $2\gamma - 1$ packets shared by the other $2\gamma - 1$ users and needed by the (2γ)-th user. Therefore, each user in every disjoint matching

has $2\gamma-1$ packets that are useful for the remaining $2\gamma-1$ users. Furthermore, such sets of packets are disjoint (empty pairwise intersections). For delivery, in each disjoint matching, each user computes the XOR of its $2\gamma-1$ useful packets and multicasts it to all other users in this disjoint matching. In this way, for every multicast transmission exactly $2\gamma-1$ users will be able to decode a useful packet using "interference cancellation" based on their cached content.

The achievable rate is given in the following.

Theorem 3: Let m, n, M be the library size, number of users and the cache size per user, respectively. Assume $n=\Lambda^z$, where $\Lambda, z \in \mathbb{Z}^+$ such that $z \geq 2\Lambda$, then there exists a (τ,γ)-RS graph with n vertices where $\tau = n^{\left(1+2\frac{\ln 10.5}{\ln \Lambda}+o(1)\right)}$ and $\gamma < n$. Using the (τ,γ)-RS graph to define a caching scheme for $r \geq \sqrt{2}$ yields the following achievable rate:

$$R^{\mathrm{RS}}(M) \leq \frac{\tau}{n} \cdot \frac{2\gamma}{2\gamma-1} \tag{35}$$

where users cache an equivalent of

$$M = 2mn^{-\frac{1}{2\Lambda^4 \ln \Lambda}} \tag{36}$$

files and each file is split into at $K^{\mathrm{RS}} = (2\gamma-1)n$ packets. □

Proof: Theorem 3 is proved in Appendix A. ■

When we consider the asymptotic regime as n becomes large, we obtain the following corollary.

Corollary 1: For $r \geq \sqrt{2}$, when $n \to \infty$, $K = O\left(n^{2-\delta}\right)$ and $M = 2mn^{-c_1\delta\exp(-c_2/8)}$, where c_1, c_2 are some positive constants, let $t = \frac{nM}{m} \in \mathbb{Z}^+$, the following rate is achievable:

$$R^{\mathrm{RS}}(M) \leq n^{\delta} + o(n^{\delta}), \tag{37}$$

where $\delta = \frac{2\ln 10.5}{\ln \Lambda}$. □

Corollary 1 can be proved using the value of τ given in Appendix A and the relation $\gamma = O\left(n/\tau\right)$. From Corollary 1, we can see that when Λ is large enough, or equivalently, n is large enough, δ can be arbitrarily small. In other words, it can be computed from (37) that $R^{\mathrm{RS}}(M) = (10.5)^{2\log_{\Lambda} n}$. The throughput achieved by the proposed achievable scheme is given as follows.

Corollary 2: Let C_r be the constant link rate under the protocol model. For $r \geq \sqrt{2}$, the per user throughput is given by

$$T^{\mathrm{RS}}(M) = \frac{C_r}{R^{\mathrm{RS}}(M)} \overset{n\to\infty}{=} C_{\sqrt{2}} \cdot n^{-\delta} + o\left(n^{-\delta}\right), \tag{38}$$

where $R^{\mathrm{RS}}(M)$ and δ are given by (35) or (37), is achievable. □

Proof: This corollary can be proved by using the same procedure as the proof of Corollary 1. ■

D. Comparison to Other Schemes

Due to the sub-quadratic packetization with the number of users n, which is significantly lower than that of the hypercube based approach proposed in Section III, the achievable rate of the RS graph based design given by (35) and (37) for $r \geq \sqrt{2}$ is obviously worse than the achievable rate of the hypercube based design given in (3) and the original design [14], when n increases and m and M are fixed. However, there is still a significant gain in terms of transmission rate compared to the conventional uncoded unicasting scheme in some parameter regimes. In the following, we compare the transmission rates of the uncoded scheme, R^{u}, and the RS Graph scheme, R^{RS} where m, M and n are the same for both schemes. By solving (36) for n we obtain

$$n = \left(2\frac{m}{M}\right)^{2\Lambda^4 \ln \Lambda} \tag{39}$$

which better defines the constraint relating n to M/m for the this scheme based on the proposed RS graph design. For the purposes of comparison, (39) holds for both the uncoded and RS schemes. The transmission rate for the uncoded scheme is

$$R^{\mathrm{u}} = n\left(1-\frac{M}{m}\right) = \left(2\frac{m}{M}\right)^{2\Lambda^4 \ln \Lambda}\left(1-\frac{M}{m}\right). \tag{40}$$

Furthermore, the transmission rate of the RS scheme given by (35) is

$$\begin{aligned} R^{\mathrm{RS}} &= n^{\delta} + o(n^{\delta}) \\ &= \left(2\frac{m}{M}\right)^{4\Lambda^4 \ln 10.5} + o\left(\left(2\frac{m}{M}\right)^{4\Lambda^4 \ln 10.5}\right) \end{aligned} \tag{41}$$

Then we obtain[7]

$$\begin{aligned} \frac{R^{\mathrm{RS}}}{R^{\mathrm{u}}} &= \frac{m-M}{m} \cdot \left(2\frac{m}{M}\right)^{2\Lambda^4(2\ln 10.5 - \ln \Lambda)} \\ &\quad + o\left(\frac{m-M}{m} \cdot \left(2\frac{m}{M}\right)^{2\Lambda^4(2\ln 10.5-\ln\Lambda)}\right). \end{aligned} \tag{42}$$

This result demonstrates that for a fixed m/M and $\Lambda > 10.5^2$, by using $\tau = \Lambda^{\frac{2\Lambda^4 \ln\left(2\frac{m}{M}\right)}{\ln \Lambda}\left(1+\frac{2\ln 10.5}{\ln\Lambda}+o(1)\right)}$ (see Appendix A), we have[8]

$$\lim_{n\to\infty} \frac{R^{\mathrm{RS}}}{R^{\mathrm{u}}} = \lim_{\Lambda\to\infty} \frac{R^{\mathrm{RS}}}{R^{\mathrm{u}}} = 0, \tag{43}$$

which shows a significant gain in terms of transmission rate of the RS graph approach compared to uncoded unicasting scheme.

Remark 4: It can be seen that the idea of designing the decentralized coded caching approach discussed in Section IV can also be extended to the RS graph coded caching methods. The major difference is that the packet sets, each of which will be randomly and uniformly cached by users, are constructed based on the RS graph. The delivery procedure is similar to that in Section IV.

VI. Achievability With Spatial Reuse

In this section, we study the hypercube and RS graph D2D coded caching schemes when spatial reuse is allowed. We use *clustering* to divide up a caching network into many smaller caching networks. We reduce the transmission range

[7]Notice that, (42) is only dependent on M, m and Λ and not dependent on z. This occurs because z is a function of M, m and Λ and for these derivations z is essentially substituted with $z = 2\Lambda^4 \ln\left(2\frac{m}{M}\right)/\ln\Lambda$.

[8]Note that the RS graph based approach cannot perform worse than the uncoded unicasting by the construction. The reason we need the condition that $\Lambda > 10.5^2$ is due to the fact that the τ obtained from [28], [29] and used in (35) is a sufficient condition and may not be necessary.

to be $r < \sqrt{2}$ in order to have localized D2D communication such that multiple clusters can communicate without interference. In our analysis, we demonstrate that clustering with the newly proposed D2D designs has the advantage of significant reduction in packetization. Furthermore, we find a throughput-packetization trade-off when using clustering based on the reuse factor $\mathcal{K}$ and link rates, C_r, for spatial reuse, and $C_{\sqrt{2}}$, with no spatial reuse. Surprisingly, for the RS graph design, in some cases, we find that there is no trade-off and throughput and packetization are strictly better for reasonable choices of parameters $\mathcal{K}$, C_r, and $C_{\sqrt{2}}$ and when n is arbitrarily large. Furthermore, the transmission rate of the hypercube and RS graph D2D schemes are dependent on the number of users in each cluster, different from the scheme of [14] which is independent. Therefore, in contrast to the spatial reuse analysis of [14] where there is potentially no order gain in throughput as $n \to \infty$ by using spatial reuse, in the newly proposed schemes, spatial reuse achieves a higher throughput gain for some parameters.

Different from to the regime where $r \geq \sqrt{2}$, in this case, we also need to design a non-trivial transmission policy to schedule concurrent active D2D transmissions. Similar to the scheduling schemes in [14], the proposed policy is based on dividing the network into clusters of equal size g_c users. Users are allowed to receive messages only from users in the same cluster.[9] Therefore, each cluster is treated as a smaller network compared to the entire network. Assuming that $g_c M \geq m$,[10] the total cache size of each cluster is sufficient to store the entire file library. Under this assumption, the cache placement and delivery schemes introduced in Sections III-A and V-C can be applied to each cluster. Hence, it can been seen that the transmission rate of each cluster is given by either

$$R_c^{\mathrm{hc}}(M) = \frac{t_{\mathrm{c}}}{t_{\mathrm{c}} - 1} \frac{m}{M} \left(1 - \frac{M}{m}\right), \tag{44}$$

for the hypercube scheme where $t_{\mathrm{c}} \triangleq g_{\mathrm{c}} M/m$, or

$$R_{\mathrm{c}}^{\mathrm{RS}}(M) = \frac{\tau}{g_{\mathrm{c}}} \frac{2\gamma}{2\gamma - 1} \overset{g_{\mathrm{c}} \to \infty}{=} g_{\mathrm{c}}^{\delta} + o(g_{\mathrm{c}}^{\delta}), \tag{45}$$

for the RS graph scheme where $t_{\mathrm{c}} \triangleq g_{\mathrm{c}} M/m$, $g_{\mathrm{c}} = \Lambda^z$ and τ, γ and δ are given in Theorem 3. As mentioned in Remark 1, a straightforward achievable transmission policy consists of grouping the set of clusters into $\mathcal{K}$ spatial reuse sets such that the clusters of the same reuse set do not interfere and can be activated simultaneously.[11] In each active cluster, a single transmitter is active per time-slot and it is received by all users in the cluster, as in classical time-frequency reuse schemes with reuse factor $\mathcal{K}$ that are currently used in cellular networks [41, Ch. 17]. An example of a reuse set is shown in Fig. 1. In particular, we can pick $\mathcal{K} = \left(\left\lceil \sqrt{2}(1+\Delta) \right\rceil + 1\right)^2$ [37].

[9]Note that this condition can be relaxed by using the similar communication scheme base on ITLinQ [40].

[10]If the condition $g_c M \geq m$ is not satisfied, we can choose a larger transmission range such that this condition is feasible.

[11]Note that the interference management in this paper is based on protocol model. However, similar scheme can be designed for AWGN or fading channel model based on the condition that treating interference as noise is optimal (ITLinQ [40]).

A. Clustering With the Hypercube Caching Approach

The achievable throughput of the hypercube approach with clustering is given by the following theorem.

Theorem 4: Let m, g_{c}, M be the library size, number of users in each cluster and the cache size per user, respectively. For $r \leq \sqrt{2}$, $t_{\mathrm{c}} = \frac{g_{\mathrm{c}} M}{m} \geq 2$, and $\frac{m}{M}, t_{\mathrm{c}} \in \mathbb{Z}^+$, the following per user throughput $T_{\mathrm{c}}^{\mathrm{hc}}(M)$ is achievable:

$$T_{\mathrm{c}}^{\mathrm{hc}}(M) = \frac{C_r}{\mathcal{K}} \frac{M}{m - M} \frac{t_{\mathrm{c}} - 1}{t_{\mathrm{c}}} \tag{46}$$

with the requirement of at most $K_{\mathrm{c}}^{\mathrm{hc}} = (t_{\mathrm{c}} - 1)\left(\frac{m}{M}\right)^{t_{\mathrm{c}}}$ packetizations. □

Proof: By using (2), we can obtain

$$T_{\mathrm{c}}^{\mathrm{hc}}(M) = \frac{C_r}{\mathcal{K}} \frac{1}{R_{\mathrm{c}}^{\mathrm{hc}}(M)} = \frac{C_r}{\mathcal{K}} \frac{M}{m - M} \frac{t_{\mathrm{c}} - 1}{t_{\mathrm{c}}}, \tag{47}$$

where $R_{\mathrm{c}}^{\mathrm{hc}}(M)$ is given by (44). Noting that the packetization of the hypercube scheme without clustering is $(t-1)\left(\frac{m}{M}\right)^t$ in the worst case and replacing t by t_{c}, we get $K_{\mathrm{c}}^{\mathrm{hc}} = (t_{\mathrm{c}} - 1)\left(\frac{m}{M}\right)^{t_{\mathrm{c}}}$. ■

The multiplicative gap of $T_{\mathrm{g}}^{\mathrm{hc}}$ and $T_{\mathrm{c}}^{\mathrm{hc}}$ throughput is given by

$$G_T^{\mathrm{hc}} = \frac{T_{\mathrm{c}}^{\mathrm{hc}}}{T_{\mathrm{g}}^{\mathrm{hc}}} = \frac{C_r}{C_{\sqrt{2}}\,\mathcal{K}} \frac{1}{G_R^{\mathrm{hc}}} = \frac{C_r}{C_{\sqrt{2}}\,\mathcal{K}} \frac{n}{g_{\mathrm{c}}} \frac{t_{\mathrm{c}} - 1}{t - 1} \tag{48}$$

Remark 5: The throughput has increased by using the clustering scheme compared to the scheme for $r \geq \sqrt{2}$ if $G_T^{\mathrm{hc}} > 1$ or in other words

$$\frac{C_r}{C_{\sqrt{2}}\,\mathcal{K}} > \frac{g_{\mathrm{c}}}{n} \frac{t - 1}{t_{\mathrm{c}} - 1}. \tag{49}$$

This relationship can be further simplified if $g_{\mathrm{c}} M \gg m$ and clustering will improve the throughput if $C_r/\mathcal{K} > C_{\sqrt{2}}$. This provides a similar result to that was shown in [14]. The exact relationship between link rate, C_r, and transmission range, r, depends on the physical channel model and wireless network design. However, C_r may not decrease as r decreases and clustering has the potential to increase throughput.

Remark 6: Clustering reduces the number of file partitions as also shown in [14]. For a cluster with g_{c} users, the ratio of the number of packetizations with and without clustering

$$\frac{K^{\mathrm{hc}}}{K_{\mathrm{c}}^{\mathrm{hc}}} = \left(\frac{t - 1}{t_{\mathrm{c}} - 1}\right) \left(\frac{m}{M}\right)^{\frac{M}{m}(n - g_{\mathrm{c}})}. \tag{50}$$

As the cluster size g_{c} becomes smaller, an exponential decrease in the number of file partitions is observed. Therefore, even when clustering does not increase throughput, spatial reuse still introduces a throughput-packetization trade-off.

B. Clustering With the RS Graph Caching Approach

In this section we first discuss the impact of clustering on packetization and throughput with the RS graph coded caching approach. We then consider two ways of reducing g_{c} based on the construction in [28], [29] and study their specific impact on throughput and cache size requirement. The RS graph scheme achieves the following throughput:

Theorem 5: Let m, g_{c}, M be the library size, number of users in each cluster and the cache size per user, respectively.

For $r < \sqrt{2}$, $g_{\rm c} = \Lambda^z$, where Λ is any positive integer such that $z \geq 2\Lambda$, and $M = 2mg_{\rm c}^{-\frac{1}{2\Lambda^4 \ln \Lambda}}$, let $t_{\rm c} = \frac{g_{\rm c} M}{m} \in \mathbb{Z}^+$, the per user throughput is given by:

$$T_{\rm c}^{\rm RS}(M) = \frac{C_r}{\mathcal{K}} \frac{g_{\rm c}}{\tau} \frac{2\gamma - 1}{2\gamma} \tag{51}$$

with the requirement of $K_{\rm c}^{\rm RS} = (2\gamma - 1) g_{\rm c}$ packetizations.□

We have the following corollary for the asymptotic regime.

Corollary 3: For $r < \sqrt{2}$, when $g_{\rm c} \to \infty$, $K_{\rm c}^{\rm RS} = O\left(g_c^{2-\delta}\right)$, $t_c \geq 2$, and $M = 2mg_c^{-c_1 \delta \exp(-c_2/8)}$, where c_1, c_2 are some positive constants, let $t_{\rm c} = \frac{g_{\rm c} M}{m} \in \mathbb{Z}^+$, the throughput is given by:

$$T_{\rm c}^{\rm RS}(M) = \frac{C_r}{\mathcal{K}} g_{\rm c}^{-\delta} + o\left(\frac{C_r}{\mathcal{K}} g_{\rm c}^{-\delta}\right), \tag{52}$$

where $\delta = \frac{2 \ln 10.5}{\ln \Lambda}$. □

Since the proofs of Theorem 5 and Corollary 3 are similar to that of Theorem 4 by using Theorem 3 and Corollary 1, the proofs are omitted due to space limitation. From Theorems 3 and 5, we find the multiplicative reduction in packetization from clustering with the RS scheme is

$$\frac{K^{\rm RS}}{K_{\rm c}^{\rm RS}} = O\left(\frac{n^2}{g_{\rm c}^2}\right). \tag{53}$$

To understand the impact of clustering on rate and throughput, we need to find a cluster size such that $g_{\rm c} = (\Lambda')^{z'}$. Let the number of users in the caching network be $n = \Lambda^z$. In the following, we explore two methods to determine $g_{\rm c}$ which are letting $\Lambda' = \Lambda$, $z' < z$ or $\Lambda' < \Lambda$, $z' = z$.

1) $\Lambda' = \Lambda$, $z' < z$: Let $n = \Lambda^z$ and $g_{\rm c} = \Lambda^{z'}$, where $z' < z$. From (37) and (45), the multiplicative gap between the transmission rate of $R_{\rm c}^{\rm RS}(M)$ and $R^{\rm RS}(M)$ is given by

$$\begin{aligned} G_R^{\rm RS} &= \frac{R_{\rm c}^{\rm RS}(M)}{R^{\rm RS}(M)} = \frac{g_{\rm c}^{\delta} + o(g_{\rm c}^{\delta})}{n^{\delta} + o(n^{\delta})} \\ &= \Lambda^{\delta(z'-z)} + o\left(\Lambda^{\delta(z'-z)}\right) \\ &= 10.5^{2(z'-z)} + o\left(10.5^{2(z'-z)}\right). \end{aligned} \tag{54}$$

From (54), we can observe that by clustering the D2D network, the transmission rate (traffic load) can be reduced significantly. For example, if $z - z' = 2$, e.g., the network is partitioned into Λ^2 clusters, $G_R^{\rm RS} = 10.5^{-4} < 10^{-4}$. Nevertheless, due to the spatial reuse gain, by using Corollary 2 and 3, the multiplicative gap between $T_{\rm c}^{\rm RS}(M)$ and $T^{\rm RS}(M)$ is given by

$$\begin{aligned} G_T^{\rm RS} &= \frac{T_{\rm c}^{\rm RS}(M)}{T^{\rm RS}(M)} = \frac{\frac{C_r}{\mathcal{K}} g_{\rm c}^{-\delta} + o\left(\frac{C_r}{\mathcal{K}} g_{\rm c}^{-\delta}\right)}{C_{\sqrt{2}}\, n^{-\delta} + o\left(C_{\sqrt{2}}\, n^{-\delta}\right)} \\ &\overset{(a)}{=} \frac{C_r}{C_{\sqrt{2}}} \frac{1}{\mathcal{K}} 10.5^{2(z-z')} + o\left(10.5^{2(z-z')}\right), \end{aligned} \tag{55}$$

where (a) is obtained by repeating the same procedure as (54). Surprisingly, clustering appears to increase throughput even when the benefit of reduced transmission range and spatial reuse is limited (i.e. $\frac{C_r}{C_{\sqrt{2}}\mathcal{K}}$ is small). For example, if $z - z' = 2$, then from (55),

$$G_T^{\rm RS} = \frac{C_r}{C_{\sqrt{2}}} \frac{1}{\mathcal{K}} 10.5^4. \tag{56}$$

As clustering is performed when $\Lambda' = \Lambda$ and $z' < z$, the required cache capacity M increases. For this clustering scheme, from Theorems 3 and 5, the cache capacity $M(g_{\rm c})$ satisfies

$$\begin{aligned} M(g_{\rm c}) &= 2mg_{\rm c}^{-\frac{1}{2\Lambda^4 \ln \Lambda}} = 2mn^{-\frac{1}{2\Lambda^4 \ln \Lambda}} \frac{g_{\rm c}^{-\frac{1}{2\Lambda^4 \ln \Lambda}}}{n^{-\frac{1}{2\Lambda^4 \ln \Lambda}}} \\ &= M(n) \exp\left(\frac{z - z'}{2\Lambda^4}\right). \end{aligned} \tag{57}$$

Note that by using (57), to achieve the promised throughput, it needs $M(g_{\rm c}) \geq M(n) \exp\left(\frac{z-z'}{2\Lambda^4}\right)$. For example, if $z - z' = 2$, it requires $M(n/\Lambda^2) = M(n) e^{\frac{1}{\Lambda^4}}$.

Remark 7: For a realizable RS design where $z = \omega(\Lambda^4)$, $\frac{M(n)}{m} = o(1)$, the effects of local caching gain and increased cache size are insignificant.[12] In other words, $\frac{1 - M(n)/m}{1 - M(n) e^{\frac{1}{\Lambda^4}}/m} \approx 1$. The factor of 10.5^4 in the gain of throughput of (56) is a direct consequence of the decrease in rate from clustering in (54). The reduction in the rate is a consequence of the reduced cluster size and increased multicasting gain.

2) $\Lambda' < \Lambda$, $z' = z$: Let $n = \Lambda^z$ and $g_{\rm c} = (\Lambda')^z$, where $\Lambda' < \Lambda$. From Corollary 1 and (44), the multiplicative gap between the transmission rate of $R_{\rm c}^{\rm RS}(M)$ and $R^{\rm RS}(M)$ is given by

$$\begin{aligned} G_R^{\rm RS} &= \frac{R_{\rm c}^{\rm RS}(M)}{R^{\rm RS}(M)} = \frac{g_c^{\delta'} + o(g_c^{\delta'})}{n^{\delta} + o(n^{\delta})} \\ &= \frac{(\Lambda')^{\frac{2 \ln 10.5}{\ln \Lambda'} z}}{\Lambda^{\frac{2 \ln 10.5}{\ln \Lambda} z}} + o\left(\frac{(\Lambda')^{\frac{2 \ln 10.5}{\ln \Lambda'} z}}{\Lambda^{\frac{2 \ln 10.5}{\ln \Lambda} z}}\right) = 1 + o(1). \end{aligned} \tag{58}$$

From (58), we see that by clustering the D2D network, the transmission rate is almost unchanged. It follows from Corollary 2 and 3 that the multiplicative gap between $T_{\rm c}^{\rm RS}(M)$ and $T^{\rm RS}(M)$ is given by

$$\begin{aligned} G_T^{\rm RS} &= \frac{T_{\rm c}^{\rm RS}(M)}{T^{\rm RS}(M)} = \frac{\frac{C_r}{\mathcal{K}} g_{\rm c}^{-\delta'} + o\left(\frac{C_r}{\mathcal{K}} g_{\rm c}^{-\delta'}\right)}{C_{\sqrt{2}}\, n^{-\delta} + o\left(C_{\sqrt{2}}\, n^{-\delta}\right)} \\ &= \frac{C_r}{C_{\sqrt{2}}} \frac{1}{\mathcal{K}} + o\left(\frac{C_r}{C_{\sqrt{2}}} \frac{1}{\mathcal{K}}\right). \end{aligned} \tag{59}$$

From (59), we can see that similar to [14], there is no fundamental cumulative gain by using both spatial reuse and coded multicasting. Therefore, depending on C_r, $C_{\sqrt{2}}$ and $\mathcal{K}$, clustering either improves throughput or introduces a throughput-packetization trade-off. Note that, these parameters are not captured by the protocol model and may depend on the operating frequency and appropriate channel model of the underlying wireless network physical layer [16].

VII. Conclusion

In this work, we study D2D coded caching network designs which require with less packetization compared to the state-of-the-art coded D2D caching schemes in [14]. We approach this problem by proposing new combinatorial caching designs

[12] From Appendix A, $M(n) = 2mn^{-\frac{1}{2\Lambda^4 \ln \Lambda}} = 2m \exp\left(-\frac{z}{2\Lambda^4}\right)$ where $n = \Lambda^z$. Thus if $z = \omega(\Lambda^4)$, then $\frac{M(n)}{m} = o(1)$.

and exploiting random caching and spatial reuse to introduce throughput-packetization trade-offs. We propose two new network design approaches, which are hypercube approach and Ruzsa-Szeméredi (RS) graph approach. In particular, the hypercube approach is unique compared to other D2D designs because, in some cases, it does not require extra packetization beyond the placement phase. The hypercube approach requires exponentially less packetization as compared to coded caching scheme in [14] and yields nearly the same per user throughput for a large number of users. The RS Graph approach was expanded from a previously studied shared-link network design [29] and modified for D2D networks. It requires only subquadratic packetization in terms of the number of users and achieves a near constant per user throughput. In addition, we demonstrate how the hypercube scheme can be expanded to design decentralized networks and discovered a clear trade-off between transmission rate and packetization. Finally, we explore our new schemes with network clustering to utilize spatial reuse and increase per user throughput and also decrease the packetization, a property that is unique to D2D caching networks.

APPENDIX A
PROOF OF THEOREM 3

To prove Theorem 3, we first introduce the following construction of RS Graph from [28], [29].

Definition 11: **(Graph Construction)** A graph, $\mathcal{G}(\mathcal{V},\mathcal{E})$, is defined such that $\mathcal{V} = [\Lambda]^z$ where $\Lambda \in \mathbb{Z}^+$, z is even, and $z \geq 2\Lambda$. Let $\mu = \mathbb{E}_{x,y}[||x-y||_2^2]$, where x and y are sampled uniformly from $\mathcal{V}$. For a pair of vertices, $u, v \in \mathcal{V}$, $(u,v) \in \mathcal{E}$ if and only if $|\ ||u-v||_2^2 - \mu| \leq z$. $\Diamond$

Given this construction, the graph consists of $\tau = n^f$ edge disjoint induced matchings and misses at most n^k edges such that $f = 1 + 2\frac{\ln 10.5}{\ln \Lambda} + o(1)$, and $k = 2 - \frac{1}{2\Lambda^4 \ln \Lambda} + o(1)$. Furthermore, given a vertex, $x \in \mathcal{V}$, the degree of x, defined as d, is bounded by

$$d = |y \in \mathcal{V} : (x,y) \in \mathcal{E}| \geq n\Big(1 - 2n^{-\frac{1}{2\Lambda^4 \ln \Lambda}}\Big). \quad (60)$$

The degree of any vertex defines the number of packets that a user does not cache. Therefore, if $\frac{M}{m} \geq 1 - \frac{d_{\text{lb}}}{n} = 2n^{-\frac{1}{2\Lambda^4 \ln \Lambda}}$, where d_{lb} is the lower bound of d, then the graph construction can be used as a cache placement scheme. Hence, we can pick $M = 2mn^{-\frac{1}{2\Lambda^4 \ln \Lambda}}$. Since $d \geq d_{\text{lb}} = n\left(1 - \frac{M}{m}\right) = n - t$, we have $t \in \mathbb{Z}^+$. By the construction of the proposed scheme described in Section V-C, each file is partitioned into $n(2\gamma - 1)$ packets, and each contains F/K bits. Moreover, for each disjoint matching in the constructed RS graph, there are 2γ transmissions, and each has size $\frac{F}{n(2\gamma-1)}$ bits. Since the number of disjoint matching is τ, there are $2\gamma \cdot \tau$ transmissions.

$$R(M) = 2\gamma \cdot \tau \cdot \frac{F}{n(2\gamma-1)} \cdot \frac{1}{F} = \frac{\tau}{n}\frac{2\gamma}{2\gamma-1}. \quad (61)$$

REFERENCES

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

[2] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[3] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 16–22, Aug. 2016.

[4] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, Sep. 2016.

[5] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite length analysis of caching-aided coded multicasting," in *Proc. 52nd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2014, pp. 914–920.

[6] 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.

[7] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Sep. 2016, pp. 161–165.

[8] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," 2017, *arXiv:1702.04563*. [Online]. Available: http://arxiv.org/abs/1702.04563

[9] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3212–3229, Jun. 2016.

[10] A. Sengupta and R. Tandon, "Improved approximation of storage-rate tradeoff for caching with multiple demands," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 1940–1955, May 2017.

[11] S.-W. Jeon, S.-N. Hong, M. Ji, G. Caire, and A. F. Molisch, "Wireless multihop device-to-device caching networks," *IEEE Trans. Inf. Theory*, vol. 63, no. 3, pp. 1662–1676, Mar. 2017.

[12] A. Liu, V. Lau, and G. Caire, "Cache-induced hierarchical cooperation in wireless device-to-device caching networks," 2016, *arXiv:1612.07417*. [Online]. Available: http://arxiv.org/abs/1612.07417

[13] S. P. Shariatpanahi, G. Caire, and B. H. Khalaj, "Physical-layer schemes for wireless coded caching," 2017, *arXiv:1711.05969*. [Online]. Available: http://arxiv.org/abs/1711.05969

[14] 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.

[15] M. Ji, G. Caire, and A. F. Molisch, "The throughput-outage tradeoff of wireless one-hop caching networks," *IEEE Trans. Inf. Theory*, vol. 61, no. 12, pp. 6833–6859, Dec. 2015.

[16] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 176–189, Jan. 2016.

[17] M. Ji, R.-R. Chen, G. Caire, and A. F. Molisch, "Fundamental limits of distributed caching in multihop D2D wireless networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2950–2954.

[18] K. Wan, D. Tuninetti, M. Ji, and G. Caire, "Novel inter-file coded placement and D2D delivery for a cache-aided fog-ran architecture," 2018, *arXiv:1811.05498*. [Online]. Available: https://arxiv.org/abs/1811.05498

[19] A. A. Zewail and A. Yener, "Device-to-device secure coded caching," 2018, *arXiv:1809.06844*. [Online]. Available: http://arxiv.org/abs/1809.06844

[20] Ç. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of D2D coded caching with uncoded cache placement and one-shot delivery," 2019, *arXiv:1901.05921*. [Online]. Available: http://arxiv.org/abs/1901.05921

[21] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Device-to-device coded caching with distinct cache sizes," 2019, *arXiv:1903.08142*. [Online]. Available: http://arxiv.org/abs/1903.08142

[22] M.-C. Lee, M. Ji, A. F. Molisch, and N. Sastry, "Throughput–outage analysis and evaluation of cache-aided D2D networks with measured popularity distributions," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5316–5332, Nov. 2019.

[23] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "Device-to-device private caching with trusted server," 2019, *arXiv:1909.12748*. [Online]. Available: http://arxiv.org/abs/1909.12748

[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] H. H. S. Chittoor, M. Bhavana, and P. Krishnan, "Coded caching via projective geometry: A new low subpacketization scheme," 2019, *arXiv:1901.07823*. [Online]. Available: http://arxiv.org/abs/1901.07823
[27] I. Z. Ruzsa and E. Szemerédi, "Triple systems with no six points carrying three triangles," in *Proc. 5th Hungarian Colloq.*, Keszthely, Hungary, 1976, pp. 939–945.
[28] N. Alon, A. Moitra, and B. Sudakov, "Nearly complete graphs decomposable into large induced matchings and their applications," in *Proc. 44th Symp. Theory Comput. (STOC)*, 2012, pp. 1079–1090.
[29] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using Ruzsa-Szeméredi graphs," 2017, *arXiv:1701.07115*. [Online]. Available: http://arxiv.org/abs/1701.07115
[30] J. Wang, M. Cheng, Q. Yan, and X. Tang, "On the placement delivery array design for coded caching scheme in D2D networks," 2017, *arXiv:1712.06212*. [Online]. Available: http://arxiv.org/abs/1712.06212
[31] J. Wang, M. Cheng, Q. Yan, and X. Tang, "Placement delivery array design for coded caching scheme in D2D networks," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3388–3395, May 2019.
[32] S. Jin, Y. Cui, H. Liu, and G. Caire, "New order-optimal decentralized coded caching schemes with good performance in the finite file size regime," 2016, *arXiv:1604.07648*. [Online]. Available: http://arxiv.org/abs/1604.07648
[33] N. Woolsey, R.-R. Chen, and M. Ji, "Device-to-device caching networks with subquadratic subpacketizations," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.
[34] N. Woolsey, R.-R. Chen, and M. Ji, "Coded caching in wireless device-to-device networks using a hypercube approach," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[35] N. Woolsey, R.-R. Chen, and M. Ji, "Cascaded coded distributed computing on heterogeneous networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 2644–2648.
[36] N. Woolsey, R.-R. Chen, and M. Ji, "Coded distributed computing with heterogeneous function assignments," 2019, *arXiv:1902.10738*. [Online]. Available: http://arxiv.org/abs/1902.10738
[37] F. Xue and P. Kumar, *Scaling Laws for Ad Hoc Wireless Networks: An Information Theoretic Approach*. Beirut, Lebanon: Now, 2006.
[38] K. Konstantinidis and A. Ramamoorthy, "Leveraging coding techniques for speeding up distributed computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[39] M. Raab and A. Steger, "'Balls into bins'—A simple and tight analysis," in *Randomization and Approximation Techniques in Computer Science*. Berlin, Germany: Springer, 1998, pp. 159–170.
[40] N. Naderializadeh, D. T. H. Kao, and A. S. Avestimehr, "How to utilize caching to improve spectral efficiency in device-to-device wireless networks," in *Proc. 52nd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2014, pp. 415–422.
[41] A. F. Molisch, *Wireless Communications*, 2nd ed. Hoboken, NJ, USA: Wiley, 2011.

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 at College Park 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. From 2014 to 2017, he was an Electrical Engineer with Northrop Grumman Corporation (NGC), Ogden, UT, developing test and evaluation methods, modernization solutions and signal processing algorithms for the sustainment of aging aircraft, and ground communication systems. His research interests include combinatoric designs and algorithms for resource allocation, coding and efficient communications in distributed computing, and private and caching networks. 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.

Rong-Rong Chen (Member, IEEE) received the B.S. degree in applied mathematics from Tsinghua University, China, in 1993, and the M.S. degree in mathematics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign in 1995 and 2003, respectively. She was an Assistant Professor with The University of Utah, from 2003 to 2011, where she has been an Associate Professor since 2011. Her main research interests include communication systems and networks, with current emphasis on distributed computing, machine learning, caching networks, statistical signal processing, image reconstructions, and channel coding. She was a recipient of the M. E. Van Valkenburg Graduate Research Award for excellence in doctoral research in the ECE Department, University of Illinois at Urbana-Champaign, in 2003. She was a recipient of the prestigious National Science Foundation Faculty Early Career Development (CAREER) Award in 2006. She was rated among the Top 15% Instructors of College of Engineering, The University of Utah, in 2017 and 2018. She has served on the technical program committees of leading international conferences in wireless communication and networks. She has served as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING and a Guest Editor for IEEE JOURNAL ON SELECTED TOPICS IN SIGNAL PROCESSING.

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. degrees in electrical engineering from the Royal Institute of Technology, Sweden, and the University of California at Santa Cruz, in 2008 and 2010, respectively, 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 Ltd.) 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 information theory, coding theory, concentration of measure and statistics with the applications of caching networks, wireless communications, distributed computing and storage, security and privacy, and (statistical) signal processing. He received the IEEE Communications Society Leonard G. Abraham Prize for the Best IEEE 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 the USC Annenberg Fellowship from 2010 to 2014.