Hierarchical Coded Caching

Nikhil Karamchandani* Urs Niesen† Mohammad Ali Maddah-Ali† Suhas Diggavi*

*Dept. of Electrical Engineering, University of California, Los Angeles, California, USA
†Bell Labs, Alcatel-Lucent, New Jersey, USA

Email: nikhil@ee.ucla.edu, urs.niesen@alcatel-lucent.com, mohammdaali.maddah-ali@alcatel-lucent.com, suhas@ee.ucla.edu

***Abstract*—It has recently been demonstrated that for single-layer cache networks, jointly designing caching and delivery can enable significant benefits over conventional caching. This was based on strategically designing the cached content to induce coded multicasting opportunities even among users with different demands and without foreknowledge of the user demands.**

In this work, we extend this coded caching approach to a multi-hop hierarchical content delivery network with two layers of caches. We propose a new caching scheme that combines two basic approaches. The first approach provides coded multicasting opportunities within each layer (through decoding and forwarding); the second approach provides coded multicasting opportunities across multiple layers (through strategic forwarding without decoding). By striking the right balance between these two approaches, we show that the proposed scheme achieves the optimal communication rates to within a constant multiplicative and additive gap. We further show that there is no tension between the rates in each of the two layers up to the aforementioned gap. Thus, both layers can simultaneously operate at approximately the minimum rate.

I. Introduction

Media content demand, driven by video, is dominating network data traffic. To enable its scalable delivery, content-distribution networks (CDNs) that cache (store) content closer to its demand have been widely deployed, see for example [1]–[6] and references therein. A common feature among the caching schemes studied in the literature is that those parts of a requested file that are available at nearby caches are served locally, whereas the remaining file parts are served via orthogonal transmissions from an origin server hosting all the files. Recently, [7], [8] proposed a new caching approach, called *coded caching*, that exploits cache memories not only to deliver part of the content locally, but also to create coded multicasting opportunities among users with different demands. It is shown there that the reduction in rate due to these coded multicasting opportunities is significant and can be on the order of the number of users in the network. The setting considered in [7], [8] consists of a single layer of caches between the origin server and the end users. The server communicates directly with all the caches via a shared link, and the objective is to minimize the required transmission rate by the server. For this basic network scenario, coded caching is shown there to be optimal within a constant factor. These results have been extended to nonuniform demands in [9] and to online caching systems in [10].

In practice, many caching systems consist of not only one but multiple layers of caches, usually arranged in a tree-like hierarchy with the origin server at the root node and the users connected to the leaf caches [2], [11], [12]. Each parent cache communicates with its children caches in the next layer, and the objective is to minimize the transmission rates in the various layers.

There are several key questions when analyzing such hierarchical caching systems. A first question is to characterize the optimal tradeoff between the cache memory sizes and the rates of the links connecting the layers. One particular point of interest is if there is any tension between the rates in the different layers in the network. In other words, if we reduce the rate in one layer, does this necessarily increase the rate in other layers? If there is no such tension, then both layers can simultaneously operate at minimum rate. A second question is how to extend the coded caching approach to this setting. Can we simply apply the single-layer scheme from [7], [8] in each layer separately or do we need to apply coding across several layers in order to minimize transmission rates?

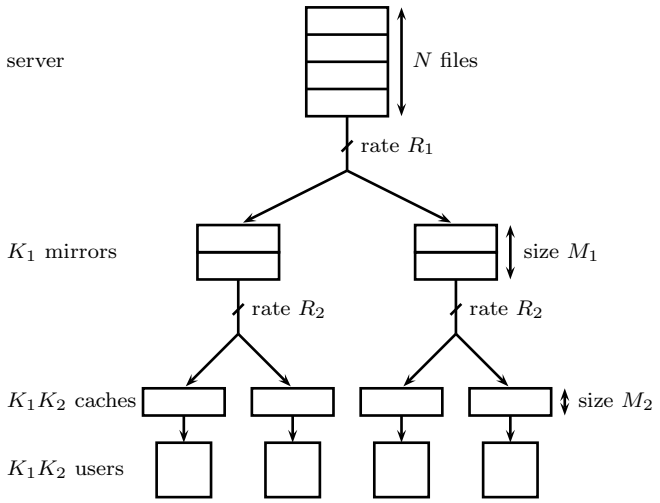

Fig. 1. System setup for the hierarchical caching problem: a server hosting N files is connected to K_1 mirrors, each able to store M_1 of the files. Every mirror, in turn, is connected to K_2 caches each able to store M_2 of the files. A single user is attached to each of these caches. Mirrors and caches are filled during the low network-traffic period, when user requests have not been revealed yet. Later during the high network-traffic period, each user requests one of the N files. The aim is to minimize the rate R_1 from the server to the mirrors and the rate R_2 from the mirrors to the caches. In the figure, $N = 4$, $K_1 = K_2 = 2$, $M_1 = 2$, and $M_2 = 1$.

In this work, we focus on a hierarchical caching system with two layers of caches as depicted in Fig. 1. For simplicity, we will refer to the first layer of caches as mirrors. We propose a new caching scheme exploiting two types of coded caching

978-1-4799-5186-4/14/$31.00 ©2014 IEEE

opportunities: the first type involves only a single layer at a time, i.e., it operates between a node and its direct children. These single-layer coding opportunities are available over the link connecting the origin server to the mirrors and also over the link connecting each mirror to the user caches. The second type involves two layers at a time. These two-layer opportunities are available between the origin server and the user caches. We show that, by striking the right balance between these two types of coded caching opportunities, the proposed caching scheme attains the approximately optimal memory-rate tradeoff to within a constant additive and multiplicative gap. Investigating the achievable rates also shows that there is no tension between the rates over the first and second layers up to the same aforementioned gap. Thus, both layers can simultaneously operate at approximately minimum rate.

The remainder of the paper is organized as follows. We describe the problem setting in Section II and provide some preliminaries in Section III. Section IV presents our main results and discusses their engineering implications. Section V introduces the proposed caching scheme and characterizes its performance. For brevity, we will skip the proofs of bounded gap here and refer the reader to [13] for the details.

II. Problem Setting

We consider a hierarchical content delivery network as illustrated in Fig. 1 in Section I. The system consists of a single origin server hosting a collection of N files, each of size F bits. The server is connected through an error-free shared link to K_1 mirror sites, each with a memory of size M_1F bits. Each mirror, in turn, is connected through an error-free shared link to K_2 users. Thus, the system has a total of K_1K_2 users. Each user has an associated cache memory of size M_2F bits. The quantities M_1 and M_2 are the normalized memory sizes of the mirrors and user caches, respectively. We refer to the jth user attached to mirror i as "user (i,j)" and the corresponding cache as "cache (i,j)". Throughout, we will focus on the most relevant case where the number of files N is larger than the total number of users K_1K_2 in the system[1], i.e., $N \geq K_1K_2$.

The content delivery system operates in two phases: a *placement phase* followed by a *delivery phase*. The placement phase occurs during a period of low network traffic. In this phase, all the mirrors and user caches store content related to the N files (possibly using randomized strategies), while satisfying the corresponding memory constraints. Crucially, this is done without any prior knowledge of future user requests. The delivery phase occurs during a period of high network traffic. In this phase, each user requests one of the N files from the server. Formally, the user requests can be represented as a matrix $\boldsymbol{D}$ with entry $d_{i,j} \in \{1,2,\ldots,N\}$ denoting the request of user (i,j). The user requests are forwarded to the corresponding mirrors and further on to the server. Based on the requests and the stored contents of the mirrors and the user caches during the placement phase, the server transmits a message $X^{\boldsymbol{D}}$ of size at most R_1F bits over the shared link to the mirrors. Each mirror i receives the server message and, using its own memory content, transmits a message $Y_i^{\boldsymbol{D}}$ of size at most R_2F bits over its shared link to users $(i,1),(i,2),\ldots,(i,K_2)$. Using only the contents of its cache (i,j) and the received message $Y_i^{\boldsymbol{D}}$ from mirror i, each user (i,j) attempts to reconstruct its requested file $d_{i,j}$.

We say that the tuple (M_1,M_2,R_1,R_2) is *feasible* if for *every* request matrix $\boldsymbol{D}$, each user (i,j) is able to recover its requested file $d_{i,j}$. The object of interest in this paper is the feasible rate region:

Definition. For memory sizes $M_1,M_2 \geq 0$, the *feasible rate region* is defined as

$$\mathcal{R}^\star(M_1,M_2) \triangleq \text{closure}\big\{(R_1,R_2) : (M_1,M_2,R_1,R_2) \text{ is feasible}\big\}. \quad (1)$$

III. Preliminaries

Consider a special case of the hierarchical caching setting with no cache memory at the users and only a single user accessing each mirror, i.e., $M_2 = 0$ and $K_2 = 1$. Let the normalized mirror memory size be $M_1 = M$ and the number of mirrors $K_1 = K$. This results in a system with only a single layer of caches (namely the mirrors).

Note that for this single-layer scenario, each mirror needs to recover the file requested by its corresponding user and then forward the entire file to it. Thus, a transmission rate of $R_2 = K_2 = 1$ over the link from the mirror to the user is both necessary and sufficient in this case. The goal is to minimize the transmission rate R_1 from the server to the mirrors.

This single-layer setting was recently studied in [7], [8], where the authors proposed a coded caching scheme. The authors showed that rate $R_1 = r(M/N,K)$ is feasible in this setting, where $r(\cdot,\cdot)$ is given by

$$r\left(\frac{M}{N},K\right) \triangleq \left[K\cdot\left(1-\frac{M}{N}\right)\cdot\frac{N}{KM}\left(1-\left(1-\frac{M}{N}\right)^K\right)\right]^+ \quad (2)$$

with $[x]^+ \triangleq \max\{x,0\}$. Furthermore, it is shown in [8] that this achievable rate R_1 is within a constant factor of the minimum achievable rate for this single-layer setting for any values of N, K, and M. We will refer to the placement and delivery procedures of the single-layer coded caching scheme as BasePlacement(N,K,M) and BaseDelivery(N,K,M), respectively, and we illustrate them in Example 1.

Example 1 (*Single-Layer Coded Caching [8]*)**.** Consider the single-layer setting as described above with $N = 2$ files and $K = 2$ mirrors, each of size $M_1 = M \in [0,2]$. For ease of notation, denote the files by A and B. In the placement phase, each mirror stores a subset of $MF/N = MF/2$ bits of each of the two files, chosen uniformly and independently at random. Each bit of a file is thus stored in a given mirror with probability $M/N = M/2$.

Consider file A and notice that we can view it as being composed of $2^K = 4$ subfiles $A = (A_\emptyset, A_1, A_2, A_{1,2})$, where

[1]For example, in a video application such as Netflix, each "file" corresponds to a short segment of a video, perhaps a few seconds to a minute long. If there are 1000 different popular movies of length 100 minutes each, this would correspond to more than $100,000$ different files.

$A_{\mathcal{S}}$ denotes the bits of file A which are exclusively stored in the mirrors in $\mathcal{S}$. For example, A_1 denotes the bits of file A which are stored only in mirror 1, and $A_{1,2}$ denotes the bits of file A which are available in both mirrors 1 and 2. For large enough file size F, we have by the law of large numbers that for any subset $\mathcal{S}$, $|A_{\mathcal{S}}| \approx \left(\frac{M}{2}\right)^{|\mathcal{S}|}\left(1-\frac{M}{2}\right)^{2-|\mathcal{S}|}F$. File B can similarly be partitioned into subfiles.

In the delivery phase, suppose for example that the first user requests file A and the second user requests file B. Then, the server transmits $A_2 \oplus B_1$, $A_\emptyset$, and $B_\emptyset$ where $\oplus$ denotes bit-wise XOR. It is easy to check that this enables each mirror to recover its respective requested file. The number of bits transmitted by the server is given by $\frac{M}{2}\left(1-\frac{M}{2}\right)F + 2\left(1-\frac{M}{2}\right)^2 F$, which agrees with the expression in (2). $\Diamond$

While the above discussion focuses on $K_2 = 1$ user accessing each mirror, the achievable scheme can easily be extended to $K_2 > 1$ users by performing the delivery phase in K_2 stages with one unique user per mirror active in each stage. The resulting rate over the first link is

$$R_1 = K_2 \cdot r(M_1/N, K_1). \tag{3}$$

IV. Main Results

As the main result of this paper, we provide an approximation of the feasible rate region $\mathcal{R}^\star(M_1, M_2)$ for the general hierarchical caching problem with two layers. We start by introducing some notation. For $\alpha, \beta \in [0,1]$, define the rates

$$R_1(\alpha,\beta) \triangleq \alpha K_2 \cdot r\left(\frac{M_1}{\alpha N}, K_1\right) + (1-\alpha)\cdot r\left(\frac{(1-\beta)M_2}{(1-\alpha)N}, K_1K_2\right), \tag{4a}$$

$$R_2(\alpha,\beta) \triangleq \alpha \cdot r\left(\frac{\beta M_2}{\alpha N}, K_2\right) + (1-\alpha)\cdot r\left(\frac{(1-\beta)M_2}{(1-\alpha)N}, K_2\right), \tag{4b}$$

where $r(\cdot,\cdot)$ is defined in (2) in Section III. Next, consider the following region:

Definition. For memory sizes $M_1, M_2 \geq 0$, define

$$\mathcal{R}_C(M_1,M_2) \triangleq \left\{\left(R_1(\alpha,\beta), R_2(\alpha,\beta)\right) : \alpha,\beta \in [0,1]\right\} + \mathbb{R}_+^2, \tag{5}$$

where $\mathbb{R}_+^2$ denotes the positive quadrant, $R_1(\alpha,\beta), R_2(\alpha,\beta)$ are defined in (4), and the addition corresponds to the Minkowski sum between sets.

As will be discussed in more detail later, the region $\mathcal{R}_C(M_1,M_2)$ is the rate region achieved by appropriately sharing the available memory between two coded caching schemes during the placement phase and then using each scheme to recover a certain fraction of the requested files during the delivery phase. Each of these two schemes is responsible for one of the two terms in $R_1(\alpha,\beta)$ and $R_2(\alpha,\beta)$. The parameters α and β dictate what fraction of the memory and what fraction of each file is allocated to each of these two schemes. The region $\mathcal{R}_C(M_1,M_2)$ is thus the rate region achieved by all possible choices of the parameters α and β.

Our main result shows that, for any memory sizes M_1, M_2, the region $\mathcal{R}_C(M_1,M_2)$ approximates the feasible rate region $\mathcal{R}^\star(M_1,M_2)$ up to a constant additive and multiplicative gap.

Theorem 1. *Consider the hierarchical caching problem in Fig. 1 with N files, K_1 mirrors, and K_2 users accessing each mirror. Each mirror and user cache has a normalized memory size of M_1 and M_2, respectively. Then we have*

$$\mathcal{R}_C(M_1,M_2) \subseteq \mathcal{R}^\star(M_1,M_2) \subseteq c_1 \cdot \mathcal{R}_C(M_1,M_2) - c_2,$$

where $\mathcal{R}^\star(M_1,M_2)$ and $\mathcal{R}_C(M_1,M_2)$ are defined in (1) *and* (5)*, respectively, and where c_1 and c_2 are finite positive constants independent of all the problem parameters.*

The proof of Theorem 1 actually shows a slightly stronger result than stated in the theorem. Recall that the parameters α and β control the weights of the split between the two simple coded caching schemes mentioned above. In general, one would expect a tension between the rates $R_1(\alpha,\beta)$ and $R_2(\alpha,\beta)$ over the first and second hops of the network. In other words, the choice of α and β minimizing the rate $R_1(\alpha,\beta)$ over the first hop will in general *not* minimize the rate $R_2(\alpha,\beta)$ over the second hop.

However, the proof of Theorem 1 shows that there exists $\alpha^\star$ and $\beta^\star$ (depending on N, M_1, M_2, K_1, and K_2) such that $R_1(\alpha^\star,\beta^\star)$ and $R_2(\alpha^\star,\beta^\star)$ are *simultaneously* approximately minimized. Thus, surprisingly, there is in fact *no tension* between the rates over the first hop and the second hop for the optimal hierarchical caching scheme up to a constant additive and multiplicative gap. This situation is illustrated in Fig. 2.

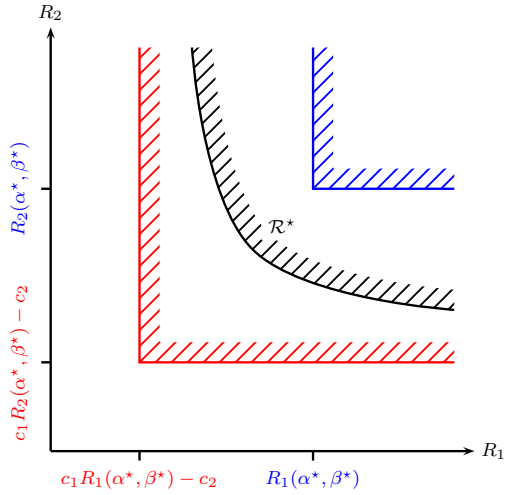

Fig. 2. For fixed memory values M_1 and M_2, the figure qualitatively depicts the feasible rate region $\mathcal{R}^\star$ and its bounds. As shown in the figure, the feasible rate region $\mathcal{R}^\star$ can be bounded by two rectangular regions with corner points $\left(R_1(\alpha^\star,\beta^\star), R_2(\alpha^\star,\beta^\star)\right)$ and $\left(c_1R_1(\alpha^\star,\beta^\star)-c_2, c_1R_2(\alpha^\star,\beta^\star)-c_2\right)$. Thus, up to a constant additive and multiplicative gap, there is no tension between the optimal rates R_1 and R_2 for the hierarchical caching problem.

For the choice of $\alpha^\star$ and $\beta^\star$, we consider three different

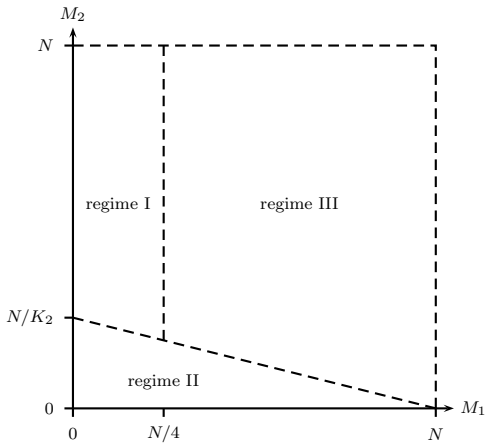

Fig. 3. Different regimes for $\alpha^\star$ and $\beta^\star$.

regimes of M_1 and M_2 as depicted in Fig. 3. We set

$$(\alpha^\star, \beta^\star) \triangleq \begin{cases} \left(\frac{M_1}{N}, \frac{M_1}{N}\right) & \text{in regime I,} \\ \left(\frac{M_1}{M_1 + M_2 K_2}, 0\right) & \text{in regime II,} \\ \left(\frac{M_1}{N}, \frac{1}{4}\right) & \text{in regime III.} \end{cases}$$

Substituting this choice into (4), we can derive the achievable rates for the proposed scheme. To complete the proof of Theorem 1, we compare these rates to appropriately defined cut-set lower bounds and show that they match in each of the three regimes up to a constant additive and multiplicative gap. For brevity, we only describe the achievable scheme in this paper and refer the reader to [13] for further details.

V. Caching Schemes

In this section, we introduce a class of caching schemes for the hierarchical caching problem. In Sections V-A and V-B, we employ the single-layer coded caching approach from Section III to construct two caching schemes for networks with two layers of caches. We will see in Section V-C how to combine these two schemes to yield a near-optimal scheme for the hierarchical caching problem.

A. Caching Scheme A

This scheme places content in the mirrors so that using the server transmission and their own content, each mirror can recover all the files requested by their attached users. In turn, each mirror then acts as a server for delivering these files to its attached users. Content is stored in the attached user caches so that by using the mirror transmission and their cache content, each user can recover its requested file. See Fig. 4 for an illustration of the scheme. By (2) and (3), the rates over the links from the server to the mirrors and from each mirror to its users are given by

$$R_1^A \triangleq K_2 \cdot r\left(\frac{M_1}{N}, K_1\right), \quad R_2^A \triangleq r\left(\frac{M_2}{N}, K_2\right). \tag{6}$$

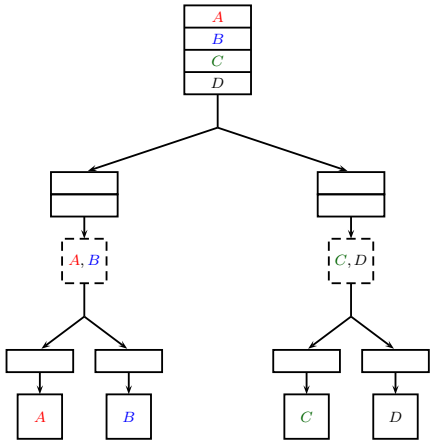

Fig. 4. Caching scheme A for a system with $K_1 = 2$ mirrors and $K_2 = 2$ users per mirror. Scheme A uses a decode-and-forward type approach to apply the single-layer coded caching scheme to a network with two layers. We independently cache content in each of the layers during the placement phase. In the delivery phase, the mirrors decode all the files requested by their users and re-encode them for their children. For example, in the figure mirror 1 decodes files A, B and re-encodes them for the two attached users.

Example 2. Consider the setup in Fig. 4 with $N = 4$ files, $K_1 = 2$ mirrors, and $K_2 = 2$ users per mirror. The mirror and user cache memory sizes are $M_1 = 2$ and $M_2 = 1$, respectively. For ease of notation, denote the files by A, B, C, and D. Using the BasePlacement procedure from Section III, each mirror independently stores a random $F/2$-bit subset of every file, and each user cache independently stores a random $F/4$-bit subset of every file.

In the delivery phase, assume the four users request files A, B, C, and D, respectively. The server uses the BaseDelivery procedure to enable the first mirror to recover files A and B and to enable the second mirror to recover files C and D. This uses a rate of $R_1^A = 2 \cdot r(1/2, 2)$. Mirror 1 then uses the BaseDelivery procedure to re-encode the files A and B for its attached users. Similarly, mirror 2 uses the BaseDelivery procedure to re-encode the files C and D for its attached users. This uses a rate of $R_2^A = r(1/4, 2)$. ◊

B. Caching Scheme B

This scheme places content across the $K_1 K_2$ user caches so that using the server transmissions and its own cache content, each user can recover its requested file. The storage capabilities of the mirrors in the network are completely ignored and the mirrors are only used to forward relevant parts of the server transmissions to the corresponding users. See Fig. 5 for an illustration. By (2) and [8, Section V.A], the rates over the links from the server to the mirrors and each mirror to its attached users are given by

$$R_1^B \triangleq r\left(\frac{M_2}{N}, K_1 K_2\right), \quad R_2^B \triangleq r\left(\frac{M_2}{N}, K_2\right). \tag{7}$$

Example 3. Consider the setup in Fig. 5 with $N = 4$ files, $K_1 = 2$ mirrors, and $K_2 = 2$ users per mirror. The user cache memory size is $M_2 = 1$ (the mirror memory size M_1 is irrelevant here). For ease of notation, let the files be A, B, C,

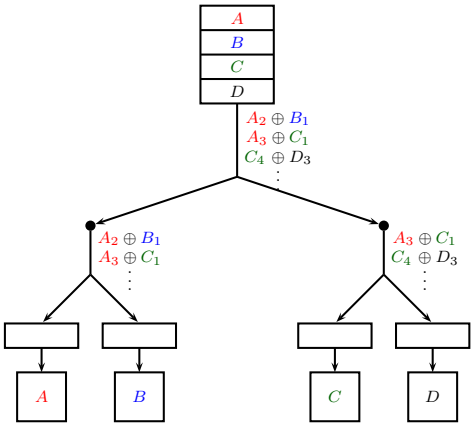

Fig. 5. Caching scheme B for a system with $K_1 = 2$ mirrors and $K_2 = 2$ users per mirror. Scheme B ignores the memory at the mirrors and uses the single-layer coded caching scheme directly between the server and the users. The mirrors are only used to forward relevant messages to their users.

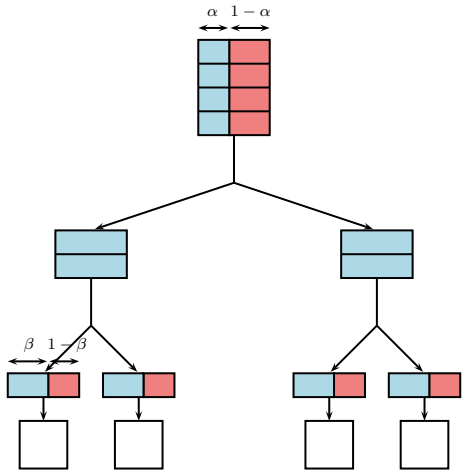

Fig. 6. Generalized caching scheme for a system with $K_1 = 2$ mirrors and $K_2 = 2$ users per mirror. For given α and β, the system is split into two disjoint subsystems. We use caching scheme A for delivering the first parts of the files over the first subsystem and use caching scheme B for delivering the second parts of the files over the second subsystem.

and D. Using the BasePlacement procedure, each user cache independently stores a random $F/4$-bit subset of every file.

In the delivery phase, assume the four users request files A, B, C, and D, respectively. The server uses the BaseDelivery procedure to enable the users to recover their requested files. This uses a rate of $R_1^B = r(1/4, 4)$. Let us focus on mirror 1. Since its attached users request files A and B, it forwards every server transmission including parts of either of those files. Thus, mirror 1 transmits $A_2 \oplus B_1$ but does not forward $C_4 \oplus D_3$. This uses a rate of $R_2^B = r(1/4, 2)$. $\Diamond$

C. Generalized Caching Scheme

The generalized scheme divides the system into two subsystems, the first one operated according to caching scheme A and the second one according to caching scheme B. Fix parameters $\alpha, \beta \in [0, 1]$. The first subsystem includes the entire memory of each mirror and a β-fraction of each user cache memory. The second subsystem includes the remaining $(1-\beta)$-fraction of each user cache memory. We split each file into two parts of size αF and $(1-\alpha)F$ bits, respectively. We use scheme A from Section V-A to store and deliver the first parts of the files. Similarly, we use scheme B from Section V-B for the second parts of the files. See Fig. 6 for an illustration.

Since our system is a composition of two disjoint subsystems, the net rate over each link is the sum of the corresponding rates in the two subsystems. From (6), the rates R_1^1, R_2^1 required by scheme A over the first subsystem are

$$R_1^1 = \alpha K_2 \cdot r\left(\frac{M_1}{\alpha N}, K_1\right), \quad R_2^1 = \alpha \cdot r\left(\frac{\beta M_2}{\alpha N}, K_2\right). \quad (8)$$

Similarly, from (7), the rates R_1^2, R_2^2 required by scheme B over the second subsystem are

$$R_1^2 = (1-\alpha) \cdot r\left(\frac{(1-\beta)M_2}{(1-\alpha)N}, K_1 K_2\right), \quad (9a)$$

$$R_2^2 = (1-\alpha) \cdot r\left(\frac{(1-\beta)M_2}{(1-\alpha)N}, K_2\right). \quad (9b)$$

Combining (8) and (9), and comparing with (4), we observe that the net rates of the generalized caching scheme are $R_1 = R_1(\alpha, \beta)$ and $R_2 = R_2(\alpha, \beta)$.

Acknowledgement

N. Karamchandani and S. Diggavi were supported in part by NSF award 1136174 and MURI award AFOSR FA9550-09-064. The aforementioned authors also gratefully acknowledge support by Intel and Verizon.

References

[1] D. Wessels, *Web Caching*. O'Reilly, 2001.
[2] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, pp. 1478–1486, Mar. 2010.
[3] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," *IEEE/ACM Trans. Netw.*, vol. 21, pp. 566–579, Apr. 2013.
[4] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, "On the scale and performance of cooperative web proxy caching," in *Proc. ACM SOSP*, pp. 16–31, 1999.
[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, pp. 126–134, Mar. 1999.
[6] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proc. ACM CoNEXT*, pp. 1–12, Nov. 2010.
[7] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *arXiv:1209.5807 [cs.IT]*, Sept. 2012. To appear in IEEE Trans. Inf. Theory.
[8] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *arXiv:1301.5848 [cs.IT]*, Jan. 2013. To appear in IEEE/ACM Trans. Netw.
[9] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *arXiv:1308.0178 [cs.IT]*, Aug. 2013.
[10] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *arXiv:1311.3646 [cs.IT]*, Nov. 2013.
[11] A. Chankhunthod, P. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical internet object cache," in *Proc. USENIX ATEC*, pp. 153–163, Jan. 1996.
[12] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Placement algorithms for hierarchical cooperative caching," in *Proc. ACM-SIAM SODA*, pp. 586–595, Jan. 1999.
[13] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. Diggavi, "Hierarchical coded caching," *arXiv:1403.7007 [cs.IT]*, Mar. 2014.