ISIT'2013 1569717269

Fundamental Limits of Caching

Mohammad Ali Maddah-Ali
Bell Labs, Alcatel-Lucent
New Jersey, USA
mohammadali.maddah-ali@alcatel-lucent.com

Urs Niesen
Bell Labs, Alcatel-Lucent
New Jersey, USA
urs.niesen@alcatel-lucent.com

***Abstract*—Caching is a technique to reduce peak traffic rates by prefetching popular content in memories at the end users. This paper proposes a novel caching approach that can achieve a significantly larger reduction in peak rate compared to previously known caching schemes. In particular, the improvement can be on the order of the number of end users in the network. Conventionally, cache memories are exploited by delivering requested contents in part locally rather than through the network. The gain offered by this approach, which we term *local caching gain*, depends on the *local* cache size (i.e., the cache available at each individual user). In this paper, we introduce and exploit a second, *global*, caching gain, which is not utilized by conventional caching schemes. This gain depends on the aggregate *global* cache size (i.e., the cumulative cache available at all users), even though there is no cooperation among the caches.**

To evaluate and isolate these two gains, we introduce a new, information-theoretic formulation of the caching problem focusing on its basic structure. For this setting, the proposed scheme exploits both local and global caching gains, leading to a multiplicative improvement in the peak rate compared to previously known schemes. Moreover, we argue that the performance of the proposed scheme is within a constant factor from the information-theoretic optimum for all values of the problem parameters.

I. Introduction

The high temporal variability of network traffic results in communication systems that are congested during peak-traffic times and underutilized during off-peak times. One approach to reduce peak traffic is to take advantage of memories distributed across the network (at end users, routers, ...) to duplicate content. This duplication of content, called content placement or caching, is performed during off-peak hours when network resources are abundant. During peak hours, when network resources are scarce, user requests can then be served exploiting these caches to reduce network congestion. In this manner, caching effectively shifts traffic from peak to off-peak hours, thereby smoothing out traffic and reducing congestion.

From the above discussion, we can see that the caching problem consists of two distinct phases. The first phase is the *placement phase*, which is based solely on the statistics of the user demands. In this phase, the network is not congested, and the main limitation is the size of the cache memories. The second phase is the *delivery phase*, which is performed once the actual demands of the users have been revealed. In this phase, the network is congested, and the main limitation is the rate required to serve the requested content.

Various versions of this problem have been studied, with the focus being mainly on exploiting the history or statistics of the user demands [1]–[7]. In these papers, the operation of the delivery phase is fixed to consist of simple orthogonal unicast or multicast transmissions. Assuming this method of delivery, the content placement is then optimized. In this approach, the main gain of caching derives from making popular content available locally. If a user requests some content that is stored in its cache, it can be served from its local memory. We hence call this the *local caching gain*. This gain is relevant if the local cache memory is large enough such that a sizable fraction of the total (popular) content can be stored locally. On the other hand, if the size of the local caches is small compared to the total amount of content, then this gain is insignificant.

In this paper, we propose a novel caching approach that, in addition to the local caching gain, is able to achieve a *global caching gain*. This gain derives from *jointly* optimizing both the caching and delivery phases, ensuring that in the delivery phase several *different* demands can be satisfied with a single multicast transmission. Since the content placement is performed without knowledge of the actual demands, it must be carefully designed such that these multicasting opportunities are created *simultaneously* for all possible requests in the delivery phase. We show that this global caching gain is relevant if the aggregate global cache size is large enough compared to the total amount of content. Thus, even though the caches cannot cooperate, the sum of the cache sizes becomes an important system parameter.

The remainder of this paper is organized as follows. Section II introduces an information-theoretic formulation of the caching problem. Section III presents main results, which are illustrated with examples in Section IV. A formal description of the proposed caching algorithm is given in Section V. Finally, Section VI explores connections between the caching problem and the index and network coding problems. Due to space constraints, all results are presented here without proofs; these can be found in the full version of the paper [8].

II. Problem Setting

We introduce a new, information-theoretic formulation of the caching problem, consisting of a server connected through a shared, error-free link to K users, as illustrated in Fig. 1. The server has access to a database of N files $W_1, \dots, W_N$ each of size F bits. Each user k has an isolated cache memory Z_k of size MF bits for some real number $M \in [0, N]$.

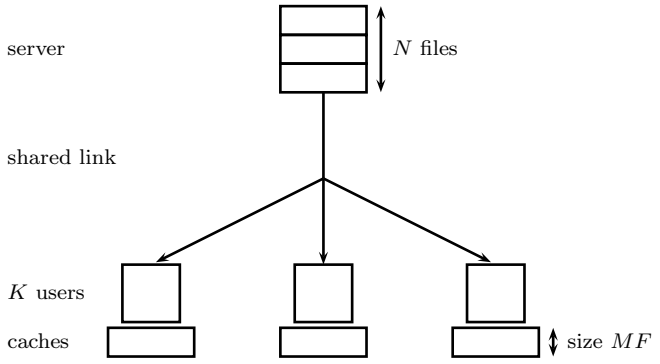

Fig. 1. The caching problem with $N = 3$ files and $K = 3$ users.

The system operates in two phases: a *placement phase* and a *delivery phase*. In the placement phase, each user k is allowed to fill the content of its cache Z_k as an arbitrary function of the database $W_1, \ldots, W_N$. In the delivery phase, only the server has access to the database of files. Each user k requests one of the files W_{d_k} in the database. The server is informed of these requests and proceeds by transmitting a signal of size RF bits over the shared link. Using the content Z_k of its cache and the signal received over the shared link, each user k aims to reconstruct its requested file W_{d_k}.

We say that a memory-rate pair (M, R) is *achievable for requests* $d_1, \ldots, d_K$ if every user k can recover its desired file W_{d_k} (with high probability as $F \to \infty$). A memory-rate pair (M, R) is said to be *achievable* if this pair is achievable for every possible request $d_1, \ldots, d_K$ in the delivery phase. We denote by $R^\star(M)$ the smallest rate R such that (M, R) is achievable. The function $R^\star(M)$ describes the *memory-rate tradeoff* for the caching problem. The aim of this paper is to characterize this memory-rate tradeoff. In other words, we aim to find the minimum rate of communication over the shared link at which all possible demand tuples can be satisfied.

For ease of exposition we consider only the most relevant case $N \geq K$, i.e., when the number of files is at least as large as the number of users in the system. For the general case, the reader is referred to the full version of the paper [8]. Furthermore, for simplicity, the problem statement here makes some idealized assumptions, which can be relaxed [9].

We illustrate these definitions with the example of the caching strategy employed by conventional caching systems.

Example 1 (*Conventional Scheme*)**.** One possible strategy is to place an equal fraction of each of the N files in each of the caches. For a memory size of MF, each user thus caches the same M/N fraction of each file in the placement phase. In the delivery phase, the server simply transmits the remaining $1 - M/N$ fraction of any requested file over the shared link. Clearly, each user can recover its requested file from the content of its local cache and the signal sent over the shared link. In the worst case, users request different files—the worst-case delivery rate for this caching scheme is thus

$$K \cdot (1 - M/N). \tag{1}$$

We refer to this approach as the *conventional caching scheme*. The gain of caching in this approach is $1 - M/N$, which is achieved because a M/N fraction of each file is available locally at each user. As we will see, this conventional caching strategy can be significantly improved upon. ♢

III. Main Results

The first result presents an achievable rate, yielding an upper bound on the memory-rate tradeoff $R^\star(M)$.

Theorem 1. *For N files and $K \leq N$ users each with cache of size $M \in \frac{N}{K} \cdot \{0, 1, \ldots, K\}$,*

$$R^\star(M) \leq R(M) \triangleq K \cdot (1 - M/N) \cdot \frac{1}{1 + KM/N}$$

is achievable. For general $0 \leq M \leq N$, the lower convex envelope of these points is achievable.

The algorithm achieving the rate in Theorem 1 is presented in Section V. The achievable rate $R(M)$ in Theorem 1 consists of three distinct factors. The first factor in $R(M)$ is K. This is the worst-case rate without caches at the users (i.e., $M = 0$).

The second factor in $R(M)$ in Theorem 1 is $1 - M/N$. Referring to (1) in Example 1, we see that this term appears also as the gain of the conventional caching scheme. We call this factor the *local caching gain*. Observe that this local gain is a function of the normalized *local* memory size M/N, and it is relevant whenever M is on the order of N.

Finally, the third factor in $R(M)$ is $\frac{1}{1+KM/N}$, which we call the *global caching gain*. This gain is a function of the normalized *global* or *cumulative* memory size KM/N, and it is relevant whenever KM is on the order of (or larger than) N. This global gain is to be interpreted as a multicasting gain available simultaneously for all possible demands. Note that, since the number of users is smaller than the number of files, in the worst case all users request different files. Hence, there are no natural multicasting opportunities. The scheme proposed in Theorem 1 carefully designs the content placement in order to create coded multicasting opportunities in the delivery phase even among users that request different files. Since the placement phase is performed without knowledge of the actual demands, care must be taken to ensure that the same multicasting opportunities are created simultaneously for every possible set of requests in the delivery phase.

We point out that the conventional caching scheme introduced in Example 1 achieves only the local caching gain, whereas the caching strategy proposed in Theorem 1 achieves both the local as well as the global caching gains. The following three examples compare these two gains.

Example 2. Consider the performance of a caching system with $N = 50$ files and $K = 50$ users as shown in Fig. 2. When each user has a cache memory large enough to store $M = 10$ files, the rate of the conventional scheme corresponds to sending 40 files over the shared link, while the proposed scheme achieves a rate corresponding to sending only 3.6 files: a reduction by a factor 11 in rate. ♢

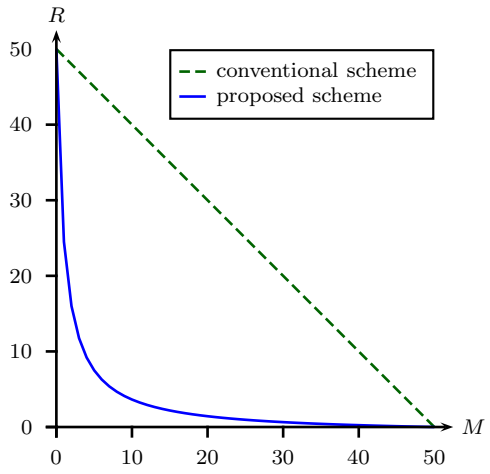

Fig. 2. Rate R required in the delivery phase as a function of memory size M for a system with $N = 50$ files and $K = 50$ users. The dashed green curve shows the performance of conventional caching strategies. The solid blue curve is the performance of the proposed new caching scheme. For a memory able to store $M = 10$ files, the rate required by the proposed scheme is a factor 11 smaller than the rate required by conventional schemes.

Example 3 ($\Theta(K)$ *Improvement in Rate*)**.** Consider a situation with the same number of users as files, i.e., $N = K$. Assume each user has enough local cache memory for half of the files so that $M = N/2$. Then the local caching gain is $1/2$ and the global caching gain is $1/(1 + K/2)$. By (1), the conventional scheme achieves a rate of $K/2$. On the other hand, by Theorem 1, the scheme proposed in this paper achieves a rate of $(K/2)/(1 + K/2) < 1$: a reduction by more than a factor $K/2$ in rate compared to the conventional scheme. The effect of this improvement can be seen in Fig. 2. $\Diamond$

Example 4 ($\Theta(K)$ *Improvement in Slope*)**.** We now compare the performance of the proposed and conventional schemes for small values of the local cache size M. We consider again the case $N = K$. From (1), the slope of the rate of the conventional scheme as a function of cache size M around $M = 0$ is equal to -1. On the other hand, by Theorem 1, the scheme proposed in this paper has slope less than $-K/2$ around $M = 0$. Therefore, the rate of the proposed scheme reduces with the local cache size at least $K/2$ times faster than the conventional scheme. Comparing the rates of the conventional and the proposed schemes in Fig. 2 for small values of M illustrates the effect of this improvement. $\Diamond$

Having established an upper bound on the memory-rate tradeoff $R^\star(M)$, we proceed with a lower bound on it.

Theorem 2. *For N files and K users each with cache of size $0 \le M \le N$,*

$$R^\star(M) \ge \max_{s\in\{1,\ldots,\min\{N,K\}\}} \Big(s - \frac{s}{\lfloor N/s \rfloor} M\Big).$$

The proof of Theorem 2 is based on a cut-set bound argument. Comparing the achievable rate $R(M)$ in Theorem 1 with the lower bound in Theorem 2, we obtain the following approximation result for the memory-rate tradeoff $R^\star(M)$.

Theorem 3. *For N files and $K \le N$ users each with cache of size $0 \le M \le N$,*

$$1 \le \frac{R(M)}{R^\star(M)} \le 12,$$

with the achievable rate $R(M)$ as defined in Theorem 1.

The bound $R(M)/R^\star(M) \le 12$ on the approximation ratio of the proposed scheme is somewhat loose due to the analytical bounding techniques that were used. Numerical simulations suggest that $R(M)/R^\star(M) \le 5$ for all N, K, and M.

Theorem 3 shows that the rate $R(M)$ of the proposed scheme in Theorem 1 is close to the information-theoretic optimum $R^\star(M)$ for all values of the system parameters. More precisely, it shows that no scheme can improve upon the rate $R(M)$ of the proposed scheme by more than a constant factor. This also implies that the local and global caching gains identified in this paper are fundamental: there are no other significant caching gains beyond these two.

IV. Examples

Before stating the general caching algorithm in Section V, we illustrate it with two simple examples.

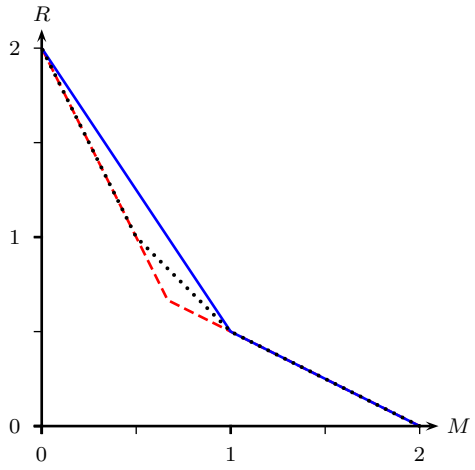

Fig. 3. Memory-rate tradeoff for $N = 2$ files $K = 2$ users. The achievable rate $R(M)$ from Theorem 1 is indicated by the solid blue curve, and the lower bound on $R^\star(M)$ from Theorem 2 by the dashed red curve. For $N = K = 2$, $R^\star(M)$ can be found exactly and is indicated by the dotted black curve.

Example 5. Let $N = K = 2$, so that there are two files, say $W_1 = A, W_2 = B$, and two users each with cache memory of size M. The upper and lower bounds in Theorems 1 and 2 on the memory-rate tradeoff $R^\star(M)$ are depicted in Fig. 3. To illustrate the proof techniques, we now show how these two bounds are derived for this simple setting.

We start with the upper bound in Theorem 1. We focus on the corner points of the memory-rate curve $R(M)$ occurring at M equal to 0, 1, and 2. Consider first the two extreme cases $M = 0$ and $M = 2$. If $M = 0$, the server can always transmit both files A and B over the shared link. Since this satisfies every possible request, the (M, R) pair $(0, 2)$ is achievable. If $M = 2$, each user can cache both files A and B in the

placement phase. Therefore, no communication is needed in the delivery phase and the (M, R) pair $(2, 0)$ is achievable.

Consider then cache size $M = 1$. The caching scheme achieving the upper bound in Theorem 1 is as follows. We split both files A and B into two subfiles of equal size, i.e., $A = (A_1, A_2)$ and $B = (B_1, B_2)$. In the placement phase, we set $Z_1 = (A_1, B_1)$ and $Z_2 = (A_2, B_2)$. In words, each user caches one exclusive part of each file. For the delivery phase, assume for example that user one requests file A and user two file B. Given that user one already has subfile A_1 of A, it only needs to obtain the missing subfile A_2, which is cached in the second user's memory Z_2. Similarly, user two only needs to obtain the missing subfile B_1, which is cached in the first user's memory Z_1. In other words, each user has one subfile of the file that the other user needs.

The server can in this case simply transmit $A_2 \oplus B_1$, where $\oplus$ denotes bitwise XOR. Since user one already has B_1, it can recover A_2 from $A_2 \oplus B_1$. Similarly, since user two already has A_2, it can recover B_1 from $A_2 \oplus B_1$. Thus, the signal $A_2 \oplus B_1$ received over the shared link helps both users to effectively exchange the missing subfiles.

All other requests can also be efficiently served: if user one requests B and user two A, the server sends $A_1 \oplus B_2$; if both users request A, it sends $A_1 \oplus A_2$; if both users request B, it sends $B_1 \oplus B_2$. This proves (M, R) pair $(1, 1/2)$ is achievable.

It is worth pointing out that in each case, the server sends a single coded multicast transmission to satisfy two (possibly different) user requests. Moreover, these coded multicasting opportunities are available simultaneously for all four possible user request pairs. This availability of *simultaneous* multicasting opportunities, enabled by careful content placement, is critical, since the placement phase has to be performed without knowledge of the actual demands in the delivery phase.

So far, we have shown that the (M, R) pairs $(0, 2)$, $(1, 1/2)$, and $(2, 0)$ are achievable. On the other hand, by dividing cache memories proportionally, it is easy to see that if two points (M_1, R_1) and (M_2, R_2) are achievable, the line connecting them is also achievable. Together, this proves the achievability of the solid blue curve in Fig. 3, which coincides with the upper bound stated in Theorem 1.

The lower bound in Theorem 2 can be evaluated to

$$R^\star(M) \geq \max\{2 - 2M, 1 - M/2\},$$

which yields the dashed red curve in Fig. 3.

For $N = K = 2$, the memory-rate tradeoff can in fact be found exactly (see the dotted black curve in Fig. 3). This shows that, while the bounds in Theorems 1 and 2 are sufficient to characterize the memory-rate tradeoff $R^\star(M)$ to within a bounded multiplicative gap, neither the achievable region nor the converse are tight in general. The details of this derivation can be found in [8]. $\Diamond$

Example 6. Let $N = K = 3$ so that there are three users and three files, say $W_1 = A$, $W_2 = B$, and $W_3 = C$. We focus again on the corner points of $R(M)$ occurring at M equal to 0, 1, 2, and 3. Trivially, (M, R) pairs $(0, 3)$ and $(3, 0)$ are achievable. The nontrivial cases are $M = 1$ and $M = 2$.

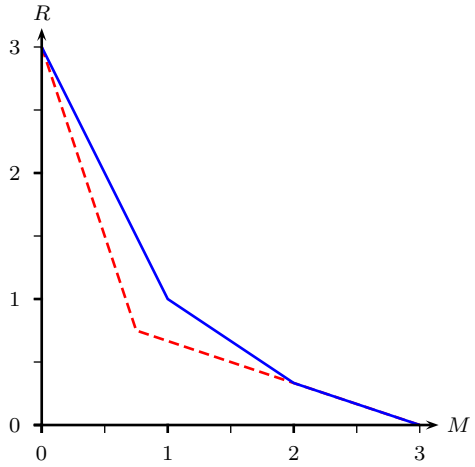

Fig. 4. Memory-rate tradeoff for $N = 3$ files and $K = 3$ users. The achievable rate $R(M)$ from Theorem 1 is indicated by the solid blue curve, and the lower bound on $R^\star(M)$ from Theorem 2 by the dashed red curve.

Consider first caches of size $M = 1$. We split each file into three subfiles of equal size, i.e., $A = (A_1, A_2, A_3)$, $B = (B_1, B_2, B_3)$, and $C = (C_1, C_2, C_3)$. In the placement phase, the cache content of user k is selected as $Z_k = (A_k, B_k, C_k)$.

For the delivery phase, assume for example that user one requests file A, user two file B, and user three file C. Then the missing subfiles are A_2 and A_3 for user one, B_1 and B_3 for user two, and C_1 and C_2 for user three. Given the cache contents, users one and two aim to exchange A_2 and B_1, users one and three aim to exchange A_3 and C_1, and users two and three aim to exchange B_3 and C_2. By sending $(A_2 \oplus B_1, A_3 \oplus C_1, B_3 \oplus C_2)$, the server enables all of these three exchanges. All other requests can be satisfied in a similar manner. Since each subfile has rate $1/3$, the proposed scheme achieves a rate of 1, and therefore (M, R) pair $(1, 1)$ is achievable.

Observe that, through careful cache placement, we have again created coded multicasting opportunities for any two users even with different demands. Moreover, these coded multicasting opportunities are available simultaneously for all 27 possible triples of user requests.

Consider next caches of size $M = 2$. We again split each file into three subfiles of equal size. However, it will be convenient to label these subfiles differently, namely $A = (A_{12}, A_{13}, A_{23})$, $B = (B_{12}, B_{13}, B_{23})$, and $C = (C_{12}, C_{13}, C_{23})$. In the placement phase, we set

$$Z_1 = (A_{12}, A_{13}, B_{12}, B_{13}, C_{12}, C_{13}),$$
$$Z_2 = (A_{12}, A_{23}, B_{12}, B_{23}, C_{12}, C_{23}),$$
$$Z_3 = (A_{13}, A_{23}, B_{13}, B_{23}, C_{13}, C_{23}).$$

This content placement can be understood as follows. Let $\mathcal{T}$ be a subset of two elements of $\{1, 2, 3\}$. Then subfiles $A_\mathcal{T}$, $B_\mathcal{T}$, $C_\mathcal{T}$ are placed into the cache of user k if $k \in \mathcal{T}$. E.g., A_{13} is cached at users one and three since in this case $\mathcal{T} = \{1, 3\}$.

For the delivery phase, assume for example that user one requests file A, user two file B, and user three file C. Observe that user one misses subfile A_{23}, which is available at users

two and three. User two misses subfile B_{13}, which is available at users one and three. And user three misses subfile C_{12}, which is available at users one and two. In other words, the three users would like to exchange the subfiles A_{23}, B_{13}, C_{12}. By sending $A_{23} \oplus B_{13} \oplus C_{12}$ over the shared link, the server enables this exchange. From its cache content, each user can recover the missing subfile. All other requests can be satisfied in a similar manner. The rate of transmission in the delivery phase is $1/3$, and therefore (M, R) pair $(2, 1/3)$ is achievable. This approach again creates simultaneous coded multicasting opportunities, but this time for all three users together.

The arguments so far show that the solid blue curve in Fig. 4 is achievable. This coincides with the upper bound in Theorem 1. The lower bound in Theorem 2 yields

$$R^\star(M) \geq \max\{3 - 3M, 1 - M/3\},$$

which results in the dashed red curve in Fig. 4. $\Diamond$

V. An Order-Optimal Caching Algorithm

We now present the general achievable scheme. The presentation here is brief; the reader is referred to [8] for details.

To simplify the description, we use the notation

$$[N] \triangleq \{1, \ldots, N\}, \qquad [K] \triangleq \{1, \ldots, K\}.$$

We now give the caching algorithm for N files, K users. We focus on the corner points of $R(M)$, which occur for any cache size M such that MK/N is an integer less than K.

procedure PLACEMENT($W_1, \ldots, W_N$)
  $t \leftarrow MK/N$
  $\mathfrak{T} \leftarrow \{\mathcal{T} \subset [K] : |\mathcal{T}| = t\}$
  for $n \in [N]$ **do**
    split W_n into $(W_{n,\mathcal{T}} : \mathcal{T} \in \mathfrak{T})$ of equal size
  end for
  for $k \in [K]$ **do**
    $Z_k \leftarrow (W_{n,\mathcal{T}} : n \in [N], \mathcal{T} \in \mathfrak{T}, k \in \mathcal{T})$
  end for
end procedure

procedure DELIVERY($W_1, \ldots, W_N, d_1, \ldots, d_K$)
  $t \leftarrow MK/N$
  $\mathfrak{S} \leftarrow \{\mathcal{S} \subset [K] : |\mathcal{S}| = t + 1\}$
  server sends $(\oplus_{k \in \mathcal{S}} W_{d_k, \mathcal{S}\setminus\{k\}} : \mathcal{S} \in \mathfrak{S})$
end procedure

The placement phase of the algorithm is designed such that, in the delivery phase, the caches enable coded multicasting between $MK/N + 1$ users with different demands. These coded multicasting opportunities are available simultaneously for all N^K possible user demands. This is again critical, since the placement phase has to be designed without knowledge of the actual demands in the delivery phase.

The efficiency of the proposed caching scheme, established in Theorem 3, shows that uncoded caching in the placement phase combined with linear coding in the delivery phase are sufficient to approximately achieve the optimal memory-rate tradeoff.

VI. Connection to Index and Network Coding

The caching problem is related to the index coding problem [10], [11] (or, equivalently [12], the network coding problem [13]). We now explain this connection.

The caching problem introduced in this paper consists of a placement phase and a delivery phase. The most important aspect of this problem is the design of the placement phase in order to facilitate the delivery phase for any possible user demands. Now, for *fixed* content placement and for *fixed* demands, the delivery phase of the caching problem induces a so-called index coding problem. However, it is important to realize that the caching problem actually consists of exponentially many parallel such index coding problems, one for each of the N^K possible user demands. Furthermore, the index coding problem itself is computationally hard to solve even only approximately [14]. The main contribution of this paper is to design the content placement such that each of the many parallel index coding problems has simultaneously an efficient analytical solution. This, in turn, leads to a provably approximately optimal solution for the caching problem.

References

[1] L. W. Dowdy and D. V. Foster, "Comparative models of the file assignment problem," *ACM Comput. Surv.*, vol. 14, pp. 287–313, June 1982.

[2] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE J. Sel. Areas Commun.*, vol. 14, pp. 1110–1122, Aug. 1996.

[3] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *Multimedia Syst.*, vol. 4, pp. 112–121, June 1996.

[4] 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.

[5] A. Meyerson, K. Munagala, and S. Plotkin, "Web caching using access statistics," in *Proc. ACM-SIAM SODA*, pp. 354–363, 2001.

[6] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, pp. 1411–1429, July 2008.

[7] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, pp. 1–9, Mar. 2010.

[8] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *arXiv:1209.5807 [cs.IT]*, Sept. 2012. Submitted to IEEE Trans. Inf. Theory.

[9] M. A. Maddah-Ali and U. Niesen, "Decentralized caching attains order-optimal memory-rate tradeoff," *arXiv:1301.5848 [cs.IT]*, Jan. 2013.

[10] Y. Birk and T. Kol, "Informed-source coding-on-demand (ISCOD) over broadcast channel," in *Proc. IEEE INFOCOM*, pp. 1257–1264, Mar. 1998.

[11] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," in *Proc. IEEE FOCS*, pp. 197–206, Oct. 2006.

[12] M. Effros, S. El Rouayheb, and M. Langberg, "An equivalence between network coding and index coding," *arXiv:1211.6660 [cs.IT]*, Nov. 2012.

[13] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, pp. 1204–1216, Apr. 2000.

[14] M. Langberg and A. Sprintson, "On the hardness of approximating the network coding capacity," *IEEE Trans. Inf. Theory*, vol. 57, pp. 1008–1014, Feb. 2011.