Object Placement for Cooperative Caches with Bandwidth Constraints

UmaMaheswari C. Devi, Malolan Chetlur, and Shivkumar Kalyanaraman

IBM Research – India, Bangalore

Abstract. The projected growth in video traffic delivered to mobile devices is expected to stress the backhaul and core of a broadband wireless network. Caches deployed at the edge elements, such as base stations, are one of alleviating this stress. Limits on the sizes of the base station caches and restrictions on frequent upgrades to the hardware necessitate that techniques that can increase the hit rates with the growing traffic, given the constraints, be explored. In this paper, we consider using cooperative caching schemes for the purpose. The edge elements are connected via bandwidth-constrained links, and hence, the assumption made in most prior work that the cooperating nodes are located on a high-speed network do not apply here. We show that the problem of placing objects to maximize hit rate in such a bandwidth-constrained caching system is NP-hard in the strong sense. We develop an efficient placement algorithm when the caches have identical characteristics and show that its performance is within a constant factor of the optimal under practical conditions. We also discuss how to extend the algorithm for the non-identical case. Our simulation experiments show that in practice, the performance of our algorithm is very close to the optimal and a few tens of cooperating nodes are sufficient to significantly increase the hit rate even with a 1% base cache size.

1 Introduction

Data and Video-on-Demand (VoD) traffic delivered over mobile networks are projected to grow tremendously in the next few years [5]. Current wireless infrastructures are not provisioned to handle this growth. The projected growth is hence expected to significantly increase the stress on not just the wireless channel, but also the wired backhaul and core of a cellular network. Wireless network operators are therefore seeking optimizations that can ease this pressure and help defer infrastructure upgrades.

One simple and effective method to reduce the backhaul traffic is to cache frequently requested content at the edge elements, such as base stations (BS) and central controllers (CC). The limit on the size of a cache that can be placed at a BS is lower than that of traditional Internet caches by an order of magnitude or more. While the smaller caches might be capable of providing good hit rates to traditional web traffic, the same may not hold for VoD and other types of multimedia traffic. This is due to the facts that video objects are much larger in size and the number of video clips is increasing by the day. The latter growth is spurred by the growth in user-generated content and IP and mobile TV, which produce numerous shows per day. In such a scenario, adequate hit rates may be obtained for the growing traffic by increasing the effective cache size by enabling cooperation and sharing of objects among the caching nodes.

E. Jeannot, R. Namyst, and J. Roman (Eds.): Euro-Par 2011, LNCS 6852, Part I, pp. 579–593, 2011.

Cooperative caching has been studied previously for traditional wired networks. However, as discussed below, most of the prior work assumes that the caching nodes are placed on a high-speed network, and hence, that bandwidth available for inter-cache communication is not a constraint.[1] Also, most of the work focuses on minimizing the average object access latency. In contrast, the bandwidth available for inter-BS or inter-CC communication in the wireless edge is limited, and our focus is on reducing the network traffic in the backhaul and core by reducing the byte miss ratio in the edge. Hence, we explore placing objects in a set of cooperating caches such that the hit rate (not access latency) of the caches is optimized subject to not violating the inter-cache communication bandwidth (ICCB) constraints.

Cellular BSs and CCs are organized in a two-level hierarchy with a few 100s of BSs connected to a CC. We assume that any pair of BSs may communicate either directly or via their parent CC. In either case, we assume that the bandwidth available for transferring objects among BSs is limited. In such a setup, we first consider the problem of placing video objects at the BSs (assuming that there is no cache at the CC) and specifying how those objects should be shared for the hit rate is maximized. We show that this problem is NP-hard in the strong sense even when all the caches have identical sizes and see identical object access patterns, an assumption made in several studies. We then develop an efficient object placement algorithm for the case of identical caches and show that it has a constant-factor approximation ratio under conditions expected to hold in practice. Thirdly, we discuss how to extend the algorithm when second-level caches are available at the CCs and relax the assumption that caches are identical. Finally, we evaluate our scheme through simulations.

Our placement algorithm is centralized and the idea is to periodically determine the placement map at a common node such as the parent CC using object popularity information gathered from the BSs. The maps would then be distributed to the BSs, which would subsequently cache the newly specified objects the first time each is requested. To reduce object churn, knowledge of objects already cached at the various nodes may be used while laying out a new map.

Related Work: Cooperative caching was (among other works) first explored as part of the Harvest project [4]. A notable follow up was the Summary Cache [6], which introduced efficient directory services. The above efforts were on the development of techniques and protocols for directing URL requests that miss to sibling or parent nodes and did not deal with bandwidth constraints.

The above work was followed by a good amount of research on object placement algorithms for a cooperating cluster of web caches. Significant works in this area include [11], which provides a 13.93-approximation algorithm for placing objects to minimize the average access cost while imposing no constraints on the available bandwidth, and [13], which extends the algorithm in [11] by including bandwidth constraints. [13] however assumes that the cache size at each node is very large and an object will be stored locally after the initial miss. Hence, unlike our paper, the bandwidth constraints apply only for the initial object placement and for periodic object updates, and not for serving requests to objects from peer nodes on a continuous basis. Also, since video objects are

[1] Works that impose a bandwidth constraint differ in the objective explored or in the exact nature of the bandwidth constraint imposed.

immutable, the need for object updates is obviated, as is the initial object placement as discussed earlier.

Replicating objects in a content-distribution network (CDN) to minimize the average number of ASs traversed [10], heuristics for minimizing the end-user retrieval cost subject to meeting end-user QoS under the assumption of sufficient combined storage for all the objects [14], and placing objects in a CDN layered on a P2P overlay [12] have also been studied. P2P techniques are redundant for BS caches as BSs are reliable entities.

The work that comes closest to ours is [2], which considers placing video objects in a 2-level hierarchical network with costs on bandwidth usage on the links connecting the nodes such that the total bandwidth cost is minimized and proposes distributed algorithms with constant-factor approximation ratios. Bandwidth constraints cannot be converted to bandwidth costs, and hence, the solution of [2] does not extend to the problem in this paper, although there is similarity in the structure of our solution and theirs.

Placing *all* of a large set of video objects in the video hub offices of a large-scale distributed VoD system while minimizing the cost of total byte transfer subject to link bandwidth limits is considered in [1]. The problem differs from ours since we also need to determine *which* of the objects need to be placed. Another problem with a similar flavor, that of distributing chunks of videos among end clients with local connectivity to reduce the stress on the access networks is considered in [9].

The rest of the paper is organized as follows. Our system model is described in Sec. 2. Sec. 3 develops a placement algorithm, derives an approximation ratio for it, and discusses extensions to the algorithm. Simulation studies are described in Sec. 4. Sec. 5 concludes.

2 System Model and Problem Formulation

We consider a cooperative caching proxy system of N nodes, $1, \ldots, N$, where node j is provided with a cache of size C_j. Each request to each of a set of M video objects, where the size of object i is S_i bytes, pass through one of the N nodes. If a requested object is cached at the node that receives the request, it is served from the node's local cache. The nodes are assumed to be provided with dedicated bandwidths, referred to as *inter-cache communication bandwidth*, (ICCB), both in the upload and download directions, that may be used for letting an object cached at a node to be borrowed by a peer node. Hence, a request that misses at a node's local cache can be served from either a peer node or the origin server. The upload and download bandwidth limits at node j are denoted B_j^u and B_j^d, respectively. The average demand in requests per second for object i at node j is denoted R_{ij}; its bandwidth is hence $R_{ij} \cdot S_i$ bytes/sec. The total bandwidth of objects borrowed by/from a node cannot exceed its ICCB limits. We are concerned with placing a subset of M objects at the N nodes and designating how objects are shared such that the total bytes served per second from the caches (byte hit rate) is maximized. Note that since nodes have dedicated ICCB, an object served by borrowing from a peer cache is considered a hit. We refer to the set of all caches at the N nodes as the *combined* or *collective* cache.

Let x_{ij} be a 0-1 integer variable denoting whether object i is placed at cache j. Similarly, let x_{ijk} denote whether object i placed in cache j is borrowed by cache k. The problem of placing objects at the caches and determining how objects are shared to maximize the byte hit rate, denoted O_Place_Gen, can then be formulated as shown in the inset to the right.

O_Place_Gen

$$\text{Maximize} \sum_{i=1}^{M}\sum_{j=1}^{N}(x_{ij}\cdot R_{ij}\cdot S_i + \sum_{k=1}^{N} x_{ikj}\cdot R_{ij}\cdot S_i)$$

$$\text{subject to} \sum_{i=1}^{M} S_i \cdot x_{ij} \le C_j, \quad j = 1,\ldots,N \tag{1}$$

$$x_{ik} + \sum_{j=1}^{N} x_{ijk} \le 1,\ i = 1,\ldots,M,\ k = 1,\cdots,N \tag{2}$$

$$x_{ijk} \le x_{ij}, \quad i = 1,\ldots,M,\ j,k = 1,\ldots,N \tag{3}$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N} x_{ijk}\cdot R_{ik}\cdot S_i \le B_k^u, \quad k = 1,\ldots,M \tag{4}$$

$$\sum_{i=1}^{M}\sum_{k=1}^{N} x_{ijk}\cdot R_{ik}\cdot S_i \le B_j^d, \quad j = 1,\ldots,M \tag{5}$$

$$x_{ij} \in \{0,1\}, x_{ijk} \in \{0,1\}, \quad i = 1,\ldots,M, \quad j,k = 1,\ldots,N \tag{6}$$

Object i served from node j, either using a copy locally cached at it or borrowed from another node k, would lead to $R_{ij}\cdot S_i$ fewer bytes per second requested from the hosting servers and transported over the core and backhaul networks. The objective function is therefore as indicated. The constraints in (1) account for the limits on the cache sizes. Constraint (2) prevents a node from both caching a node locally as well as borrowing from one or more caches, while (3) ensures that node k borrows an object i from node j only if i is cached at j. (4) and (5) ensure that the limits on uplink and downlink bandwidths available for inter-cache transport are not violated at any node.

In O_Place_Gen, the objective function and all the constraints are linear in the decision variables, so it is an integer linear program. Solving it with generic integer program methods can therefore require exponential time. It turns out that even a simpler special case of the problem with uniform object and cache sizes, denoted S and C, respectively, identical uplink and downlink bandwidth limits, denoted B, and identical popularity distributions at all nodes, denoted R_i for object i (the bandwidth for object i would be $R_i \cdot S$ bytes per sec at all nodes), is actually NP-hard in the strong sense, so an exact solution to it or the general problem cannot be obtained in polynomial time using alternative methods either, unless P=NP. The special case, denoted O_Place_Spl, is obtained from O_Place_Gen by replacing R_{ij}'s and R_{ik}'s with R_i, S_i's with S, C_j with C, and B_k^u and B_j^d with B. A complete problem statement is omitted due to space constraints.

3 Hardness Result and Approximation Algorithm

The special case of the object placement problem O_Place_Spl is NP-hard in the strong sense as we show in the longer version of this paper. The reduction is from the 3-PARTITION (3-PART) problem, which is NP-complete in the strong sense. Hence, a pseudo-polynomial-time algorithm or an FPTAS are also not possible for O_Place_Spl, apart from a polynomial-time algorithm.

3.1 Hardness Proof

3-PART is a number problem [7, pp. 224 and 94] defined as follows.

Definition 1 (3-PART)**:** Given set E of $3m$ elements, $e_1, e_2, \ldots, e_{3m}$, a bound $K \in \mathbb{Z}_+$, and a size $s(e_i) = s_i \in \mathbb{Z}_+$ for each $e_i \in E$ such that $K/4 < s_i < K/2$ and $\sum_{i=1}^{3m} s_i = mK$. The problem is to determine whether E can be partitioned into m disjoint sets $E_1, E_2, \ldots, E_m$ such that $\sum_{e \in E_i} s(e) = K$, for $1 \leq i \leq m$.

Theorem 1. O_Place_Spl *is NP-hard in the strong sense.*

Proof: To prove the theorem, we show that the decision version of O_Place_Spl is NP-complete in the strong sense. It is easy to see that the decision version is in NP. We provide a pseudo-polynomial reduction [7] from 3-PART to it.

Consider an arbitrary instance of 3-PART and construct an instance of O_Place_Spl, denoted objpl-3-part, from it, as follows. Let $N = m$, $M = 3m$, $C = 3 \cdot S$, $S = 1$, and $B = (m-1) \cdot K$. Let $R_i = s_i/S$ for $1 \leq i \leq N$. Let $B_i = R_i \cdot S = s_i$ denote the bandwidth of object i. We now show that a solution to 3-PART exists if and only if there is a solution to objpl-3-part with objective value exactly equal to $N \cdot m \cdot K = m^2K$.

$\Leftarrow$ Assume that there is a solution to objpl-3-part with objective value exactly m^2K.

Because $C = 3 \cdot S$, each node can store at most three objects locally in its cache. We first show that this number is exactly three. If some node stores fewer than three objects, then the total number of objects stored in the combined cache is less than $3m$. Thus, there exists at least one object that is not served by the combined cache, and hence, the objective value of the solution to objpl-3-part cannot equal or exceed m^2K, which contradicts our assumption. Thus, each node stores exactly three objects in its cache. Next, we show that for each node, the total bandwidth of the three objects stored in its cache is exactly K. For this, first note that for the objective value to equal m^2K, each of the m nodes should serve all the $3m$ objects from the combined cache. The total bandwidth of all the objects is mK. The total bandwidth of the objects that a node borrows from other caches cannot exceed $(m-1) \cdot K$ (since the downlink bandwidth at each node (B) is limited to $(m-1) \cdot K$). Hence, each node should serve the remaining $mK - (m-1)K = K$ bytes from its local cache. Thus, the total bandwidth of the three objects that each node caches is exactly K. Therefore, since $B_i = s_i$, a solution to 3-PART can be obtained from a solution to objpl-3-part by assigning element e_i to set E_j if object i is assigned to node j (that is $x_{ij} = 1$).

$\Rightarrow$ A solution to objpl-3-part with objective value exactly m^2K that satisfies cache capacity constraints and bandwidth constraints can be obtained by simply setting $x_{ij} = 1$ if e_i is assigned to set E_j, and $x_{ijk} = 1$ for all $k \neq j$, if $x_{ij} = 1$.

The reduction can be performed in polynomial time. All numbers in objpl-3-part are polynomially bounded by the numbers in 3-PART. Thus, the decision version of O_Place_Spl is NP-complete in the strong sense. O_Place_Spl is hence NP-hard in the strong sense. ■

3.2 Efficient Placement Algorithm

In this section, we focus on designing an efficient centralized algorithm for solving O_Place_Spl. We assume that at least a few of the top K most popular objects have

bandwidth at most $B/(N-1)$. ($K = C/S$, the number of objects that fit in a cache.) Otherwise, the scope for cooperation would be very limited and one may consider enabling cooperation among fewer caches (*i.e.*, with smaller N).

Identifying Heuristics. Before presenting an algorithm for O_Place_Spl, we present some rules of thumb that have been used in its design. In what follows, we will refer to an object that is cached at all N nodes as *fully replicated*. An object that is cached at two or more nodes, but not all nodes, is said to be *partially replicated*, and one that is cached at a single node as *unreplicated*. An unreplicated object that is shared by all nodes is said to be *totally shared*, while an unreplicated or a partially-replicated object that is shared by some but not all nodes or borrowed partly by all nodes is referred to as *partially shared*. An object cached at a node is said to be partly borrowed by a peer node if part of the requests to the object at the peer node that are evenly distributed is served from the caching node.

Rule 1. *Cache objects with the largest bandwidths* (*in fully-replicated, partially-replicated, or unreplicated manner*).

This rule is quite obvious and is used by most caching systems.

Rule 2. *Since ICCB is constrained, replicate, either fully or paritially, objects of larger bandwidths.*

If ICCB is abundant, then bytes served from the collective cache can be maximized by having unique copies of as many objects as possible in the constituent caches and serving those objects from the combined cache at every node. In such a case, most objects (if not all) are unreplicated and the rule does not apply.

If ICCB is limited, then it can be shown that fully replicating one or more objects will serve to increase the combined hit rate. To see that replicating higher bandwidth objects is beneficial, suppose a higher bandwidth object, H, is unreplicated or partially replicated, while a lower bandwidth object, L, is fully replicated. Then, a node $\mathcal{N}$ that does not cache H has two options: it either fetches H from the hosting server or borrows it from a peer. It is easy to see that simply replacing L by H would in the former case increase the number of bytes served locally from $\mathcal{N}$ (while not decreasing the number of bytes borrowed from the other caches and served). In the latter case, the replacement would lead to H being served locally. The downlink bandwidth that consequently gets freed up at $\mathcal{N}$ can be used to borrow at least L, and potentially, a few more objects. So, the total bytes that $\mathcal{N}$ serves from the combined cache is not lowered. Thus, replicating larger bandwidth objects serves, in general, to increase the bytes served.

Rule 3. *Among the objects chosen for caching, unreplicate and totally share those with lower bandwidths, subject to not violating the ICCB constraints.*

The rationale for this rule is similar to that for the prior one.

Object-Placement Algorithm. Let $\mathcal{O}$ denote the set of all M objects arranged in non-increasing order of their bandwidths. We start with the set of objects with the highest demand (that is, the largest bandwidth objects), referred to as $\mathcal{O}_C$, that will fit in a cache of size C. (These will be the objects that each node caches in a non-cooperative setting.) These would form the initial set of replicated objects, while the initial shared object set is $\emptyset$.

Let $\bar{\mathcal{O}}_C$ denote the set of objects in $\mathcal{O}$ excluding those in $\mathcal{O}_C$. Since ICCB B is constrained, not all objects can be unreplicated and shared, and by Rule 2, high bandwidth objects should be replicated. Our goal is to identify the boundary at which unreplication and sharing should commence.

Given that ICCB is B, the amount of data that each node serves from the combined cache can be at most B bytes per second higher than the total bandwidth of the objects in $\mathcal{O}_C$. Let $\mathcal{O}_{inc} \subseteq \bar{\mathcal{O}}_C$ denote the set of objects brought into the combined cache when cooperation is enabled. By Rule 3, as many of these objects should be totally shared. If all the objects in $\mathcal{O}_{inc}$ could be totally shared, then the total bandwidth of all the objects in $\mathcal{O}_{inc}$ could be at most B. Furthermore, for every $N-1$ objects brought into

1: b : `array` $1..M$ `of real sorted descending` {object bandwidths}
2: shr_from : `integer` {starting index of unreplicated and shared objects in $\mathcal{O}_C$}
3: L : `integer` {no. of objects in $\mathcal{O}_C$ that are unreplicated and shared}
4: B_{top} : `real` {total bandwidth of unreplicated and shared objects in $\mathcal{O}_C$}
5: B_{inc} : `real` {total bandwidth of unreplicated and shared objects in $\bar{\mathcal{O}}_C$}
6: {Determine the objects with total bandwidth at most $B/(N-1)$ at the tail of the objects in $\mathcal{O}_C$}
7: $i := K$; {K is the no. of objects that can be held in a cache}
8: $B_{top} := 0$;
9: **while** $B_{top} + b[i] \leq B/(N-1)$ **do**
10: $B_{top} := B_{top} + b[i]$;
11: $i := i - 1$;
12: **end while**
13: $shr\_from := i + 1$;
14: $L := K - shr\_from + 1$;
15: /* Select $(N-1) \cdot L$ objects from $\bar{\mathcal{O}}_C$*/
16: $B_{inc} := \sum_{i=K+1}^{K+(N-1)\cdot L} b[i]$
17: $\mathcal{O}_{shared}$:= objects $K - L + 1 \ldots K + (N-1) \cdot L$;
18: **while** $NB/(N-1) > B_{inc} + B_{top}$ **do**
19: /* Check if unreplicating and sharing the next lowest bandwidth object from $\mathcal{O}_C$ can increase the total bandwidth of objects from $\bar{\mathcal{O}}_C$ brought into the combined cache and shared */
20:
21: **if** $b[shr\_from - 1] + \sum_{i=K+(N-1)(L+1)}^{K+(N-1)(L+1)} b[i] + B_{inc} + B_{top} \leq NB/(N-1)$ **then**
22: $shr\_from := shr\_from - 1$; $L := L + 1$;
23: $B_{top} := B_{top} + b[shr\_from]$;
24: $B_{inc} := B_{inc} + \sum_{i=K+(N-1)(L+1)}^{K+(N-1)(L+1)} b[i]$;
25: Update $\mathcal{O}_{shared}$ to include the newly selected objects;
26: **end if**
27: **end while**
28: /* Distribute the objects in $\mathcal{O}_{shared}$ using a balanced fit heuristic such that the total bandwidth of all objects assigned to any node is at most $B/(N-1)$ */
29: **for** each object with index O in $\mathcal{O}_{shared}$ **do**
30: /* consider objects in non-increasing order of their bandwidths */
31: assign O to the node with the largest unused ICCB among those with spare physical slots if such a node exists and unused ICCB at the node is at least $(N-1) \cdot b[O]$;
32: mark O as totally shared from the node caching it;
33: **end for**
34: assign the remaining objects in $\mathcal{O}_{shared}$ to nodes with available physical slots; mark them unshared

Fig. 1. Object Placement Algorithm PA

the combined cache, due to cache capacity constraints, at least one object from $\mathcal{O}_C$ should be unreplicated and shared (to make room for the incoming objects). Thus, if $\ell = |\mathcal{O}_{inc}|$, $L = \lceil \ell/(N-1) \rceil$ of the $\mathcal{O}_C$ objects should be unreplicated and shared.

By the ICCB constraint B, at each node, the total bandwidth of all the objects that are unreplicated and totally shared cannot exceed $B/(N-1)$. Thus, the total bandwidth of all the unreplicated and totally shared objects in the combined cache cannot exceed $NB/(N-1)$. Our objective of maximizing the total bytes served from the combined cache thus reduces to the following:

Phase 1. Choosing L objects from $\mathcal{O}_C$ and ℓ objects from $\bar{\mathcal{O}}_C$ for sharing such that the total bandwidth of the objects from $\bar{\mathcal{O}}_C$ is maximized and the constraints below hold.

(C1) $\ell \leq (N-1) \cdot L$

(C2) The total bandwidth of the $L + \ell$ objects chosen is at most $NB/(N-1)$

Phase 2. Partitioning the $L + \ell$ chosen objects among the N nodes such that the total bandwidth of the objects assigned to each node is at most $B/(N-1)$.

Listing for an algorithm, denoted PA, that accomplishes the above is provided in Fig. 1. Choosing objects that should be unreplicated and totally shared is performed in the first phase in lines 7–27. In this phase, L is initially set to the number of the lowest bandwidth objects in $\mathcal{O}_C$ with total bandwidth at most $B/(N-1)$ (lines 7–17). If the combined bandwidth of the first $(N-1) \cdot L$ objects from $\bar{\mathcal{O}}_C$, B_{inc}, and the L objects from $\mathcal{O}_C$, B_{top}, is at least $NB/(N-1)$, then the algorithm moves to the second phase. Since objects are arranged in non-increasing order of bandwidths, $B_{inc} \leq (N-1) \cdot B_{top}$ holds at every step.

On the other hand, if the combined bandwidth is less than $NB/(N-1)$, then the **while** loop in line 18 is entered. L is incremented by one and ℓ by $N-1$ as long as the combined bandwidth of the chosen objects remains less than $NB/(N-1)$. The first phase ends when no more objects can be brought in from $\bar{\mathcal{O}}_C$. At its end, NL objects are marked for sharing.

In the second phase, the objects chosen for sharing are partitioned among the nodes. In the first step of this phase in lines 29–32, objects are distributed such that the total bandwidth of all the objects assigned and totally shared from a node is at most $B/(N-1)$. Since distributing objects without violating the ICCB constraint is a bin packing problem, for which feasible solutions are known to *not exist* for all instances, not all objects can be expected to be successfully assigned. The remaining objects are filled in the available slots of all the caches in the next step in line 34. Exactly $K - L$ objects are fully replicated while no object is partially replicated. Hence, each cache can hold exactly L more objects for a total of NL objects in all the caches. Thus, since the total number of objects chosen for distribution is NL, all objects will be successfully assigned to some cache but not all may necessarily be shared.

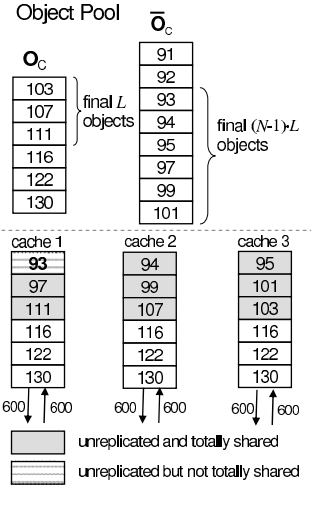

Fig. 2. Placement example

Example. To better understand the algorithm, consider the example in Fig. 2. Here $N = 3$ and $C = 6S$ so that $K = 6$. $M = 14$ objects, and their bandwidths are indicated in the boxes. The objects in $\mathcal{O}_\mathsf{C}$ and $\bar{\mathcal{O}}_\mathsf{C}$ are as indicated. ICCB B is 600. Since $B = 600$, in the first phase, $L = 3$ objects and $\ell = 2L = 6$ objects as marked in the figure could be selected for sharing from $\mathcal{O}_\mathsf{C}$ and $\bar{\mathcal{O}}_\mathsf{C}$, respectively. B_{top} is 321, B_{inc} is 579, and $B_{top} + B_{inc} = 900 = NB/(N-1)$.

In the second phase, eight of the objects selected in the first phase could be distributed among the three caches using the heuristic in lines 29–32 such that ICCB is respected. The final object (with lowest bandwidth) is assigned to the first cache but is marked unshared as otherwise ICCB would be violated. It can be verified that the objects cannot be partitioned among the caches such that the constraints are satisfied.

Byte hit rate after cooperation increases by 579 Bps for Cache 1 and 486 Bps for each of the other two nodes. Hit rates for the latter two can be increased by $93/2 = 46.5$ Bps by serving half their requests to the ninth object from Cache 1.

Algorithm complexity: If the objects are sorted by their bandwidths, the complexity of the algorithm can easily be seen to be $O(NK)$. Otherwise, it is $O(NK + M \log M)$, which is $O(M(K + \log M))$.

3.3 Approximation Ratio

We now derive an approximation ratio for Algorithm PA, assuming Zipf-like distribution [3] or its generalization, the MZipf distribution [8], for object popularity distributions. The approximation ratio would hold as long as the ratio of the probability of accesses of objects with ranks i and $i+1$ is at most $\frac{i+1}{i}$.

PA chooses L and $(N-1)L$ contiguous objects from the tail and head of $\mathcal{O}_\mathsf{C}$ and $\bar{\mathcal{O}}_\mathsf{C}$, respectively, such that their total bandwidth is maximized subject to not exceeding $NB/(N-1)$. Let the two subsets be denoted $\mathcal{O}_\mathsf{C}^\mathsf{PA}$ and $\bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA}$, respectively, and let $\mathcal{BW}(.)$ denote the bandwidth function. The increase in hit rate achieved by PA per node is therefore at most $\mathcal{BW}(\bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA})$. (It would be less than $\mathcal{BW}(\bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA})$ if the objects $\mathcal{O}_\mathsf{C}^\mathsf{PA}$ and $\bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA}$ cannot be partitioned among the nodes.) The hit rate per node obtained by an optimal algorithm may be higher by less than the bandwidth corresponding to the next $N-1$ objects from $\bar{\mathcal{O}}_\mathsf{C}$. To see this note that since PA could not choose the next $N-1$ objects, the total bandwidth of those objects and an additional lightest object from $\mathcal{O}_\mathsf{C}$ along with objects in $\mathcal{O}_\mathsf{C}^\mathsf{PA}$ and $\bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA}$ exceeds $NB/(N-1)$. Hence, choosing any other object from $\mathcal{O}_\mathsf{C}$ would not enable choosing $N-1$ objects from $\bar{\mathcal{O}}_\mathsf{C}$ with larger bandwidth than the next $N-1$.

Let the maximum bandwidth $\mathcal{BW}_{\max}$ of any object in $\mathcal{O}_\mathsf{C}^\mathsf{PA} \cup \bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA}$ be at most $f \cdot \frac{B}{N-1}$, where $0 < f < 1$, and let $R = \lfloor \frac{1}{f} \rfloor$. In general, for $\frac{1}{n+1} < f \leq \frac{1}{n}$, $R = n$ holds. Also, $R = \lfloor \frac{1}{f} \rfloor \Rightarrow R \leq \frac{1}{f}$, and hence,

$$f \leq \frac{1}{R}. \tag{7}$$

To determine an approximation ratio, we need to determine a lower bound on the increase to hit rate achieved by PA. As discussed above, it would be less than $\mathcal{BW}(\bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA})$ if objects in $\mathcal{O}_\mathsf{C}^\mathsf{PA} \cup \bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA}$ are not all fully shared. If β denotes the bandwidth of objects in $\mathcal{O}_\mathsf{C}^\mathsf{PA} \cup \bar{\mathcal{O}}_\mathsf{C}^\mathsf{PA}$ that are not fully shared, then the increase in hit rate achieved by PA is

given by $\mathcal{BW}(\bar{\mathcal{O}}_{\mathcal{C}}^{\mathsf{PA}}) - \beta$. The following lemma provides a lower bound on the sharing achieved by PA.

Lemma 1. *The total bandwidth of the objects that are unreplicated and fully shared at the end of Phase 2 of* PA *is at least* $\frac{R}{R+1}(\mathcal{BW}(\mathcal{O}_{\mathcal{C}}^{\mathsf{PA}}) + \mathcal{BW}(\bar{\mathcal{O}}_{\mathcal{C}}^{\mathsf{PA}}))$.

Proof: The total number of objects chosen for sharing is NL. These objects may be assigned to one of the N nodes and the maximum number of objects assigned to a node cannot exceed L. An object assigned to a node may be fully shared if the total bandwidth of all the objects assigned previously to the same node and the new object is at most $B/(N-1)$.

During the partition phase of PA, let Ω be the first object that could not be assigned in a fully-shared manner, and let w denote its bandwidth. Then for each node $\mathcal{C}$, one of following conditions hold: **(1)** The total assigned bandwth in $\mathcal{C}$ is at least $(B/(N-1) - w)$. **(2)** $\mathcal{C}$ is full with L assigned objects (has no empty slots), but the total bandwidth of the objects assigned to it is less than $B/(N-1)$.

Let n of the N nodes be of type 1, with condition 1 holding, and the remaining $N-n$, of type 2. Since objects are assigned in monotonically decreasing order of their bandwidths, the bandwidth of every object assigned to a node of type 2 is at least w. Then, the total bandwidth $\mathcal{B}$, of all the objects before Ω assigned to the N nodes is at least $n(\frac{B}{N-1} - w) + (N-n)wL$. By (7) and the definition of f, the bandwidth of any object is at most $B/(R(N-1))$. Hence, if $L \leq R$, the total bandwidth of any subset of L objects is at most $B/(N-1)$. Thus, the NL objects in $\mathcal{O}_{\mathcal{C}}^{\mathsf{PA}} \cup \bar{\mathcal{O}}_{\mathcal{C}}^{\mathsf{PA}}$ can be partitioned among the N nodes such that each is fully shared. Hence, for the rest of the proof take $L > R$. Since the total bandwidth expression is an increasing function of L, $\mathcal{B}$ is at least $\frac{nB}{N-1} + ((R+1)N - (R+2)n)w$. We consider the following cases.

Case 1: $n \leq \frac{1}{R+1}N$**.** In this case at least $R/(R+1)$ of the nodes are of type 2, which are full. Thus, at least a fraction $R/(R+1)$ of the objects are fully shared. Since the objects are assigned in the order of decreasing bandwidths, the total shared bandwidth is at least a fraction $R/(R+1)$ of the bandwidth of the objects in $\mathcal{O}_{\mathcal{C}}^{\mathsf{PA}}$ and $\bar{\mathcal{O}}_{\mathcal{C}}^{\mathsf{PA}}$.

Case 2: $\frac{1}{R+1}N < n \leq \frac{R}{R+1}N$**.** The proof for this case is a little involved and hence omitted due to space constraints. It will be made available in a longer version of the paper.

Case 3: $\frac{R}{R+1}N < n \leq \frac{R+1}{R+2}N$**.** Since $(R+1)N - (R+2)n \geq 0$ holds and $\mathcal{B} \geq \frac{nB}{N-1} + ((R+1)N - (R+2)n)w$, $\mathcal{B} \geq \frac{R}{R+1}NB(N-1)$.

Case 4: $n > \frac{R+1}{R+2}N$**.** In this case, $(R+1)N - (R+2)n < 0$, and hence, $\frac{nB}{N-1} + ((R+1)N - (R+2)n)w$ is a decreasing function of w. If $w > \frac{1}{R+1}\frac{B}{N-1}$, then since objects are assigned in decreasing bandwidth order and each of the N nodes has at least R objects assigned (because as discussed above $R < L$ and the bandwidth of any object is at most $\frac{1}{R}\frac{B}{N-1}$), $\mathcal{B} \geq R \cdot N \cdot \frac{1}{R+1}\frac{B}{N-1} \geq \frac{R}{R+1}(\mathcal{BW}(\mathcal{O}_{\mathcal{C}}^{\mathsf{PA}}) + \mathcal{BW}(\bar{\mathcal{O}}_{\mathcal{C}}^{\mathsf{PA}}))$. If not $\mathcal{B}$ is minimized for $w = \frac{1}{R+1}\frac{B}{N-1}$ and hence $\mathcal{B} \geq \frac{nB}{N-1} + ((R+1)N - (R+2)n)\frac{1}{R+1}\frac{B}{N-1}$, which simplifies to $\frac{NB}{N-1} - \frac{n}{R+1}\frac{B}{N-1}$. This expression is minimized at $n = N$, yielding $\mathcal{B} \geq \frac{R}{R+1}\frac{NB}{N-1}$. ■

Thus, the bandwidth of objects not fully shared is at most $\frac{1}{R+1}(\mathcal{BW}(\mathcal{O}_C^{\mathsf{PA}}) + \mathcal{BW}(\bar{\mathcal{O}}_C^{\mathsf{PA}}))$ and the increase in hit rate is at least $\frac{R}{R+1}\mathcal{BW}(\bar{\mathcal{O}}_C^{\mathsf{PA}}) - \frac{1}{R+1}\mathcal{BW}(\mathcal{O}_C^{\mathsf{PA}})$.

The lemma below provides a lower bound on $\mathcal{BW}(\bar{\mathcal{O}}_C^{\mathsf{PA}})$. Recall that $K = C/S$ denotes the number of objects that fit in a cache.

Lemma 2. *If the object popularity distribution follows Zipf-like or MZipf, then* $\mathcal{BW}(\bar{\mathcal{O}}_C^{\mathsf{PA}}) \geq \frac{\ln \frac{K+(N-1)L}{K+1}}{\ln \frac{K+1}{\max(K-L,1)}} \times \mathcal{BW}(\mathcal{O}_C^{\mathsf{PA}})$.

Proof: Follows from the facts that the probability of accessing an object at rank i is proportional to $1/i$ for the Zipf object popularity distribution and lower than $1/i$ for Zipf-like and MZipf distributions, and $\sum_{i=1}^{n_1} \frac{1}{i} - \ln(n_1) \geq \sum_{i=1}^{n_2} \frac{1}{i} - \ln(n_2 + 1)$, for all $n_1, n_2 \geq 1$. ■

We are now ready to provide an approximation ratio. Let $\alpha = L/K$.

Theorem 2. *The approximation ratio of* PA *when object popularities conform to the Zipf, Zipf-like or MZipf distributions is given by* $\frac{L+1}{L}\frac{(R+1)\ln(\frac{K+(N-1)L}{K+1})}{R\ln(\frac{(K+(N-1)L)}{K+1}-\ln(\frac{K+1}{\max K-L,1})}$, *which is,* $\frac{L+1}{L}\frac{(R+1)\ln(\frac{1+(N-1)\alpha}{1+\frac{1}{K}})}{R\ln(\frac{1+(N-1)\alpha}{1+\frac{1}{K}})+\ln(\frac{\max(1-\alpha,\frac{1}{K})}{1+\frac{1}{K}})}$, *which is* $\frac{L+1}{L}\frac{(R+1)\ln(1+(N-1)\alpha)}{R\ln(1+(N-1)\alpha)+\ln(\max(1-\alpha,\epsilon))}+\delta$.

Proof: As discussed earlier, by Lemma 1, the effective increase to hit rate achieved by Algorithm PA is at least $(R \cdot \mathcal{BW}(\bar{\mathcal{O}}_C^{\mathsf{PA}}) - \mathcal{BW}(\mathcal{O}_C^{\mathsf{PA}}))/(R+1)$. Letting $\mathcal{BW}(\bar{\mathcal{O}}_C^{\mathsf{PA}}) = \kappa \cdot \mathcal{BW}(\mathcal{O}_C^{\mathsf{PA}})$, the increase to hit rate achieved by PA is at least $(R\kappa - 1) \cdot \mathcal{BW}(\mathcal{O}_C^{\mathsf{PA}})/(R+1)$.

To determine the approximation ratio, we need to determine a bound on the increase in hit rate achieved by an optimal algorithm. As discussed at the beginning of this subsection, this value is at most the bandwidth of an additional $N-1$ objects from $\bar{\mathcal{O}}_C$. Since objects are arranged in decreasing order of bandwidths, the total bandwidth of these additional objects would be at most the bandwidth of any $N-1$ objects in $\bar{\mathcal{O}}_C^{\mathsf{PA}}$. Since $|\bar{\mathcal{O}}_C^{\mathsf{PA}}| = (N-1)L$, the increase in hit rate achieved by an optimal algorithm would be at most $\frac{(N-1)(L+1)}{(N-1)L}$ times $\mathcal{BW}(\bar{\mathcal{O}}_C^{\mathsf{PA}}) = \frac{(L+1)}{L} \times \kappa \cdot \mathcal{BW}(\mathcal{O}_C^{\mathsf{PA}})$. The approximation ratio, given by the ratio of the optimal increase in hit rate to the effective increase achieved by PA is hence $\frac{(R+1)\kappa}{R\kappa-1} \cdot \frac{L+1}{L}$. Using the lower bound provided by Lemma 2 for κ and simplifying, we obtain the approximation ratios in the lemma. ■

Discussion: The approximation ratio ρ in Thm. 2 decreases with increasing N and R. For a given N and R, ρ initially decreases with α and then increases. ρ is valid for all values except when N, R, and L are all at most 3, in which case, cooperation is of very little or no use anyway. When K is at least 1000, for $\alpha \leq 0.1$, that is, when at most 10% of the objects in cache are chosen for sharing, $\rho < 2.96$ for $N \geq 5$, for all R, and $\rho < 2.7$ for $R \geq 3$, for all N. In general, ρ is small if α is not large (≤ 0.7) or one of R and N is not too small, with values at least 3 and 5, respectively, which are very reasonable. Recall that the achievable increase in hit rate depends on the ICCB B and for an appreciable increase, say $x\%$, B should be at least $x\%$ of the total bandwidth of all the objects. For the Zipf distribution, the bandwidth of the object with rank n is $1/(n \cdot ln(K))$ of the total object bandwidth. Hence, for $x \geq 10$, the bandwidths of objects with rank 10 and higher is less than $1/90^{\text{th}}$ the total bandwidth for a modest

$M = 10000$. Thus, $R \geq 9$, and in practice can be expected to be much higher. For $R \geq 10$, $\rho \leq 1.76$ and $\rho \leq 3.48$ for α as high as 0.99 and 0.9, respectively, for all N. The approximation ratio of the placement algorithm PA can thus be taken to be a small constant for all practical purposes.

3.4 Extensions to Algorithm PA

Hierarchical Caching: Suppose a second-level parent cache of K' objects is provided in the path of the object requests, *e.g.*, at the CC in the wireless backhaul. In the absence of cooperation among child nodes, the most popular K objects will be replicated at the children, while the next K' popular objects would be placed at the parent. Which objects to unreplicate and share at the children when cooperation is enabled would depend on the limit on the amount by which the content served from the parent node may be increased. If this traffic need not be limited as long as the total hit rate increases, then an object placement may be obtained by assuming a cache of $K + K'$ objects at each child and applying algorithm PA, but restricting L to at most K. The $K - L$ most popular objects should then be replicated at all the child nodes, the next K' objects placed at the parent, and the next NL objects placed in one of the children as specified by PA. The reason for considering $K + K'$ objects as opposed to K during placement is to reduce the mean bandwidth of the objects that are shared, and thereby improve the efficiency of partitioning them in a fully shared manner without violating ICCB. If there is a restriction on the traffic that may be increased on the parent to child links, the restriction should be used to determine the cache size that PA should assume to determine a placement.

Relaxing the assumptions: The most restrictive of the similarity characteristics assumed for the caches and objects is that of identical sizes for all the objects. Identical object popularity distributions can be expected in a cluster of a few tens of BSs, which as discussed in Sec. 4, are sufficient in practice to achieve close to ideal hit rates. This is because at least a couple of thousand BSs are typically deployed in a mid-size city and hence 20-30 BSs can be expected to cover just a fraction of a city with somewhat homogeneous object access patterns. Homogeneity would also be enhanced by the larger expected mobility within a smaller region. The assumption of identical cache sizes may be expected to hold for the same reason that the number of BSs needed for good hit rates can be found within a small geography. If the assumption does not hold, it may easily be overcome by running PA with the smallest of the cache sizes. The additional capacity in the larger caches may simply be used for storing additional objects beyond those specified by PA for higher hit rates at the larger caches. Handling non-identical object sizes is discussed below.

Non-Identical Object Sizes: If object sizes are not identical, then since it is the bandwidth per unit size that matters, objects should be ordered by their popularity instead of bandwidths. Next, instead of choosing $N - 1$ objects from $\bar{\mathcal{O}}_C$ for every object chosen from $\mathcal{O}_C$, Algorithm PA should be modified to choose the maximum number of objects whose combined size does not exceed $N - 1$ times the size of an object chosen from $\mathcal{O}_C$. While non-homogeneity in object sizes can lead to inefficiencies in object selection and distribution, they can be expected to be minimal when the object pool is large.

4 Empirical Evaluation

In this section, we present the results of simulations conducted to evaluate the performance of our placement algorithm. We conducted experiments for varying values of total objects M, nodes N, and cache size C. Object size S was set to 1GB. Taking the total bandwidth served by BSs into account, the total object bandwidth was set to 20 Mbps, yielding a request rate of 0.0025/sec. ICCB B was set to 5 Mbps, for a maximum achievable increase of 25% to the hit rate. The M-Zipf distribution [8], in which the probability of accessing object of rank i is proportional to $1/(i+q)^{\gamma}$, was used for object popularities, with $q = 50$ $\gamma = 0.75$.

Representative results with $M = 20,000$, that is a total corpus size of 20 TB, and varying cache sizes as indicated are plotted in Fig. 3. Inset (a) plots the increase in hit rate as a % of the total object bandwidth of 20 Mbps. Since B is 25% of the total object bandwidth, the maximum achievable increase to the hit rate is 25%. The hit rate increase is rather low for small values of N. We also determined an upper bound to the optimal achievable increase for all the cases. The plots of the optimal increase almost coincide with the observed increase and hence have been omitted. The low hit rates for small N are therefore not due to the partitioning inefficiency of PA. This is because for $M = 20,000$, the bandwidth due to the most popular object is roughly 0.001% of the total bandwidth, hence R is quite large, easily exceeding 200. The approximation ratio as given by Thm. 2 is therefore close to 1. The low hit rates for small N are rather due to the larger values for $B/(N-1)$, and hence a large L, as indicated by the plots of $\alpha = L/K$ in Fig. 3(b). For large L, the ratio of the mean bandwidth of objects in $\mathcal{O}_C^{PA}$ and $\bar{\mathcal{O}}_C^{PA}$ is high. Since the total bandwidth of the two subsets is constrained to be at most $NB/(N-1)$, $\mathcal{BW}(\bar{\mathcal{O}}_C^{PA})$, as a fraction of the total bandwidth, is low. Recall that the optimal increase in hit rate only slightly exceeds $\mathcal{BW}(\bar{\mathcal{O}}_C^{PA})$, and hence when N is small, the increase, both optimal and observed, are low. The hit rate

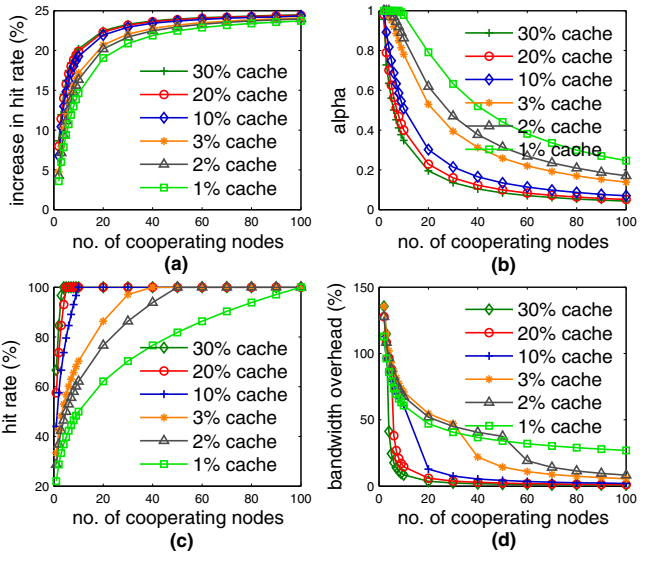

Fig. 3. Performance evaluation results for the placement algorithm PA. **(a)** Additional hit rates for varying % of cache sizes. **(b)** Values of $\alpha = L/K$ for the runs in (a). **(c)** Hit rates with varying N when ICCB is not a limitation. **(d)** Bandwidth overhead for the runs in (c). The legend entries are in the curve order in insets (a) and (c) and reverse order in insets (b) and (d).

increases with increasing N and is quite good for $N > 10$. The difference in the increase achieved with the largest (30% cache) and smallest caches (1% cache) is around 6% for $N \leq 10$, with the maximum of 6.83% seen for $N = 5$. The gap narrows for higher values of N. Similar trends were observed for varying M and object size S. The results indicate that a few tens of cooperating nodes are sufficient to achieve adequate hit rates. Also, the number of nodes needed to achieve a given increase to hit rate increases with decreasing cache size, by a factor of around two for an order of magnitude smaller cache. This is despite the fact that the base hit rate of a larger cache is higher.

Inset (c) plots the maximum cumulative hit rate achieved by PA when ICCB is not constrained. In this case, we also determined the minimum ICCB needed to achieve the observed hit rate. Bandwidth overhead %, given by $\frac{\text{minimum needed ICCB} - \text{increase in hit rate}}{\text{increase in hit rate}} \times 100\%$ is plotted in inset (d). It can be noted from inset (c) that a hit rate of 100% is reached at $N = N_{100\%} = \lceil \frac{\text{total corpus size}}{\text{cache size}} \rceil$, which is as expected. However, the bandwidth overhead at $N \leq N_{100\%}$ is quite high. This is because when $N \leq N_{100\%}$, the caching system is space constrained, and hence, all the objects are unreplicated and fully shared, including the popular, high-bandwidth objects. As N increases beyond $N_{100\%}$, the number of high bandwidth objects that are replicated increases, bringing down the bandwidth overhead. N needed to achieve 100% hit rate with minimal overhead is roughly an order of magnitude larger for a cache that is an order of magnitude smaller.

5 Conclusion

We have explored cooperative caching among the edge elements of a wireless infrastructure to ease the traffic stress expected in the wireless backhaul and core due to the manifold increase in video traffic. We have proposed an efficient object placement algorithm for cooperative caching that has a constant factor approximation ratio under practical conditions. Our simulation studies show that, in practice, the performance of the algorithm is very close to the optimal, and enabling cooperation among a few 10s of nodes may be sufficient to reap significant benefits. The viability of converting the proposed algorithm to a distributed one will be considered as part of future work.

References

1. Applegate, D., Archer, A., Gopalakrishnan, V., Lee, S., Ramakrishnan, K.K.: Optimal content placement for a large-scale vod system. In: ACM Co-NEXT (2010)
2. Borst, S., Gupta, V., Walid, A.: Distributed caching algorithms for content distribution networks. In: INFOCOM (2010)
3. Breslan, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and zipf-like distributions: Evidence and implications. In: Proceedings of IEEE INFOCOM, pp. 126–134 (1999)
4. Chankhunthod, A., Danzig, P., Neerdaels, C., Schwartz, M., Worrell, K.: A hierarchical internet object cache. In: USENIX Annual Technical Conference, pp. 153–163 (September 1996)
5. Cisco Systems Inc. Cisco visual networking index: Global mobile data traffic forecast update (2009-2014)

6. Fan, L., Cao, P., Almeida, J., Broder, A.: Summary cache: A scalable wide-area web cache sharing protocol. In: SIGCOMM, pp. 254–265 (September 1998)
7. Garey, M., Johnson, D.: Computers and Intractability: a Guide to the Theory of NP-Completeness, vol. ch. 4. W. H. Freeman and company, NY
8. Hafeeda, M., Saleh, O.: Traffic modeling and proportional partial caching for peer-to-peer systems. IEEE/ACM Transactions on Networking 16(6), 1447–1460 (2008)
9. Han, D., Andersen, D., Kaminsky, M., Papagiannaki, D., Seshan, S.: Hulu in the neighborhood. In: COMSNETS (2011)
10. Kangasharju, J., Roberts, J.W., Ross, K.W.: Object replication strategies for content distribution networks. Computer Communication Journal 25(4), 376–383 (2002)
11. Korupolu, M.R., Plaxton, C.G., Rajaraman, R.: Placement algorithms for hierarchical cooperative caching. In: SODA, pp. 586–595 (1998)
12. Song, Y., Ramasubramanian, V., Sirer, E.: Cobweb: a proactive analysis-driven approach to content distribution. In: SOSP, Poster (2005)
13. Venkataramani, A., Weidmann, P., Dahlin, M.: Bandwidth constrained placement in a wan. In: Principles of Distributed Computing, pp. 134–143 (2001)
14. Xu, Z., Bhuyan, L.: Qos-aware object replica placement in cdns. In: GLOBECOM (2005)