This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Distributed Caching Algorithms for Content Distribution Networks

Sem Borst[†], Varun Gupta[⋆], Anwar Walid[†]
[†]Alcatel-Lucent, Bell Labs, 600 Mountain Avenue, P.O. Box 636, Murray Hill, NJ 07974-0636
[⋆]Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

***Abstract*—The delivery of video content is expected to gain huge momentum, fueled by the popularity of user-generated clips, growth of VoD libraries, and wide-spread deployment of IPTV services with features such as CatchUp/PauseLive TV and NPVR capabilities. The 'time-shifted' nature of these personalized applications defies the broadcast paradigm underlying conventional TV networks, and increases the overall bandwidth demands by orders of magnitude. Caching strategies provide an effective mechanism for mitigating these massive bandwidth requirements by replicating the most popular content closer to the network edge, rather than storing it in a central site. The reduction in the traffic load lessens the required transport capacity and capital expense, and alleviates performance bottlenecks.**

In the present paper, we develop light-weight cooperative cache management algorithms aimed at maximizing the traffic volume served from cache and minimizing the bandwidth cost. As a canonical scenario, we focus on a cluster of distributed caches, either connected directly or via a parent node, and formulate the content placement problem as a linear program in order to benchmark the globally optimal performance. Under certain symmetry assumptions, the optimal solution of the linear program is shown to have a rather simple structure. Besides interesting in its own right, the optimal structure offers valuable guidance for the design of low-complexity cache management and replacement algorithms. We establish that the performance of the proposed algorithms is guaranteed to be within a constant factor from the globally optimal performance, with far more benign worst-case ratios than in prior work, even in asymmetric scenarios. Numerical experiments for typical popularity distributions reveal that the actual performance is far better than the worst-case conditions indicate.

I. Introduction

The delivery of digital video content is anticipated to show tremendous growth over the next several years, driven by the huge popularity of user-generated video clips and the expansion of VoD (Video-on-Demand) libraries. It is estimated that YouTube alone attracts tens of millions of viewers a day, generating around 2000 TB of traffic. While the daily number of VoD users is unlikely to be that high, the size of a high-definition movie dwarfs that of a typical video clip, and just 1 user requesting 1 movie a month would involve similar bandwidth demands as 10 users watching 20 clips a month. Even stronger growth is likely to be fueled by the proliferation of IPTV services with personalized features such as CatchUp/PauseLive TV and NPVR (Network Personal Video Recorder) capabilities.

The common characteristic of these 'time-shifted' TV services as well as VoD libraries and sites like YouTube is that users can select from a huge collection of content material at any time they want. This is a radical departure from conventional TV networks, where users can only tune in to a limited number of channels at any given time. As a result, there can be as many different play-out sessions as there are active viewers, which could be in the hundreds of thousands in a major metropolitan area. This runs counter to the broadcast paradigm embraced in conventional TV networks, and raises a need for unicast sessions, increasing the overall bandwidth demands by orders of magnitude.

Caching strategies provide an effective mechanism for mitigating the massive bandwidth requirements associated with large-scale distribution of personalized high-definition video content. In essence, caching strategies exploit storage capacity to absorb traffic by replicating the most popular content closer to the network edge rather than storing it in a central location that requires high processing power and represents a single point-of-failure. The reduction in the traffic load translates into a smaller required transport capacity and capital expense as well as fewer performance bottlenecks, thus enabling better service quality at a lower price, e.g. short and predictable download times. The scope for cost savings and performance benefits from caching increases as the cost of memory continues to drop at a higher rate than that of transmission gear.

The design of efficient caching strategies involves a broad range of interrelated problems, such as accurate prediction of demand, intelligent content placement, and optimal dimensioning of caches. In the context of content delivery networks, these problems inherently entail strong distributed and spatial features, as the caches are physically scattered across the network and the user requests are generated in geographically dispersed nodes. These facets add content look-up and request routing as a major further problem dimension, and severely complicate the reliable estimation of demand and efficient content placement. In tracking popularity for instance, we face a fundamental trade-off between averaging over larger user populations to reduce sampling error and capturing possible variations across heterogeneous user communities. In view of implementation considerations, we further prefer low-complexity content placement algorithms which operate in a mostly distributed fashion and yet implicitly coordinate their actions so as to approach globally optimal performance.

Motivated by the above issues, we aim to devise light-weight cooperative content placement algorithms so as to maximize the traffic volume served from cache and thus minimize the bandwidth cost. The bandwidth cost need not

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

be actual monetary expenses, but could also represent some metric reflecting the congestion levels on the various network links, like link weights in OSPF for example. We assume that demand estimates are given, and that suitable mechanisms for on-line content look-up and request routing are available, e.g. the distributed directory services in [2], [9], [15], [18].

As a canonical scenario, we focus on a cluster of distributed caches, either connected directly or via a parent node. We formulate the content placement problem as a linear program in order to obtain a benchmark of the globally optimal performance. Under certain symmetry assumptions, the optimal solution of the linear program is shown to have a rather simple structure. Besides interesting in its own right, the knowledge of the optimal structure offers useful insight for the design of low-complexity cooperative content placement and eviction algorithms. We establish that the proposed algorithms are guaranteed to operate within a constant factor from the globally optimal performance while requiring only local actions by each of the caches, with benign worst-case ratios, even in asymmetric scenarios. Numerical experiments for typical topologies and popularity distributions demonstrate that the actual performance is far better than the worst-case conditions suggest.

The benefits of cooperative caching have been investigated before in the setting of distributed file systems, starting with the taxonomy in [8], as well as large-scale information systems and web-oriented content distribution networks (CDN's). Prior studies in these contexts include simulation experiments [7], [9], [14], [16], prototypes [20], and analytical results and algorithms [1], [3], [4], [6], [11], [12], [13], [19]. Most of the above-mentioned work has focused on minimizing access latency, and disregarded the bandwidth consumption involved. In fact, the main rationale for cooperative caching in distributed file systems is that bandwidth is assumed to be abundant, and hence can be freely leveraged to improve the response time performance. In contrast, for high-definition video objects with sizes of a few GB's and hour-long durations, minimizing the bandwidth usage is a far more relevant objective than reducing the initial play-out delay by a few hundred milliseconds. A somewhat similar comment applies with respect to web-oriented CDNs, such as Akamai, designed for relatively small web objects as opposed to high-definition video applications.

Although the relevant performance metrics may differ in various application scenarios, a strong conceptual similarity emerges when the notion of 'access cost' is adopted. This cost could either represent the additional latency incurred when fetching content from remote caches or main memory, or the bandwidth consumed when retrieving content from a peer node or video head end, depending on the scenario of interest. Indeed, our mathematical formulation of the content placement problem shows strong resemblance with earlier ones for which a wide range of approximation algorithms have been proposed. Korupolu *et al.* [12] present a polynomial-time exact algorithm for the hierarchical placement problem based on a reduction to minimum-cost flow which generalizes the results in [13]. They further develop a constant-factor distributed amortizing algorithm which is at most 13.93 times from optimal. Awerbuch *et al.* [1] consider general on-line cooperative caching on arbitrary networks and present a polylog(n)-competitive algorithm, where n is the number of caches. Swamy [19] shows that the optimal solution to the relaxed integer program for the case of equal object sizes can be rounded to an integer solution while losing a factor of at most 10, an improvement over the factor of 20.5 in [3]. Kangasharju *et al.* [11] consider object placement with the objective of minimizing the average number of autonomous systems a request must traverse. Replication strategies are proposed and studied numerically, but no performance guarantees are given.

In contrast to [3], [4], [19], we focus on specific topologies, motivated by real system deployments, and prove that the optimal solution of the relaxed integer program has a simple structure. In addition, we use the insight into the optimal structure to develop efficient distributed content placement algorithms with far more favorable performance ratios than the ones in [3], [4], [19]. When it comes to the design of actual content placement and eviction algorithms, a further crucial distinction with earlier studies in different contexts manifests itself. In our setting, transferring content pro-actively into a peer cache carries a significant cost penalty, and it only makes sense to cache an item when it is actually requested. This limits the use of cooperative caching algorithms proposed in different contexts that ignore the bandwidth consumption when moving content around so as to reach the optimal placement.

On a final related note, it is worth observing that caches could be located as close to the users as to actually reside in set-top boxes, creating a degree of similarity with P2P networks [10], [17]. Most work on P2P algorithms however pays little attention to the bandwidth efficiency and the impact on network traffic. The bandwidth usage is in fact an issue of great concern to ISP's, and has spurred a strong interest in mechanisms for localizing (or even throttling) P2P traffic flows. While the latter mechanisms serve a similar purpose, they aim at minimizing the network traffic for a given content placement, whereas our focus is on a slightly more controlled environment where the content placement can be actively managed.

The remainder of the paper is organized as follows. In Section II we present a detailed model description and linear programming problem formulation. Under certain symmetry assumptions, we show in Section III that the optimal solution of the linear program has a rather simple structure. (Because of page limitations, we only outline the main insights and results, and refer to [5] for detailed technical derivations and proof arguments.) In Section IV we examine a scenario with intra-level cache collaboration only, and use the knowledge of the optimal structure to develop distributed content placement algorithms with tight performance guarantees. We then turn attention to a scenario with inter-level cache collaboration only in Section V. In Section VI numerical experiments are presented to evaluate the performance of the proposed content placement algorithms. We make some concluding remarks in Section VII.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

II. Model description and problem formulation

We consider a cache cluster consisting of M 'leaf' nodes indexed $1,\ldots,M$, which are either directly connected or indirectly via node 0 as a common parent somewhere along the path to the root node, as represented in Figure 1. The parent and leaf nodes are endowed with caches of sizes $B_0, B_1, \ldots, B_M$, while the root node is endowed with a cache of sufficient capacity to store all the content items. We assume a static collection of N content items of sizes $s_1,\ldots,s_N$. Denote by d_{in} the demand for the n-th content item in node i, $i = 0,\ldots,M$, $n = 1,\ldots,N$. We assume that there is no direct demand for content at the parent node, i.e., $d_{0n} \equiv 0$ for all $n = 1,\ldots,N$, although that assumption is not particularly essential.

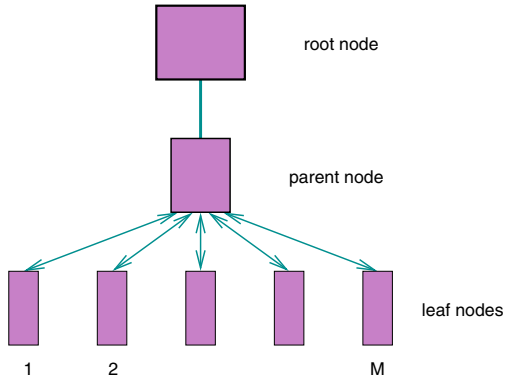


Fig. 1. Graphical illustration of cache cluster.

It is worth emphasizing that the cache cluster need not be a stand-alone network, but could in fact be part of a larger hierarchical tree topology as illustrated in Figure 2. For ease of operation, IPTV networks tend to have a mostly hierarchical tree structure, but they may also have some degree of logical connectivity among nodes within the same hierarchy level, either directly or via a 'U-turn' through a common parent node. Cost analysis reveals that it rarely pays off to install caches at more than one or two levels, even if the network consists of four or five hierarchy levels as is commonly the case. Also, there are implementation issues involved in integrating caches with other lower-layer devices at certain hierarchy levels. Hence, the two-level cache cluster constitutes a fairly canonical scenario that covers most cases of practical interest. However, most of the structural properties that we will obtain in fact extend to any number of hierarchy levels.

Denote by c_0 the unit cost incurred when transferring content from the root node to the parent node, by c_i the unit cost associated with transferring content from the parent node to node i, and by c_{ij} the unit cost incurred when transferring content from leaf node i to leaf node j, $i \neq j = 1,\ldots,M$. We assume that $c_{ij} \leq c_0 + c_i$ for all $i = 1,\ldots,M$, $i \neq j = 1,\ldots,M$, i.e., it is cheaper to transfer content from a peer node than from the root node.

Let the 0–1 decision variable x_{in} indicate whether the n-th content item is stored in the cache of node i or not. The 0–1 variable x_{ijn} indicates whether requests for the n-th content item at node i are served from the cache of node j or not, with $j = -1$ representing the root node.

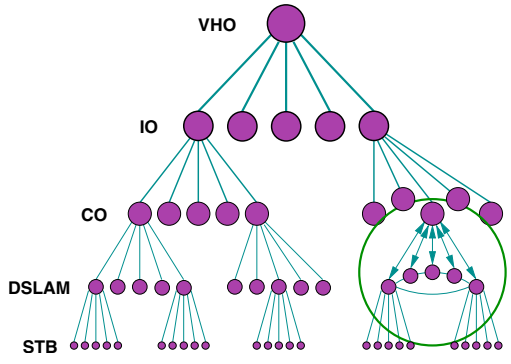


Fig. 2. Cache cluster embedded in hierarchical tree network.

The problem of minimizing the bandwidth expenses may then be formulated as follows

$$\begin{aligned}
\min \quad & \sum_{i=1}^{M}\sum_{n=1}^{N} s_n d_{in}((c_0+c_i)x_{i,-1n} + c_i x_{i0n} + \sum_{j\neq i} c_{ij}x_{ijn}) \\
\text{sub} \quad & \sum_{n=1}^{N} s_n x_{in} \leq B_i, \quad i = 0,\ldots,M \\
& x_{ijn} \leq x_{jn}, \quad i = 1,\ldots,M, i \neq j = 0,1,\ldots,M, \forall n \\
& x_{in} + x_{i,-1n} + x_{i0n} + \sum_{j\neq i} x_{ijn} \geq 1, \quad i = 1,\ldots,M, \forall n,
\end{aligned}$$

where the last inequality constraint will in fact hold with equality at optimality. The (equivalent) problem of maximizing the bandwidth savings may be stated as follows:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{M}\sum_{n=1}^{N} s_n d_{in}((c_0+c_i)x_{in} + c_0 x_{i0n} + \sum_{j\neq i}(c_0+c_i-c_{ij})x_{ijn}) \\
\text{sub} \quad & \sum_{n=1}^{N} s_n x_{in} \leq B_i, \quad i = 0,\ldots,M \\
& x_{ijn} \leq x_{jn}, \quad i = 1,\ldots,M, i \neq j = 0,1,\ldots,M, \forall n \\
& x_{in} + x_{i0n} + \sum_{j\neq i} x_{ijn} \leq 1, \quad i = 1,\ldots,M, \forall n,
\end{aligned}$$

with $\sum_{j\neq i}$ short-hand notation for $\sum_{j=1}^{i-1} + \sum_{j=i+1}^{M}$. We will focus on distributed algorithms with provable performance ratios for maximizing the metric of bandwidth savings.

III. Symmetric scenario

In this section, we assume that the leaf nodes are symmetric in terms of bandwidth costs, demand characteristics, and cache sizes, i.e., $c_i = c$, $c_{ij} = c'_i = c'$, $d_{in} = d_n$, and $B_i = B$ for all $i = 1,\ldots,M$. The assumption of equal bandwidth cost and cache sizes is a fairly natural one, given the way IPTV networks tend to be configured. The assumption of symmetric demands is more restrictive, as the popularity of content items may exhibit variation across different user communities. In

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

practice however, the available demand estimates may simply not have the required level of granularity to reliably distinguish among possibly heterogeneous popularity characteristics, and assuming them to be homogeneous may be the most reasonable option. Furthermore, the optimal content placement turns out to have a distinct and relatively simple structure in case of symmetric demands. Besides interesting in its own right, the knowledge of the optimal structure offers valuable insight for the design of low-complexity cache management and replacement algorithms. As will be shown later, these algorithms perform remarkably well in terms of both worst-case guarantees and average metrics, even when the actual demand characteristics are highly asymmetric.

Henceforth, we will relax the integrality constraints on the decision variables, so as to obtain linear programs which provide lower and upper bounds for problems $\mathrm{P}_{\min}$ and $\mathrm{P}_{\max}$, respectively. The optimal solutions of the continuous relaxations actually turn out to be 'mostly integral', i.e., only have a few fractional variables. If we additionally allow the nodes to partially cache items, which we call 'chunking', then the fractional optimal solution can be exactly achieved under mild conditions on c_{ij}. Specifically, the fractional optimal solution can be achieved via chunking in case of either source-dependent or destination-dependent costs.

Under the above-mentioned symmetry assumptions, it may be shown that problems $\mathrm{P}_{\min}$ and $\mathrm{P}_{\max}$ reduce to

$$\max \quad \sum_{n=1}^{N} s_n d_n (Mc'' u_n + c' \sum_{i=1}^{M} x_{in} + (c' - c) \sum_{i=1}^{M} x_{i0n}) \quad (1)$$

$$\text{sub} \quad \sum_{n=1}^{N} s_n x_{0n} \le B_0 \quad (2)$$

$$\sum_{n=1}^{N} s_n x_{in} \le B, \qquad i = 1, \ldots, M \quad (3)$$

$$u_n \le 1, \qquad n = 1, \ldots, N \quad (4)$$

$$u_n \le x_{0n} + \sum_{i=1}^{M} x_{in}, \qquad n = 1, \ldots, N \quad (5)$$

$$x_{in} \le 1, \qquad i = 0, \ldots, M, n = 1, \ldots, N \quad (6)$$

$$x_{i0n} \le x_{0n}, \qquad i = 1, \ldots, M, n = 1, \ldots, N \quad (7)$$

$$x_{i0n} + x_{in} \le 1, \qquad i = 1, \ldots, M, n = 1, \ldots, N, \quad (8)$$

with $c'' := c + c_0 - c'$ and $u_n := \min\{1, x_{0n} + \sum_{i=1}^{M} x_{in}\}$ representing the fraction of content item n that is collectively stored in the cache cluster, including the parent node.

We now make three important observations about the structure of the optimal solution of the above linear program:

1) Without loss of generality, we can assume $x_{in} = x_n$ for all $i = 1, \ldots, M$ and $n = 1, \ldots, N$.
2) The optimal solution satisfies $x_{0n} + x_n \le 1$.
3) Optimality requires $u_n = 1$ when $u_n < x_{0n} + \sum_{i=1}^{M} x_{in}$.

For compactness, denote $c''' := M(c_0 + c) - (M-1)c' = Mc'' + c'$. Introducing the variables $p_n = u_n - x_{0n}$, $q_n = (x_{0n} + \sum_{i=1}^{M} x_{in} - u_n)/(M-1)$, and using the above three observations, problem (1)–(8) may be equivalently stated as

$$\max \quad \sum_{n=1}^{N} s_n d_n (c''' p_n + c'(M-1) q_n + Mc_0 x_{0n}) \quad (9)$$

$$\text{sub} \quad \sum_{n=1}^{N} s_n x_{0n} \le B_0 \quad (10)$$

$$\sum_{n=1}^{N} s_n (p_n + (M-1) q_n) \le MB \quad (11)$$

$$p_n + x_{0n} \le 1, \qquad n = 1, \ldots, N \quad (12)$$

$$q_n + x_{0n} \le 1 \qquad n = 1, \ldots, N \quad (13)$$

Problem (9)–(13) represents a knapsack-type problem, with two knapsacks of sizes B_0 and MB and $2N$ items of sizes $a_n = s_n$, $a_{N+n} = (M-1)s_n$, $n = 1, \ldots, N$. Items $N+1, \ldots, 2N$ cannot be included in the first knapsack. Inclusion of item n in the first knapsack precludes inclusion of that item as well as item $n+N$ in the second knapsack, $n = 1, \ldots, N$. The value of item n when included in the first knapsack is Mc_0, while the values of items n and $N+n$ when included in the second knapsack are c''' and $(M-1)c'$, respectively. The latter observation may be interpreted as follows. Storing item n in the first of the M leaf nodes yields a bandwidth savings of $s_n d_n c'''$, while storing it in each additional node yields a further bandwidth savings of $s_n d_n c'$. The variable p_n indicates whether or not at least one copy of content item n is stored in the leaf nodes. (As we will see later, a fractional value can occur for at most one content item.) The variable q_n can only be non-zero when $p_n = 1$, and indicates whether content item n is fully replicated across the leaf nodes or not. (Again, a fractional value can occur for at most one content item, as will be seen later.) The variable x_{0n} indicates whether content item n is stored in the parent cache or not. (A fractional value can occur for at most two content items.) When $x_{0n} = 1$, the values of p_n and q_n must be zero. This reflects the fact that storing a content item both in the parent and leaf nodes can not be optimal.

The optimal solution of problem (9)–(13) has a distinct structure which we now proceed to describe. Assume that the content items are indexed in descending order of d_n, i.e., $d_1 \ge d_2 \ge \cdots \ge d_N$. We distinguish two cases, depending on whether Mc is larger or smaller than $(M-1)c'$, which will be treated separately in the next two subsections.

A. Case A: $Mc \ge (M-1)c'$

We first consider the case $Mc \ge (M-1)c'$, which means that it is more advantageous to store unreplicated content in the leaf nodes than in the parent node.

Observe that, given the values of x_n, x_{0n}, we can assume without loss of generality the following caching behavior:

- Subcase $c < c'$: The parent node caches the first x_{0n} fraction of item n and the leaf nodes cache arbitrary x_n fractions of item n, distinct from the parent node, and to maximize the coverage of item n.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

- Subcase $c \geq c'$: The leaf nodes cache x_n fractions of item n so as to maximize the coverage of item n, and the parent node caches an arbitrary x_{0n} fraction of item n which is still uncovered by the leaf nodes.

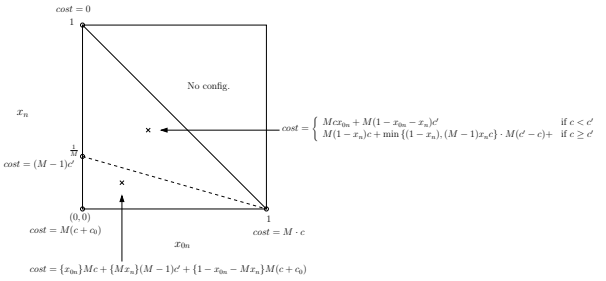
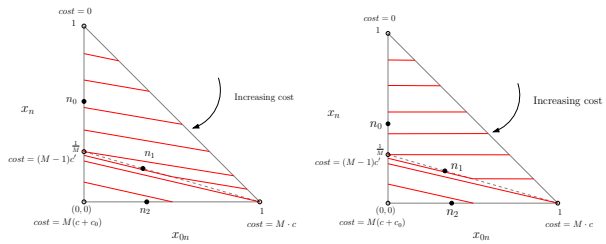


Fig. 3. An illustration of the valid symmetric configurations and their costs.

Now we can obtain the cost associated with an item with some configuration (x_{0n}, x_n). As shown in Figure 3, the space of possible configurations is the area $x_{0n} + x_n \leq 1$. We distinguish two regions: $x_{0n} + Mx_n \leq 1$ (item n is not completely covered by the parent and leaf nodes) and $x_{0n} + Mx_n \geq 1$ (item n is completely covered and possibly overreplicated). For the case $c < c'$, the cost of any configuration lying in any of the above two regions can be written as the convex combination of the costs of the vertices of the triangular region it belongs to, with the coefficients being the respective barycentric coordinates (the case $c \geq c'$ needs special handling). This is explicitly illustrated in Figure 3 for a point in the region $x_{0n} + Mx_n \leq 1$.

Given the above, we now introduce the concept of isocost curves of different configurations which we will use to obtain the structure of optimal configuration for different parameter settings. Simply put, isocost curves are curves connecting configurations ((x_{0n}, x_n) pairs) which have the same cost. For case A, the isocost curves are shown in Figure 4. A configuration of the entire system can be represented by N points on this plot, one for each of the items. We refer to the configurations corresponding to (i) full replication in leaf nodes, (ii) single copy in leaf node only, (iii) single copy in parent node only, and (iv) no copy as *vertex configurations*.

A case-by-case analysis yields that there can be at most two items in non-vertex configurations as stated in the next proposition.



(a) Subcase $c < c'$ (b) Subcase $c \geq c'$

Fig. 4. Illustration of iso-cost curves (solid lines) in the configuration space for case A.

Proposition 3.1: In the optimal configuration for case A, there can be at most two items which are in a non-vertex configuration.

It is easy to argue from the structure of isocost curves that if there is at least one item with a single copy in the parent node (0,1) and at least one item with a single copy in the leaf nodes $(1/M, 0)$, the possible non-vertex points are of the form n_0, n_1 and n_2 ($n_0 < n_1 < n_2$ under optimality) as shown in Figure 4. Thus under the optimal configuration, items $1, \ldots, n_0 - 1$ are cached in all the leaf nodes, items $n_0 + 1, \ldots, n_1 - 1$ have a single copy in the leaf nodes, items $n_1 + 1, \ldots, n_2 - 1$ have a copy in the parent node only, and items $n_2 + 1, \ldots, N$ are not cached anywhere. Further, one of n_0, n_1 or n_2 must have a vertex configuration, i.e., one of the following must be true: (i) item n_0 has a single copy at a leaf node; (ii) item n_1 has a single copy at the parent node; (iii) item n_2 is not cached.

B. Case B: $Mc \leq (M-1)c'$

We now turn to the case $Mc \leq (M-1)c'$, which means that it is more attractive to store unreplicated content in the parent cache than in the leaf nodes. In the previous case, the most popular unreplicated content was stored in the leaf nodes, and the second tier of content was stored in the parent cache, which will now be reversed.

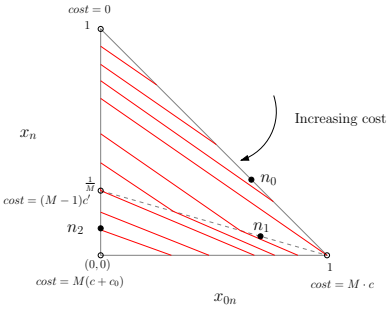


Fig. 5. Illustration of iso-cost curves (solid lines) in the configuration space for case B.

Proposition 3.2: In the optimal configuration for case B, there can be at most two items which are in a non-vertex configuration.

Similar to case A, Figure 5 shows the isocost curves for case B. As in case A, if there is at least one item with a single copy in the parent node and at least one item with a single copy in the leaf nodes, the only possible non-vertex points are of the form n_0, n_1 and n_2 as shown in Figure 5 (and we avoid discussing the less interesting corner cases). Thus under the optimal configuration, items $1, \ldots, n_0 - 1$ are cached in all the leaf nodes, items $n_0 + 1, \ldots, n_1 - 1$ have a copy in the parent node only, items $n_1 + 1, \ldots, n_2 - 1$ have a single copy in the leaf nodes, and items $n_2 + 1, \ldots, N$ are not cached anywhere. Further, one of n_0, n_1 or n_2 must have a vertex configuration, i.e., one of the following must be true: (i) item n_0 has a single copy at a parent node node; (ii) item n_1 has a single copy at a leaf node; (iii) item n_2 is not cached.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

IV. Intra-level cache cooperation

In this section we focus on the special case where content can only be stored at the leaf nodes and not at the parent node, i.e., $B_0 = 0$, which we will refer to as *intra*-level cache cooperation.

A. Symmetric demands, cache sizes and costs

We start with the case that the leaf nodes are symmetric in terms of bandwidth cost, demand characteristics and cache sizes, i.e., $c_i = c$, $c'_{ij} = c'_i = c' < c_0 + c$, $d_{in} = d_n$ and $B_i = B$ for all $i = 1, \ldots, M$. In a similar fashion as in Section III, problem $\mathrm{P}_{\max}$ may then be shown to simplify to

$$\max \quad \sum_{n=1}^{N} s_n d_n (c''' p_n + (M-1) c' q_n) \tag{14}$$

$$\text{sub} \quad \sum_{n=1}^{N} s_n (p_n + (M-1) q_n) \le MB \tag{15}$$

$$p_n \le 1, \qquad n = 1, \ldots, N \tag{16}$$

$$q_n \le 1, \qquad n = 1, \ldots, N \tag{17}$$

with $c''' := M(c_0 + c) - (M-1)c' = Mc'' + c'$.

Problem (14)–(17) represents a knapsack-type problem, with a knapsack of size MB and $2N$ items of sizes $a_n = s_n$, $a_{N+n} = (M-1)s_n$, $n = 1, \ldots, N$. The value of item n is $Mc'''s_n d_n$, while the value of item $N+n$ is $(M-1)c's_n d_n$.

The optimal solution of problem (14)–(17) has a relatively simple structure which we now proceed to describe. Without loss of generality, assume that the content items are indexed in descending order of d_n, i.e., $d_1 \ge d_2 \ge \cdots \ge d_N$. For convenience, we further assume unit-size content items, i.e., $s_n = 1$ for all $n = 1, \ldots, N$.

1) For some N_1, items $1, \ldots, N_1$ are replicated at each node.
2) Item $N_1 + 1$ is possibly replicated but not at all the nodes.
3) For some $N_2 > N_1 + 1$, items $N_1 + 1, \ldots, N_2$ have a single copy in the cluster.

Based on the above insights, we now propose a simple distributed algorithm, `Local-Greedy` for cooperative caching. Specifically, with item n at node i we associate a utility value which is defined as $d_n c'''$ if item n is not cached in any other node, and $d_n c'$ otherwise (i.e. if item n is cached in some other node). As opposed to a central agent populating all the caches, under `Local-Greedy` each node modifies its local content to achieve the maximum gain in global utility, and is distributed in this sense.

Algorithm `Local-Greedy`
Select a node i and an item n. If item n is currently not stored at node i, and it has higher utility than some item m (e.g. with minimum utility) that is currently stored at node i, then replace item m by item n.

It would be natural for the selection of a node i and item n to be governed by actual received requests, meaning that items only get cached when they are being transferred through nodes anyway and are not pre-fetched. It is clear that `Local-Greedy` must eventually converge to some stable configuration, assuming each node-item combination is selected with nonzero probability. While in the restrictive case of symmetric demands, cache sizes and bandwidth costs, it might seem that the globally optimal configuration should be reached, this is not always the case as the next theorem shows.

Theorem 4.1: Under symmetric demands, cache sizes and bandwidth costs, *`Local-Greedy`* is a $\frac{4}{3}$-approximation algorithm for the metric of maximizing bandwidth savings. Further, there exist demand vectors for which *`Local-Greedy`* may not achieve more than $\frac{3}{4}$ of the optimal.

B. Arbitrary demands, cache sizes and costs

The `Local-Greedy` algorithm is a special case of the algorithm `Local-Greedy-Gen` specified below:

Algorithm `Local-Greedy-Gen`
Select a node i and an item n. If item n is currently not stored at node i, then replace that by some item m that is currently stored at node i, if and only if that increases the global utility.

We now provide matching upper and lower bounds on the performance of `Local-Greedy-Gen` under arbitrary demand vectors, cache sizes and bandwidth costs.

Theorem 4.2: *`Local-Greedy-Gen`* is a 2-approximation algorithm for the metric of maximizing bandwidth savings under arbitrary demands, cache sizes and bandwidth costs. Further, there exist demand vectors for which *`Local-Greedy-Gen`* may not achieve more than $\frac{1}{2}$ of the optimal, even under equal cache sizes and symmetric bandwidth costs.

The 2-approximation ratio of `Local-Greedy-Gen` extends to arbitrary topologies – the only requirement is that all nodes have knowledge of the global utility. For the cache cluster scenario considered in this paper, the latter knowledge only requires limited information exchange under suitable assumptions on the bandwidth costs, which may be enabled by existing protocols such as Internet Cache Protocol or Cache Digest.

V. Inter-level cache cooperation

In this section we focus on the special case where content can only be fetched from the parent node and not from a peer, i.e., $c_{ij} = \infty$, which we will refer to as *inter*-level cache cooperation.

A. Homogeneous demands, equal cache sizes

We start with the case that the leaf nodes have equal cache sizes, i.e., $B_i = B$ for all $i = 1, \ldots, M$, and that the demand characteristics satisfy $d_{i1} \ge d_{i2} \ge \cdots \ge d_{iN}$ for all $i = 1, \ldots, M$, possibly after suitable reindexing of the content items. Note that the demand characteristics are no longer assumed to be symmetric, and could for example only be identical up to node-dependent scaling factors, representing the overall traffic volumes at the various nodes.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

In this case, optimality forces $x_{ijn} = 0$, $j \neq -1, 0, i$, and problem $\mathrm{P}_{\max}$ may be shown to reduce to

$$\max \quad \sum_{i=1}^{M}\sum_{n=1}^{N} s_n d_{in}((c_0 + c_i)x_{in} + c_0 \sum_{i=1}^{M} x_{i0n}) \qquad (18)$$

$$\text{sub} \quad \sum_{n=1}^{N} s_n x_{0n} \leq B_0 \qquad (19)$$

$$\sum_{n=1}^{N} s_n x_{in} \leq B, \;\; i = 1, \ldots, M \qquad (20)$$

$$x_{i0n} \leq x_{0n}, \;\; i = 1, \ldots, M, n = 1, \ldots, N \qquad (21)$$

$$x_{in} + x_{i0n} \leq 1, \;\; i = 1, \ldots, M, n = 1, \ldots, N \qquad (22)$$

The optimal solution of problem (18)–(22) has a relatively simple structure. Specifically, the most popular content items are fully replicated in all the leaf nodes, and single copies of the second-tier items are stored in the cache of the parent node.

It is worth observing that the optimal content placement can be achieved through a simple 'greedy' strategy where each individual node aims to maximize the local hit rate, i.e., the fraction of traffic served from cache of the requests it receives. In that case, each of the leaf nodes will store the most popular content items. As a result, the parent node will not receive any requests for the most popular items, and end up storing the second-tier items.

B. Arbitrary demands, cache sizes

We now analyze the performance of the greedy content placement strategy under arbitrary demands and cache sizes. While the strategy will no longer yield the optimal content placement in general, it is guaranteed to achieve a certain fraction of the maximum achievable bandwidth savings as the next theorem shows.

Theorem 5.1: For arbitrary cost values, item sizes, and cache sizes, the greedy content placement strategy is guaranteed to achieve at least a fraction $\frac{(M-1)c_{\min}+Mc_0}{(M-1)c_{\min}+(2M-1)c_0} \geq \frac{M}{2M-1}$ of the maximum achievable bandwidth savings, with $c_{\min} := \min_{i=1,\ldots,M} c_i$.

The performance ratio in the above proposition is tight, and attained in (unnatural) cases where some items are highly popular in some leaf nodes and not popular at all in others.

VI. Numerical experiments

In this section we present the numerical experiments that we have conducted to evaluate the performance of the proposed algorithms. Throughout we assume that there are $M = 10$ leaf nodes, each endowed with a cache of size B. The parent node has a cache of size B_0 (possibly zero), while the root node is endowed with a cache of sufficient capacity to store all the content items. The parameters c_0, c and c' represent the unit cost incurred when transferring content from the root node to the parent node, from the parent node to one of the leaf nodes, and between any pair of leaf nodes, respectively. In all the experiments we assume $c_0 = 2$, $c = 1$, and $c' = 1$.

The system offers a collection of $N = 10,000$ content items. For convenience, we assume the content items to have a common size of $S = 2$ GB, although this is not particularly essential. For compactness, denote by $K = B/S$ the number of content items that can be stored in each of the leaf nodes.

Each of the leaf nodes receives an average of 1 request every 160 seconds i.e., the aggregate request rate per leaf node $\nu = 0.00625$ sec^{-1}. We assume that the popularity of the various content items is governed by a Zipf-Mandelbrot distribution with shape parameter α and shift parameter q, i.e., the normalized popularity p_n of the n-th most popular item is proportional to $(q+n)^{-\alpha}$. The request rate for the n-th most popular item at each of the leaf nodes is $d_n = p_n\nu$, with ν the total request rate per leaf node defined above.

Below we present two sets of experimental results. In the first set of results, we assess the gains from cooperative caching, and demonstrate that judicious replication can yield significant bandwidth savings, even with small cache sizes. In the second set of experiments, we examine the performance of the `Local-Greedy` algorithm, and show that it operates close to globally optimal performance.

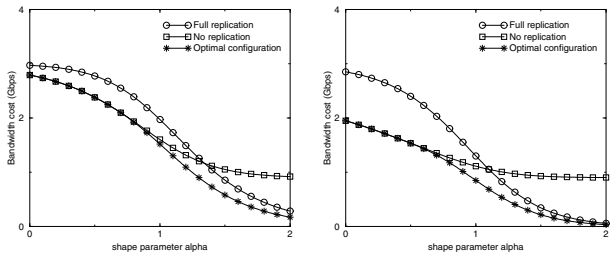
A. Gains from cooperative caching

In order to quantify the gains from cooperative caching, we compare the minimum bandwidth cost as characterized by the optimal solution of problem (14)–(17) with the bandwidth cost incurred in two other scenarios: (i) full replication, where each leaf node stores the K most popular content items; (ii) no replication, where only a single copy of the MK most popular content items is stored in one of the leaf nodes. The bandwidth costs for these three scenarios are compared as function of the shape parameter α of the Zipf-Mandelbrot distribution with the shift parameter fixed at $q = 10$. Note that without caching the bandwidth cost would be $M\nu S(c+c_0) = 10 \times 0.00625 \times 2 \times 3 = 0.375$ GBps, which amounts to 3 Gbps.

Figures 6(a)–(c) show the results for cache sizes $B = 200$ GB, $B = 1$ TB, and $B = 2$ TB, respectively. Figure 6(d) shows similar results for a cache size $B = 1$ TB, where in addition the parent node has a cache of size $B_0 = 2$ TB. As the results demonstrate, the bandwidth costs markedly decrease with increasing values of α, reflecting the well-known fact that caching effectiveness improves as the popularity distribution gets steeper. Even when $B = 200$ GB, which means that the collective cache space can hold no more than 10% of the total content, caching reduces the bandwidth cost to a fraction of what they would have been in the absence of caching.

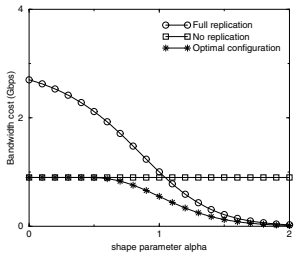
It is worth observing that the best between either full or zero replication is often not much worse than the optimal content placement. However, neither full nor zero replication performs well across the entire range of α values, and it is crucial to adjust the degree of replication based on the steepness of the popularity distribution. In case of the Zipf-Mandelbrot distribution, one could in principle calculate the optimal degree of replication as function of α. In practice, however, the value of α may not be so easy to obtain, or the popularity statistics may not even conform to a Zipf-Mandelbrot distribution

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

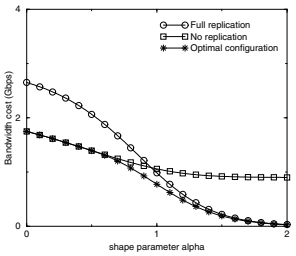


(a) Cache size $B = 200$ GB

(b) Cache size $B = 1$ TB

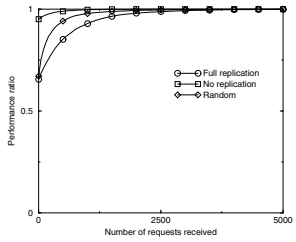


(c) Cache size $B = 2$ TB

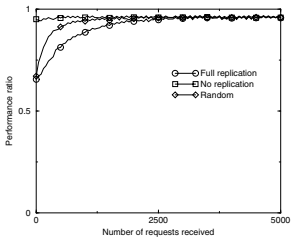


(d) $B = 1$ TB, $B_0 = 2$ T

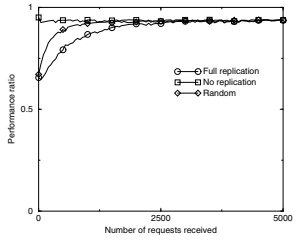
Fig. 6. Bandwidth cost as function of shape parameter α for various scenarios and cache sizes.



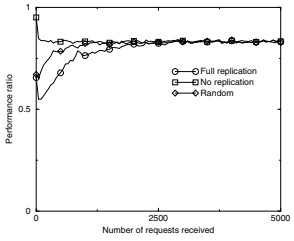
(a) Static popularity



(b) Slow aging $R = 200$



(c) Moderate aging $R = 100$



(d) Fast aging $R = 20$

Fig. 7. Performance ratio as function of number of requests, with possible dynamic content ingestion and aging.

altogether. Hence it makes more sense to adjust the degree of replication based on the actual observed demands, rather than fitting a particular distribution. The algorithms described in Section IV do just that, without requiring any knowledge of α or even relying on the popularity statistics obeying a Zipf-Mandelbrot distribution.

B. Performance of `Local-Greedy` *algorithm*

We now proceed to examine the performance of the `Local-Greedy` algorithm. Throughout we assume that each of the leaf nodes has a cache of size $B = 1$ TB, and that the popularity of the various content items is governed by a Zipf-Mandelbrot distribution with shape parameter $\alpha = 0.8$ and shift parameter $q = 10$.

In order to study the dynamic evolution of the content placement, we conduct a simulation experiment where the various leaf nodes receive requests over time, sampled from the above-described popularity distribution. Whenever a particular node receives a request for an item that it has not presently stored, it decides whether to cache it and if so, which currently stored item to evict, as prescribed by the `Local-Greedy` algorithm. In order to monitor the performance and convergence, we track the objective value over time, and compare it with the value of the optimal content placement. In the optimal content placement, items 1 through 165 are fully replicated, and single copies of items 166 through 3515 are stored. For the initial content placement, we distinguish three different scenarios: (i) full replication, where each leaf node initially stores the K most popular content items; (ii) no replication, where initially only a single copy of the MK most popular content items is stored in one of the leaf nodes. (iii) random, where each leaf node stores K content items selected uniformly at random, independently of all other leaf nodes.

Figure 7(a) shows the performance of the `Local-Greedy` algorithm as ratio of the optimal content placement. Note that the objective value achieved by the algorithm gets progressively closer to the optimum as the system receives more requests and replaces content items over time. After only 3000 requests (out of a total number 10,000 content items) the `Local-Greedy` algorithm has come to within 1% of the optimal content placement, and stays there, performing markedly better that the worst-case ratio of 3/4 might suggest.

While the algorithm converges for all three initial states, the scenario with no replication appears to be the most favorable one. This may be explained from the fact that in the optimal content placement only items 1 through 165 are fully replicated. It takes relatively few requests to replicate these items starting from no replication, whereas it takes far more requests to replace all the duplicates of items 166 through 500 by single copies of items 501 through 3515 starting from full replication.

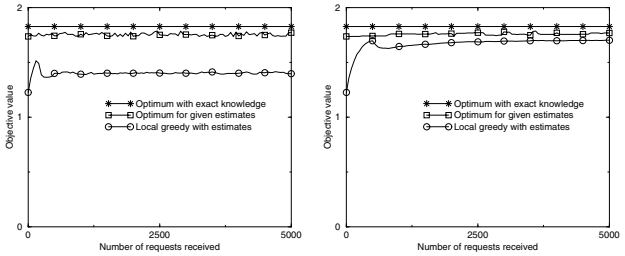
In the next experiment, we explore how dynamic content ingestion and aging affects the performance of the `Local-Greedy` algorithm. As a point of reference, the overall popularity law is assumed to remain the same over time, in the sense that the relative popularity of the n-th most popular item continues to be p_n. However, the relative popularity of a given item decays over time, and we specifically assume that the rank of each item is decremented by one after every R requests received in the system. In other words, for every request that the system receives, the ranks of all the items are

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

decremented by $1/R$, which may be interpreted as a measure for the rate of aging. In order to keep the overall popularity law the same, the bottom item with rank N is supposed to be removed while a new item with rank 1 is ingested after every R requests received by the system. Equivalently, each individual item receives an average of R requests over its active lifetime.

Figures 7(b)–(d) show the results for various aging factors. The `Local-Greedy` algorithm continues to 'converge', but now only comes to within a certain margin from the 'optimal' content placement, which increases with the rate of aging. This may be explained from the fact that the constant churn requires the cache content to be continuously refreshed to maintain optimality, whereas previously the cache content gradually settled into a fixed state as the optimum was approached. The `Local-Greedy` algorithm is constrained in terms of the rate at which it can make the required updates by the requests that it receives, and suffers more as the required rate of updates grows relative to the number of requests, i.e., as the aging rate increases. By the same token, the 'optimal' content placement is granted an unfair advantage here, as the additional bandwidth that is required to maintain optimality, is not accounted for. In other words, it only provides an upper bound, and the `Local-Greedy` algorithm may therefore in fact be closer to the true optimum than the numbers suggest.

So far, we have assumed that the relative popularities of the various content items are known exactly. In practice, popularities are unlikely to be known with perfect accuracy, and will need to be estimated. In order to examine the impact of estimation errors, we conducted an experiment where the popularities are no longer assumed to be known, but estimated based on the actual received requests using geometrically smoothed average with some time constant T as proxy for the length of the measurement window. Figures 8(a)–(b) plot the results for various values of the time constant. As to be expected, the optimum is approached closer as the accuracy of the estimates improves for larger time constants, although the initial convergence is possibly slower. For comparison purposes, the figures also show what the optimum objective value is for the given popularity estimates. This provides an indication for the best that any algorithm in general and `Local-Greedy` in particular can be expected to do, given the available estimates, and suggests that for larger time constants the performance gap is mostly due to estimation errors.



(a) Time constant $T = 100$ (b) Time constant $T = 10,000$

Fig. 8. Objective value as function of number of requests, with popularity estimation.

VII. Conclusion

We have developed lightweight cooperative cache management algorithms, which achieve close to globally optimum performance, with favorable worst-case ratios, even in asymmetric scenarios. Numerical results for typical popularity distributions demonstrate that the actual performance is far better than the worst-case guarantees suggest. The numerical results also illustrate the complex interaction between distributed cache cooperation and popularity estimation in the presence of dynamic content ingestion and aging. Finding optimal content placement algorithms and performance bounds for time-varying popularity statistics remains as a challenging issue for further research.

References

[1] B. Awerbuch, Y. Bartal, A. Fiat (1998). Distributed paging for general networks. *J. Alg.* **28** (1), 67–104.
[2] B. Awerbuch, D. Peleg (1995). Online tracking of mobile users. *J. ACM* **37**, 1021–1058.
[3] I.D. Baev, R. Rajaraman (2001). Approximation algorithms for data placement in arbitrary networks. *Proc. SODA 2001*, 661–670.
[4] I.D. Baev, R. Rajaraman, C. Swamy (2008). Approximation algorithms for data placement problems. *SIAM J. Comput* **38**, 1411–1429.
[5] S.C. Borst, V. Gupta, A. Walid (2009). Self-organizing algorithms for cache cooperation in content distribution networks. Technical Report ITD-08-48439B, Alcatel-Lucent, Bell Labs. `http://ect.bell-labs.com/who/sem/publications/publications//bgw08.pdf`
[6] H. Che, Z. Wang, Y. Tung (2001). Analysis and design of hierarchical Web caching systems. In: *Proc. IEEE Infocom 2001*, 1416–1424.
[7] M.D. Dahlin, R.Y. Wang, T.E. Anderson, D.A. Patterson (1994). Cooperative caching: using remote client memory to improve file system performance. In: *Proc. OSDI '94*.
[8] L.W. Dowdy, D.V. Foster (1982). Comparative models of the file assignment problem. *ACM Comput. Surv* **14**, 287–313.
[9] L. Fan, P. Cao, J. Almeida, A.Z. Broder (1998). Summary cache: A scalable wide-area Web cache sharing protocol. In: *Proc. ACM SIGCOMM '98*, 254–265.
[10] Y. Huang, Z. Xias, Y. Chen, R. Jana, M. Rabinovich, B. Wei (2007). When is P2P technology beneficial to IPTV service? In: *Proc. ACM NOSSDAV '07*.
[11] J. Kangasharju, J.W. Roberts, K.W. Ross (2002). Object replication strategies in content distribution networks. *Comp. Commun. J.* **25**, 376–383.
[12] M.R. Korupolu, C.G. Plaxton, R. Rajaraman (2001). Placement algorithms for hierarchical cooperative caching. *J. Alg.* **38** (1), 260–302. Preliminary version appeared in: *Proc. SODA '99*, 586–595.
[13] A. Leff, J.L. Wolf, P.S. Yu (1993). Replication algorithms in a remote caching architecture. *IEEE Trans. Parallel Distr. Syst.* **4** (11), 1185–1204.
[14] J. Ni, D.H.K. Tsang (2005). Large-scale cooperative caching and application-level multicast in multimedia content delivery networks. *IEEE Commun. Mag.* **43**, 98–105.
[15] C.G. Plaxton, R. Rajaraman, A.W. Richa (1999). Accessing nearby copies of replicated objects in a distributed environment. *Th. Comput. Syst.* **32**, 241–280.
[16] P. Sarkar, J.H. Hartman (2001). Hint-based cooperative caching. *ACM Trans. Comp. Syst.* **18**, 387–419.
[17] T. Silverston, O. Fourmaux (2007). Measuring P2P IPTV systems. In: *Proc. ACM NOSSDAV '07*.
[18] M. van Steen, F.J. Hauck, A.S. Tanenbaum (1996). A model for worldwide tracking of distributed objects. In: *Proc. TINA '96*, 203–212.
[19] C. Swamy (2004). Algorithms for the data placement problem. Preprint.
[20] G.M. Voelker, E.J. Anderson, T. Kimbrel, M.J. Feeley, J.S. Chase, A.R. Karlin, H.M. Levy (1998). Implementing cooperative prefetching and caching in a globally-managed memory system. In: *Proc. ACM SIGMETRICS '98*.