Collaborative Hierarchical Caching with Dynamic Request Routing for Massive Content Distribution

Jie Dai†, Zhan Hu†, Bo Li†, Jiangchuan Liu‡, Baochun Li§
†*Hong Kong University of Science and Technology*
‡*Simon Fraser University*, §*University of Toronto*

***Abstract*—Massive content delivery in metropolitan networks has recently gained much attention with the successful deployment of commercial systems and an increasing user popularity. With an enormous volume of content available in the network, as well as the growing size of content owing to the popularity of high-definition video, the exploration of capacity in the caching network becomes a critical issue in providing guaranteed service. Yet, collaboration strategies among cache servers in emerging scenarios, such as IPTV services, are still not well understood so far. In this paper, we propose an efficient collaborative caching mechanism based on the topology derived from a real-world IPTV system, with a particular focus on exploring the capacity of the existing system infrastructure. We observe that collaboration among servers is largely affected by the topology characteristics and heterogeneous capacities of the network. Meanwhile, dynamic request routing within the caching network is strongly coupled with content placement decisions when designing the mechanism. Our proposed mechanism is implemented in a distributed manner, and is amenable to practical deployment. Our simulation results demonstrate the effectiveness of our proposed mechanism, as compared to conventional cache cooperation with static routing schemes.**

I. Introduction

The delivery of large volumes of content — such as high-definition videos and operating system software updates — is becoming a daily routine over the Internet, mostly provided by content service providers. For example, Netflix now accounts for around 30% of peak downstream traffic in US with its rapid growth of online video subscribers. Operating system releases, such as Ubuntu 11.04, have also relied on the public Internet using mirror servers deployed worldwide [1]. Furthermore, more cloud-oriented applications require massive amounts of content to be delivered between service providers and end users.

Such explosively increasing demand has posed significant challenges to network infrastructure providers, who promise quality of service to users. On one hand, the large amount of available content consumes a huge amount of storage resources if such content is replicated at service providers. On the other hand, the delivery of massive content may also overwhelm link capacities, raising the issue of balancing the tradeoff between network infrastructure investments and quality of service guarantees. As such, a critical challenge now is to best utilize existing network infrastructures to better support future demand for massive content delivery.

It is well known that the network infrastructure at a service provider is, in general, a caching network, in which multiple cache servers are deployed at different locations. Servers are connected in a certain topology, and cooperate to resolve requests sent from clients. More specifically, cache servers collaboratively make storage decisions and route content requests inside the cache hierarchy. This requires a carefully designed placement strategy for cached content, along with a specific request routing mechanism. In a conventional caching network that supports web content delivery, the perceived service quality can be improved with collaborative caching by minimizing the end-to-end latency of data packets. However, the problem becomes even more challenging when the collaborative caching mechanism is used to aid massive content delivery.

Despite a large amount of deployed commercial systems and modelling research on traditional hierarchical caching networks, there exists few works in the literature that address challenges with collaborative caching when massive content delivery is considered, especially with implications from real-world systems. In this paper, we investigate the capacity provisioning problem in hierarchical caching networks, based on a real-world IPTV system. Our caching network topology has been obtained from a commercial deployment of China Telecom Guangzhou [2], which provides IPTV service to millions of users in a metropolitan network. According to our observations, the overall topology is similar to a hierarchical structure that is widely applied in web caching systems [3]. However, the collaborative caching problem that we address in this paper exhibits a number of different features. Existing works that investigate cache hierarchies have mainly assumed static content request routing mechanisms. Requests are simply forwarded to the upper-layer parent server when the content is not locally available. Such fixed routing paths have simplified the problem of finding the optimal content placement inside the network. In contrast, in this paper, we point out that in order to maximize the potential for cache cooperation in the existing infrastructure, dynamic request routing needs to be designed jointly with content placement strategies in a tightly coupled fashion.

Another important feature that distinguishes our work from similar problems is the large amount of bandwidth that is consumed as massive contents are delivered [4]. The link capacity is rarely the bottleneck for web cache servers [5].

†The research was supported in part by a grant from NSFC/RGC under the contract N_HKUST610/11, by grants from HKUST under contracts RPC11EG29, SRFI11EG17-C and SBI09/10.EG01-C, by a grant from Huawei Technologies Co. Ltd. under the contract HUAW18-15L0181011/PN, by a grant from Guangdong Bureau of Science and Technology under the contract GDST11EG06.

Thus, the methods applied in web systems usually define the optimization objectives using the notion of "cost," which is derived from the estimation of distance between requests and cache servers. However, it is necessary to consider the impact of hard bandwidth constraints for links in our target system. In this sense, a dynamic request routing scheme also helps to avoid the over-utilization of links in the cache hierarchy. Comparatively speaking, our study tries to optimize the maximum amount of supported requests while emphasizing the link capacity constraint, which is a practical concern in real-world IPTV systems.

In this paper, we take advantage of insights from a real-world topology to derive an efficient collaborative caching mechanism. Based on heterogeneous request patterns at different locations of the system and asymmetric settings of cache capacities, we design our strategic content placement strategy and the corresponding request routing rules. More specifically, we decompose these two problems into different layers of cooperation, so that practical distributed algorithms can be achieved. Our simulation results show that excellent performance can be achieved with the proposed mechanism, compared to conventional cache cooperation with static routing.

The remainder of the paper is organized as follows. In Sec. II, we discuss our contribution with comparisons to related works. In Sec. III, we present characteristics of our real-world system, along with a description of our problems at hand. Sec. IV introduces the cache cooperation framework and the corresponding caching strategies. Sec. V evaluates the performance of the proposed caching mechanisms. Finally, we conclude the paper in Sec. VI.

II. Related Work

The cooperation of cache servers in hierarchical caching networks has long been investigated in web caching systems. Wolman *et al.* [5] evaluated the potential advantages and drawbacks of inter-proxy cooperation in a large-scale Web environment. The analysis is conducted on a tree-like cache hierarchy through extensive measurements. Korupolu *et al.* [3] considered a cache content placement problem in a hierarchical network environment. They provided approximation algorithms that minimize the average access cost measured by the hop count. Rodriguez *et al.* [6] proposed a caching architecture in hierarchical web caching systems where leaf nodes connected to the same parent node can cooperate together. Similar cache content placement strategies, proposed in [7]–[9], that formulated the problem with the objective of minimizing average object access costs, which is affected by the distance between the source of requests and the closest server with the requested content. Our work differs in its focus on massive content delivery, that consumes large amounts of link bandwidth in the network infrastructure. As a result, our work consists of different problem constraints and optimization objectives.

Request routing policies have also been discussed in the context of Content Delivery Network (CDN). Wang *et al.* [10] explored strategies to redirect requests in CDNs that aimed to achieve load balancing and traffic locality. Laoutaris *et al.* [11] introduced the cache inference problem that infers the characteristics of caching agents. The design is used in the decision of forwarding missed streams to available proxy caches. Betkas *et al.* [12] discussed joint content placement and the request routing strategies. However, request routing in these studies generally represents server selection strategies in CDNs. In contrast, our work focuses on a real-world system infrastructure that provides specific link-level routing schemes. With dynamic request routing, we maximize the potential of capacity provisioning in the network, combined with concerns on strategic cache content placement.

Cache content placement in IPTV systems has recently drawn attention in several studies. Borst *et al.* [13] developed cooperative cache management algorithms that aimed to minimize bandwidth costs. The work is based on the assumption that the bandwidth cost is positively correlated to the packet hop count. However, there was a lack of discussions on content redirection mechanisms. Compared to their works, we include more specific cooperation mechanisms by exploring the three-level cache hierarchy with our proposed request routing scheme. Moreover, we consider heterogenous settings of user demands and link capacities. Applegate *et al.* [14] formulated collaborative caching as a global optimization problem, which can take hours to be solved. Compared to their works, we propose a lightweight collaborative caching mechanism that is specifically designed for cache cooperation in practical deployments, taking advantage of specific characteristics of the cache topology.

III. Collaborative Caching Mechanism

In this section, we first analyze the potential of cache collaboration by presenting basic properties of the system topology, and then propose our challenges and a practical decomposition of our problem.

A. Hierarchical Cache Topologies

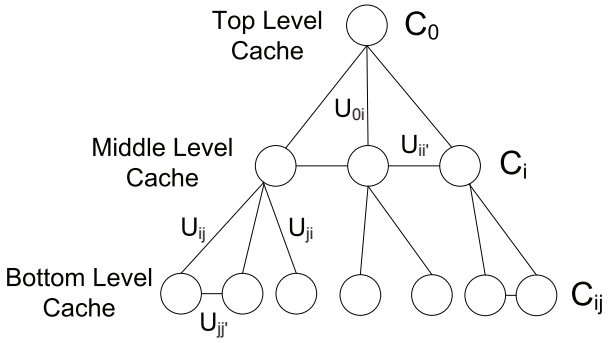

Fig. 1. The topology of a hierarchical caching network.

Hierarchical topologies have been applied in many existing systems that provide public IPTV or massive content delivery services. Fig. 1 represents a typical three-level cache topology

derived from a real-world IPTV system deployed by China Telecom in Guangzhou. As indicated by Fig. 1, we can categorize the set of cache servers into three groups, located at the top level, middle level, and bottom level, and represented by $\{C_0, C_i, C_{ij}\}$, respectively. Middle level servers C_i, each with the storage capacity B_i, denotes the ith child of a top level server C_0. A cache server C_{ij}, with its storage capacity B_{ij}, indicates the jth child of cache server C_i.

Each content request, issued by clients, attempts to access a certain content $k \in K$, with its content size s_k. The content can be considered as video segments in IPTV systems or file blocks in file downloading systems. The storage decision $X^k = \{0, 1\}$ is used to denote whether or not content k is placed at the cache server. Content requests are first directed to the corresponding bottom level server C_{ij} based on the location of requests. After receiving requests, C_{ij} either returns the corresponding contents if they are locally available, or routes requests to other cache servers in the hierarchy if contents are unavailable. Content requests are possibly routed to directly-connected bottom level servers or the parent server C_i. Similar actions are taken at the middle level server that either returns the corresponding content, or routes them to other cache servers through dynamic request routing. Requests can be directed to other sibling nodes of C_{ij} to satisfy the downstream content requests, or to neighboring servers at the middle level. A "last resort" option is to route requests to the top level server C_0, which is the source content server that stores all available contents in the entire network. Content requests will eventually be served or rejected given the constraints on content availability and link capacities.

The links that connect different cache servers in the hierarchy have heterogenous capacities. The link capacity between a pair of nodes at the bottom level is represented by $U_{jj'}$. The capacity of the link from the bottom level server to the middle level server is denoted by U_{ji}. The reverse direction from the middle to the bottom level has a link capacity U_{ij}, which is considered to be sufficiently large in the system. The link capacity between a pair of nodes at the middle level is $U_{ii'}$. Finally, The link capacity from the source server to middle level servers is indicated by U_{0i}.

B. Problem Definition and Decomposition

The problem we investigate in this paper is a joint problem of cache content placement and a corresponding dynamic request routing scheme. Storage and routing decisions are made based on user request patterns, heterogenous cache sizes, link capacities and the specific system topology. Our objective is to explore the capacity of the existing system infrastructure by maximizing the amount of supported requests.

Intuitively, shorter paths of data packets result in less traffic in the network backbone. To maximize the amount of supported requests, it is more favourable to replicate popular contents at each of the bottom level servers. However, this strategy is questionable when we introduce the cooperation among cache nodes through dynamic request routing in the cache hierarchy. Moreover, the problem becomes even more complicated when we consider the bandwidth consumed during the delivery of massive contents in IPTV or file downloading systems, which is especially important when we deal with heterogeneous link capacities and user demands in real-world systems. It has been shown in previous work [15] that the content placement problem in the cache hierarchy is NP-hard even without considering dynamic routing schemes and link capacity constraints.

To address this challenge, we propose an appropriate decomposition of the problem to find an efficient yet practical solution. Based on the topology derived from the system, we divide the original problem into sub-problems that focus on the cooperation in different cache levels. We first discuss the cooperation among interconnected servers at the bottom level. This is derived from the practical scenario where bottom level servers co-located in the same location are connected by high capacity links. The cooperation among bottom level servers connected via the middle level cache will then be investigated. Such a form of cooperation is achieved through the unique downstream content retrieval with dynamic routing decisions. The third step involves the cooperation among middle level cache servers. Content requests received at the middle level can be directed to appropriate neighboring servers through the proposed mechanism. We will derive practical content placement and dynamic request routing schemes in subsequent discussions.

IV. Cache Content Placement and Request Routing

In this section, we present our mathematical model that aim to address all the aforementioned challenges. Each level of cache cooperation is analyzed with the topological properties, and the corresponding collaborative caching scheme is also proposed.

A. Direct Bottom Level Cooperation

We define the average arrival rate of content requests at bottom level server C_{ij} as λ_{ij}. The proportion of requests that attempt to access content k is represented by p_k, and we assume that such a distribution pattern is uniform for all fully connected servers co-located in the same location. For convenience, a request to content k is said to be a type-k request. Thus the rate of type-k requests at server C_{ij} is represented by $\lambda_{ij}^k = \lambda_{ij} p_k$. Combined with concerns on the segment size, we treat $\lambda_{ij}^k s_k$ as the bandwidth requirement of type-k requests in one time unit.

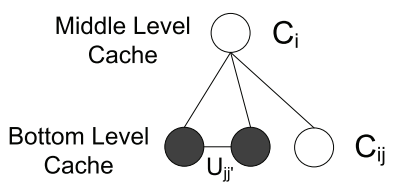

Fig. 2. Direct Bottom Level Cooperation.

As denoted by Fig. 2, fully interconnected cache servers at the bottom level are able to disseminate contents via direct links. In the deployed system that we observe, these servers are usually placed in the same equipment room while interconnecting links are typically with high bandwidth capacities. We then investigate the sub-problem that aims to maximize the amount of supported traffic at this level by utilizing the capacity of direct links. We use $J = \{j, j', j''...\}$ to represent the interconnected cache set $\{C_{ij}, C_{ij'}, C_{ij''}...\}$. The notation $y^k_{jj'}$ is used to represent the fraction of type-k requests at C_{ij} being directed to $C_{ij'}$, which actually indicates the routing strategy of interconnected cache servers at the bottom level. We then formulate the supported traffic maximization problem as:

$$\begin{aligned}
\text{Maximize} \quad & \sum_{j\in J}\sum_{j'\in J}\sum_{k\in K}\lambda^k_{ij}s_k y^k_{jj'} + \sum_{j\in J}\sum_{k\in K}\lambda^k_{ij}s_k X^k_{ij} \\
\text{Subject to:} \quad & \sum_{k\in K} s_k X^k_{ij} \le B_{ij} \quad \forall j \in J \\
& \sum_{k\in K}\lambda^k_{ij}s_k y^k_{jj'} \le U_{jj'} \quad \forall j, j' \in J \\
& X^k_{ij} + \sum_{j'\in J} y^k_{jj'} \le 1 \quad \forall j \in J, k \in K \\
& 0 \le y^k_{jj'} \le X^k_{ij'} \quad \forall j, j' \in J, k \in K \\
& X^k_{ij} = \{0, 1\} \quad \forall j \in J, k \in K
\end{aligned} \tag{1}$$

Constraints in problem (1) include the storage capacity constraint, the link capacity constraint, and the relation between content storage and the request forwarding decision. It is always optimal to serve a request by the first encountered server with the requested content. Thus, we have $X^k_{ij} + \sum_{j'\in J} y^k_{jj'} \le 1$. This guarantees that the request will not be directed to other servers if the content is locally available. Without loss of generality, we assume that the link capacity $U_{jj'}$ is consistently larger than $\sum_{k\in K}\lambda^k_{ij}s_k$, $\forall j \in J$. The assumption is practical based on our observations in the real-world system. This leads to the following theorem, which is used in cache cooperation of interconnected bottom level servers.

Theorem 1: Given $\sum_{j\in J} B_{ij} \le \sum_{k\in K} s_k$, for any content k such that $X^k_{ij} = 1$, we have $X^k_{ij'} = 0$, $\forall j' \ne j$.

Proof: Suppose in the optimal solution of problem (1), we have $X^k_{ij} = X^k_{ij'} = 1$ and $j \ne j'$. With the assumption that $U_{jj'} \ge \sum_{k\in K}\lambda^k_{ij}s_k$, the constraint $\sum_{k\in K}\lambda^k_{ij}s_k y^k_{jj'} \le U_{jj'}$ can be removed from the problem. We change the value of $X^k_{ij'}$ to 0 and the corresponding value of $y^k_{j'j}$ to 1, which generates an assignment no worse than the optimal solution. Given $\sum_{j\in J} B_{ij} \le \sum_{k\in K} s_k$, we set the value of $X^{k'}_{ij'}$ from 0 to 1 for certain content k' such that $X^{k'}_{ij} = 0$ $\forall j \in J$. The routing decision is made as $y^{k'}_{jj'} = 1$ $\forall j \ne j'$, which yields an assignment that surpasses the optimal solution with $\sum_{j\in J}\lambda^{k'}_{ij}s_{k'}$. This is a contradiction. ■

Theorem 1 suggests that all duplicated contents in the interconnected server set need to be removed and replaced by other unavailable contents. Moreover, all directly connected servers can be treated as one single server with aggregated incoming requests, storage capacities and bandwidth capacities in subsequent steps of our analysis. There is no need to consider the strategic placement of contents with such a combination, since the traffic can be easily routed through direct links at the bottom level.

B. Indirect Bottom Level Cooperation

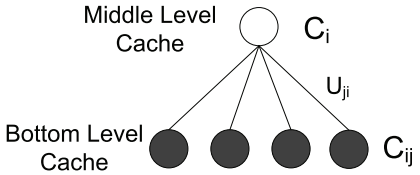

Fig. 3. Indirect Bottom Level Cooperation.

If the content is available at the bottom level, the request could be immediately served by the attached cache server or the interconnected server set. When the request is not fulfilled, it will then be routed to the corresponding upper level server. To maximize the overall amount of supported requests, we now proceed to explore the potential of indirect cache cooperation among bottom level servers as shown in Fig. 3. The main idea is to achieve downstream content retrieval through a path of the tree rooted at the parent middle level server. There exists a number of challenges in finding appropriate caching mechanisms in this form of cooperation. First, cache servers are not placed in the same place as compared to the interconnected server set at the bottom level. Therefore, user demands received by cache servers are also heterogeneous, which can be represented by different access patterns p^k_{ij}. In addition, the caching decision becomes more complicated owing to the bandwidth capacity constraints U_{ji} and the storage capacity constraints B_{ij}, which are usually asymmetric in practical systems.

In this step, we use $J = \{j, j', j''...\}$ to represent the child server set of C_i as $\{C_{ij}, C_{ij'}, C_{ij''}...\}$. These child servers are connected via the middle level server C_i. Since content delivery among nodes are sharing uplinks of bottom level servers, it would be challenging to keep track of the status of the pairwise content delivery. We instead define the proportion of the uplink of C_{ij} that is dedicated to serve type-k requests as q^k_{ij}. The request routing decision is then made at C_i based on the value of q^k_{ij}. For the bottom level cache C_{ij}, the total traffic handled locally is given by $\sum_{k\in K} X^k_{ij}s_k\lambda^k_{ij}$, in which $\lambda^k_{ij} = \lambda_{ij}p^k_{ij}$. The amount of requests supported by other bottom level servers constitutes another part of the optimization objective. We then formulate the indirect bottom level cooperation problem as:

$$\text{Maximize} \quad \sum_{j\in J}\sum_{k\in K} q_{ij}^k U_{ji} + \sum_{j\in J}\sum_{k\in K} \lambda_{ij}^k s_k X_{ij}^k \tag{2}$$

$$\text{Subject to:} \quad \sum_{k\in K} s_k X_{ij}^k \le B_{ij} \quad \forall j \in J \tag{3}$$

$$\sum_{j\in J} q_{ij}^k U_{ji} \le \sum_{j\in J} \lambda_{ij}^k s_k (1 - X_{ij}^k) \quad \forall k \in K \tag{4}$$

$$\sum_{k\in K} q_{ij}^k \le 1 \quad \forall j \in J \tag{5}$$

$$0 \le q_{ij}^k \le X_{ij}^k \quad \forall j \in J, k \in K \tag{6}$$

$$X_{ij}^k = \{0,1\} \quad \forall j \in J, k \in K \tag{7}$$

To achieve a distributed solution with heterogeneous settings of input parameters, a conventional method is to apply Lagrangian relaxation. Constraints (4) and (6) can be incorporated into the objective function by associating a Lagrangian multiplier with each constraint. η_k is associated with constraint (4) and γ_{ij}^k is associated with constraint (6). Then the Lagrangian dual problem is represented as:

$$\begin{aligned} \text{Minimize} \quad & L(\eta_k, \gamma_{ij}^k) \\ \text{Subject to:} \quad & \eta_k \ge 0, \gamma_{ij}^k \ge 0 \end{aligned} \tag{8}$$

The objective function $L(\eta_k, \gamma_{ij}^k)$ in the dual problem is:

$$\begin{aligned} & L(\eta_k, \gamma_{ij}^k) \\ & = \max \sum_{j\in J}\sum_{k\in K} (U_{ji} - \eta_k U_{ji} - \gamma_{ij}^k) q_{ij}^k \\ & \quad + \sum_{j\in J}\sum_{k\in K} (\eta_k \lambda_{ij}^k s_k + (\gamma_{ij}^k + (1-\eta_k)\lambda_{ij}^k s_k) X_{ij}^k) \end{aligned}$$

The Lagrangian subproblem can then be decomposed into $|J|$ storage allocation problems and $|J|$ link allocation problems. Both storage and link allocation problems can be solved in a distributed manner at each bottom level cache server. The Lagrangian multipliers are then updated at each iteration with the coordination of the middle level server C_i, in order to find the optimal solution through the subgradient algorithm. However, our simulation studies show that our algorithm above requires a long time to converge, especially when there exists a large number of objects. Inspired by the Lagrangian relaxation method, we propose a heuristic algorithm by limiting the decision of q_{ij}^k at each iteration. The proposed algorithm can achieve much faster convergence with a near-optimal performance.

We decompose the Lagrangian multiplier η_k into coefficients η_{ij}^k to achieve a finer granularity in controlling the storage and bandwidth allocation decisions at each iteration. η_{ij}^k reflects the interest of storing a particular content and the willingness of sharing stored content. We then omit γ_{ij}^k and put constraint (6) into the optimization problem. At each iteration t, cache server C_{ij} first solves the storage allocation problem as the following:

$$\begin{aligned} \text{Maximize} \quad & \sum_{k\in K} (1-\eta_{ij}^k)\lambda_{ij}^k s_k X_{ij}^k \\ \text{Subject to:} \quad & \sum_{k\in K} s_k X_{ij}^k \le B_{ij} \\ & X_{ij}^k = \{0,1\} \quad \forall k \in K \end{aligned} \tag{9}$$

In a practical system setting, contents can be divided into segments with equal sizes for convenience. Therefore, the optimal solution of the storage allocation problem is given as:

$$X_{ij}^k(t) = \begin{cases} 1, & \text{for } k \in [1, z); \\ 0, & \text{for } k \in [z, |K|] \end{cases} \tag{10}$$

In solution (10), the content set K is sorted in descending order by the critical index $(1-\eta_{ij}^k)\lambda_{ij}^k$, and $z = \min\{h : \sum_{k=1}^h s_k > B_{ij}\}$. Given the storage decision at each bottom level server, the middle level server C_i acts as a master node to collect the temporal information from C_{ij}. It calculates the content demand $Q_k(t)$ for k at the current iteration as the following, which will then be distributed to bottom level servers:

$$Q_k(t) = \sum_{j\in J} \lambda_{ij}^k s_k (1 - X_{ij}^k(t)) \quad \forall k \in K \tag{11}$$

Then each cache server C_{ij} solves the bandwidth allocation problem individually such that:

$$\begin{aligned} \text{Maximize} \quad & \sum_{k\in K} (U_{ji} - \eta_{ij}^k U_{ji}) q_{ij}^k \\ \text{Subject to:} \quad & \sum_{k\in K} q_{ij}^k \le 1 \\ & q_{ij}^k U_{ji} \le Q_k(t) \quad \forall k \in K \\ & 0 \le q_{ij}^k \le X_{ij}^k(t) \quad \forall k \in K \end{aligned} \tag{12}$$

The optimal bandwidth allocation in the collaboration can be achieved as:

$$q_{ij}^k(t) = \begin{cases} \frac{Q_k(t)}{U_{ji}} & \text{for } k \in [1, z) \text{ and } X_{ij}^k(t) = 1; \\ \frac{U_{ji} - \sum_{k=1}^{z-1} Q_k(t)}{U_{ji}} & \text{for } k = z; \\ 0 & \text{for } k \in (z, |K|] \text{ or } X_{ij}^k(t) = 0 \end{cases} \tag{13}$$

In solution (13), the content set K is sorted in descending order by the critical index $(1-\eta_{ij}^k)$, and $z = \min\{h : \sum_{k=1}^h Q_k(t) X_{ij}^k(t) > U_{ji}\}$.

The parent cache server C_i is then responsible for gathering temporal results from bottom level servers and updating the value of η_{ij}^k for the next iteration:

$$\eta_{ij}^k(t+1) = \eta_{ij}^k(t) + \theta(t)(\sum_{j\in J} q_{ij}^k(t) U_{ji} - Q_k(t)) * f(\sum_{k\in K} q_{ij}^k(t)) \tag{14}$$

The insufficient provisioning of bandwidth in the collaboration for certain content k leads to a decreasing value of η_{ij}^k, which then inspires servers to store and share the corresponding content. The function $f()$ is positively correlated to the remaining bandwidth of current allocation $(1-\sum_{k\in K} q_{ij}^k(t))$. $\theta(t) = 1/t$ denotes the step size in the current iteration t. Both of them ensure the quick convergence of the proposed heuristic algorithm.

The entire process of indirect bottom level cooperation is then summarized in Algorithm 1.

Algorithm 1 Indirect Bottom Level Cooperation

1) Initialize coefficients $\eta_{ij}^k(0) = 0, \forall j \in J, k \in K$.
2) Iterate until coefficients η_{ij}^k converge to ${\eta_{ij}^k}^*$:
   a) Calculate $X_{ij}^k(t)$ according to Equation (10), $\forall j \in J, k \in K$.
   b) Calculate demands $Q_k(t)$ at C_i.
   c) Calculate $q_{ij}^k(t)$ according to Equation (13), $\forall j \in J, k \in K$.
   d) Update coefficients η_{ij}^k according to Equation (14), $\forall j \in J, k \in K$.
3) Obtain the near-optimal solution as ${q_{ij}^k}^* = q_{ij}^k(t)$ and ${X_{ij}^k}^* = X_{ij}^k(t)$.

Based on the previous results of ${q_{ij}^k}^*$ and ${X_{ij}^k}^*$, we then define the routing map $\mathbf{R_b} = [R_{bj}^1, R_{bj}^2, ..., R_{bj}^k, ...]$ for the dynamic request routing in the indirect bottom level cooperation at C_i. The probability of routing a request for content k to C_{ij} is given as:

$$R_{bj}^k = \frac{{q_{ij}^k}^* U_{ji}}{\sum_{j\in J} \lambda_{ij}^k s_k (1 - {X_{ij}^k}^*)} \tag{15}$$

Note that the summation of the probability $\sum_{j\in J} R_{bj}^k$ might be smaller than 1 since not all requests are promised to be fulfilled in indirect bottom level cooperation even though the contents are available. Remaining requests are routed to neighboring middle level servers or the top level server with probability $(1 - \sum_{j\in J} R_{bj}^k)$.

We have now obtained a cache cooperation mechanism that fully explores the potential of downstream content retrieval. The assignment is in a distributed fashion while both asymmetric link capacities and heterogeneous user demands are considered. Note that the capacity of the middle level server is not involved in making a decision, which allows us to further explore the opportunity of middle level cache cooperation in the next subsection. Compared to cache cooperation with static routing, our exploration of indirect bottom level cooperation can be beneficial in the scenario of heterogenous user demands as servers can fulfill others' requests through downstream content retrieval, while the exploration of the middle level cooperation can be beneficial in the scenario of homogeneous user demands, as middle level servers do not have to store nearly duplicated contents with our dynamic routing scheme.

C. Middle Level Cooperation

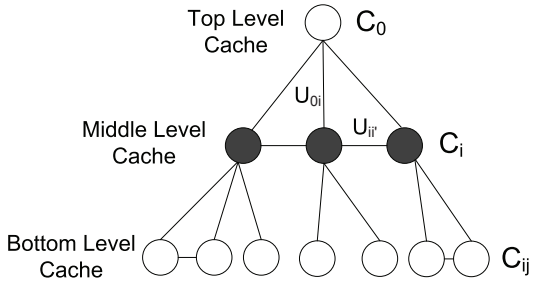

Fig. 4. Middle Level Cooperation.

Given our proposed mechanisms, the intra-links of the bottom level and the inter-links between the middle and bottom level have been utilized in cache cooperation. The amount of requests being processed at the middle level is then given below:

$$\lambda_i^k = \sum_{j\in J} \lambda_{ij}^k (1 - {X_{ij}^k}^*) - \sum_{j\in J} {q_{ij}^k}^* U_{ji}/s_k \quad \forall i \in I, k \in K \tag{16}$$

On this basis, we now proceed to explore middle level cache cooperation as indicated by Fig. 4. We observe that any intra-link at the middle level is dedicated to the content transfer between a pair of servers, called neighboring servers. To ensure a minimum delay in each data packet, we impose the restriction that requests in middle level cooperation can only be routed to neighbors of each node. The main challenge here is to find the appropriate content placement within the neighboring server set with respect to heterogeneous demands and link capacities. We use $I = \{i, i', i''...\}$ to represent the child cache servers of C_0 as $\{C_i, C_{i'}, C_{i''}...\}$. For each cache server C_i, we define its neighboring set as $N_i = \{n_{ij} | j = 1, 2, ...\}$, $N_i \subseteqq I$. Based on demands at each cache server, we then associate a cost-utility index with each content k at server C_i when considering cache content placement:

$$u_i^k = \frac{u_{il}^k + u_{ir}^k}{s_k} \quad \forall i \in I, k \in K \tag{17}$$

The utility function includes two parts as the local utility u_{il}^k and the remote utility u_{ir}^k. The local utility u_{il}^k should be positively correlated to demand λ_i^k. Meanwhile, it is also related to content availability of content k within the neighboring set N_i, which affects middle level cache cooperation. The remote utility u_{ir}^k also needs to reflect the potential of cache cooperation, which will be positively correlated to content demand rates at neighboring servers. The information above on neighboring servers can be obtained through periodical information exchanges. Furthermore, both utilities are constrained by the link capacity $U_{n_{ij}i}$ and $U_{in_{ij}}$ among middle level servers as a special characteristic of massive content distribution. The cost of placing content k at C_i is the storage

cost s_k. We then derive the utility function based on the aforementioned consideration:

$$u_{il}^k = \lambda_i^k s_k (1 - \max_{n_{ij} \in N_i} X_{n_{ij}}^k (1 - \rho^{\frac{U_{n_{ij}i}}{avg(U)}})) \tag{18}$$

$$u_{ir}^k = \sum_{n_{ij} \in N_i} \lambda_{n_{ij}}^k s_k (1 - X_{n_{ij}}^k)(1 - \rho^{\frac{U_{in_{ij}}}{avg(U)}})) \tag{19}$$

In this formulation, ρ is a tunable parameter that satisfies $0 < \rho < 1$. The smaller the value of ρ, the greater sensitivity to remote content availability will be. The notation $avg(U)$ denotes a non-negative average link capacity as $\max(\sum_{n_{ij} \in N_i} U_{n_{ij}i}/|N_i|, \epsilon > 0)$. Our definition of the utility function has addressed many concerns in our system design. We first investigate its properties using two special scenarios in link capacity settings.

Proposition 1: The caching decision of C_i is not affected by the neighboring server n_{ij} with limited link connectivity such that $U_{n_{ij}i} = U_{in_{ij}} \to 0$.

Proof: In this case, there will be no content delivery between this pair of middle level servers. From the definition of the utility we have:

$$\rho^{\frac{U_{n_{ij}i}}{avg(U)}} = \rho^{\frac{U_{in_{ij}}}{avg(U)}} = 1$$

It precludes the impact of $X_{n_{ij}}^k$ and $\lambda_{n_{ij}}^k$ from u_{ir}^k and u_{il}^k. Therefore, the caching decision of local server C_i is not affected by the status of neighboring server n_{ij}. ■

Proposition 2: When the link capacity is unlimited, no utility will be gained for duplicating contents within a pair of cache servers.

Proof: In this scenario, we have $U_{n_{ij}i} = U_{in_{ij}} \to \infty$, which suggests that the cache cooperation can be achieved as long as the content is stored in neighboring server n_{ij}. From the definition we have,

$$\rho^{\frac{U_{n_{ij}i}}{avg(U)}} = \rho^{\frac{U_{in_{ij}}}{avg(U)}} = 0$$

If the neighbor server has content k as $X_{n_{ij}}^k = 1$, we have the utility $u_{ir}^k(n_{ij}) = \lambda_{n_{ij}}^k s_k(1 - X_{n_{ij}}^k) = 0$ and $u_{il}^k = \lambda_i^k s_k(1 - \max_{n_{ij} \in N_i} X_{n_{ij}}^k) = 0$. Therefore, no utility will be gained for duplicating k at C_i for this pair of servers. ■

In other cases of link capacity settings, the utility function also conforms to the practice of cache cooperation design. The increase of link capacity $U_{n_{ij}i}$ leads to an increase with $1 - \rho^{\frac{U_{n_{ij}i}}{avg(U)}}$, which further results in a decreasing concern on local demands when the content is stored in the corresponding neighboring server. Similarly, the increase of $U_{in_{ij}}$ implies that demands from the server with a larger link capacity are given higher priorities when considering content placement. Moreover, both local and remote utilities decreased for contents that are already placed in neighboring servers, which emphasizes the requirement of content heterogeneity.

The selection of parameter ρ has also significantly affected the performance of our proposed mechanism. On one hand, a higher request heterogeneity among neighbors leads to a larger value of ρ, which satisfies both requirements of local demands and the content heterogeneity requirement among neighboring servers. On the other hand, the value of ρ should be carefully chosen in the case of homogeneous request patterns. Generally, the setting of ρ should be negatively correlated to the bandwidth provisioned, *i.e.*, positively correlated to the amount of arriving requests. This can help satisfy popular content requests without dynamic routing while maintaining a certain level of storage heterogeneity for the benefit of cache cooperation.

Based on the analysis above, we then propose the content placement strategy for middle level cache cooperation in Algorithm 2.

Algorithm 2 Content Placement in Middle Level Cooperation

1) For each content $k \in K$, associate cost-utility index u_i^k based on Equation (17).
2) Sort contents in a descending order based on u_i^k.
3) Fill in the cache storage in a greedy manner, *i.e.*, contents with higher ranks are preferentially stored and contents with lower ranks are preferentially evicted.

Dynamic routing at the middle level is based on content availability at neighboring servers and the link capacity $U_{n_{ij}i}$. Since each link is dedicated to the content transfer between a pair of servers, our routing design therefore aims to deliver appropriate types and amounts of requests to each neighboring server. We define the routing map $\mathbf{R_{n_{ij}}} = [R_{n_{ij}}^1, R_{n_{ij}}^2, ..., R_{n_{ij}}^k, ...]$, $k \in K$ for each neighbor n_{ij} and perform the logical subtraction for subsequent calculations:

$$\mathbf{R_{n_{ij}}} = \mathbf{X_{n_{ij}}} - \sum_{j' \neq j} \mathbf{X_{n_{ij'}}} - \mathbf{X_{C_i}} \quad \forall n_{ij} \in N_i \tag{20}$$

$$\mathbf{R_{n_{ij}}^m} = \mathbf{X_{n_{ij}}} - \mathbf{R_{n_{ij}}} - \mathbf{X_{C_i}} \quad \forall n_{ij} \in N_i \tag{21}$$

$\mathbf{R_{n_{ij}}}$ denotes contents that only have single copies in neighbor set N_i. Thus type-k requests are routed to n_{ij} if $R_{n_{ij}}^k = 1$. $\mathbf{R_{n_{ij}}^m}$ denotes contents that have multiple copies in neighbor set N_i, in which the dynamic request routing still needs careful consideration. To address this challenge, we further define a global availability map as:

$$\mathbf{R_g} = \sum_{n_{ij} \in N_i} \mathbf{X_{n_{ij}}} - \sum_{n_{ij} \in N_i} \mathbf{R_{n_{ij}}} - \mathbf{X_{C_i}} \tag{22}$$

We then adjust the routing map $\mathbf{R_{n_{ij}}}$ based on request rates at C_i and the comparison between the value of $U_{n_{ij}i}$ and $\sum_{k \in K} \lambda_i^k R_{n_{ij}}^k$.

- If $\sum_{k \in K} \lambda_i^k R_{n_{ij}}^k \leq U_{n_{ij}i}$, we iteratively move the content k such that $R_{n_{ij}}^k = 0$, $R_{n_{ij}}^{mk} = R_g^k = 1$ into $\mathbf{R_{n_{ij}}}$ by setting $R_{n_{ij}}^k = 1$, $R_{n_{ij}}^{mk} = R_g^k = 0$. We repeat the process until any movement of contents leads to

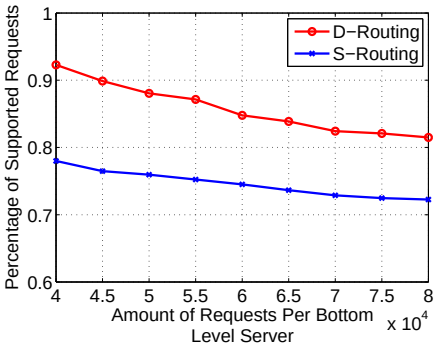

Fig. 5. The percentage of supported requests in collaborative caching versus different amounts of content requests.

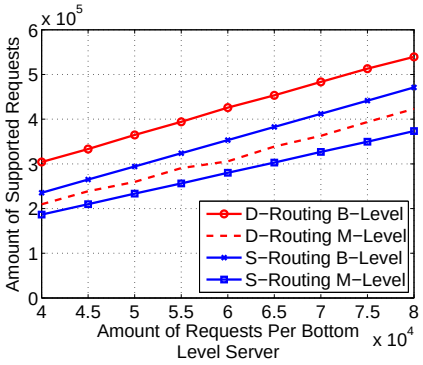

Fig. 6. The amount of supported requests in different cache levels versus different amounts of content requests.

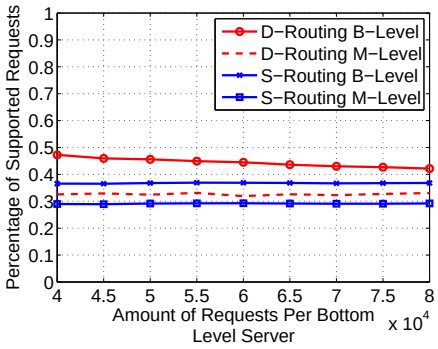

Fig. 7. The percentage of supported requests in different cache levels versus different amounts of content requests.

$\sum_{k\in K}\lambda_i^k R_{n_{ij}}^k > U_{n_{ij}i}$, or we cannot find content k such that $R_{n_{ij}}^k = 0$, $R_{n_{ij}}^{mk} = R_g^k = 1$.

- If $\sum_{k\in K}\lambda_i^k R_{n_{ij}}^k > U_{n_{ij}i}$, we iteratively remove the content such that $R_{n_{ij}}^k = 1$ from $\mathbf{R_{n_{ij}}}$ by setting $R_{n_{ij}}^k = 0$. We repeat the process until $\sum_{k\in K}\lambda_i^k R_{n_{ij}}^k \le U_{n_{ij}i}$.

After the aforementioned processes, a set of routing maps defines a near-optimal request routing scheme based on content availability and heterogenous link capacities. Type-k requests are routed to n_{ij} if $R_{n_{ij}}^k = 1$. Note that routing maps are mutually exclusive such that $\mathbf{R_{n_{ij}}} - \mathbf{R_{n_{ij'}}} = \mathbf{R_{n_{ij}}}$.

D. Dynamic Request Routing

Based on our decomposed cache cooperation design proposed in Sec. IV, we have solved the coupled problems of content placement and dynamic request routing, considering practical concerns in massive content distribution. For any type-k request, we summarize our dynamic request routing scheme in Algorithm 3.

Algorithm 3 Dynamic Request Routing

1) After receiving a type-k request at bottom level server C_{ij}, fulfill the request if $X_{ij}^k = 1$.
2) If not yet fulfilled, check the content availability of the interconnected server set, route the request to the corresponding server if the content is available.
3) If not yet fulfilled, route the request to the middle level server C_i, fulfill the request if $X_i^k = 1$.
4) If not yet fulfilled, check the following three options:
   a) Check the routing map $\mathbf{R_b}$. Route the request for content k to bottom level server C_{ij} with probability R_{bj}^k according to Equation (15).
   b) Otherwise, check the routing maps $\mathbf{R_{n_{ij}}}$ for each n_{ij}. If $R_{n_{ij}}^k = 1$ for some n_{ij}, route the request to the neighboring server n_{ij}.
   c) As a last resort, route the request to the top level server.

V. Performance Evaluation

In this section, we evaluate our cache cooperation scheme using system settings derived from a real-world IPTV system, in comparison with a conventional static hierarchical caching scheme. Unlike our proposed dynamic request routing, static hierarchical caching usually defines fixed routing paths from the bottom level to the top level, and makes collaborative caching decisions correspondingly. Therefore, the most popular contents are replicated at the lower level while less popular contents are placed at the upper level.

We use Python to implement a simulator that constructs the cache topology as a three-level hierarchy, in which each top and middle level server has four children servers. We use the Zipf-Mandelbrot model [16] to formulate the content request pattern with the shape parameter $0.7 \le \alpha \le 0.8$ and $4 \le q \le 5$. The average link capacity of U_{ji}, $U_{ii'}$, and U_{0i} is set to 1GB/s, 2GB/s and 4GB/s initially based on the real-world system. The storage capacity of top level, middle level, and bottom level servers are given as 100%, 20% and 5% of the entire content set. The parameter ρ is adjusted between 0.5 and 0.8 to adapt to different settings of the system.

We first compare the cache cooperation performance between dynamic (denoted as D-Routing) and static request routing (denoted as S-Routing) in Fig. 5. It can be observed that the percentage of supported requests of D-Routing consistently outperforms that of S-Routing with 13% to 18% improvements. The performance gap is achieved through downstream content retrieval or the middle level cooperation in D-Routing. However, the gap is gradually decreasing with an increasing number of content requests, as long as link capacities in D-Routing have been fully utilized.

We then analyze the amount of supported traffic in different levels of cache cooperation in Fig. 6. The traffic supported by the bottom level is calculated as the summation of requests directly satisfied at the bottom level and the contribution from indirect bottom level cooperation. The traffic supported by the middle level is a combination of direct content transfer from middle level servers and the support from middle level cache cooperation. It is not surprising that both middle level and bottom level of S-Routing fulfill less demands compared to that of D-Routing. Meanwhile, bottom level servers satisfy much more demands than middle level servers in D-Routing, which can be explained as most content requests are already served in D-Routing through downstream content retrieval.

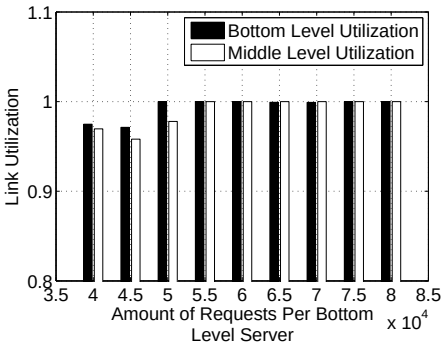

Fig. 8. The link utilization in D-Routing versus different amounts of content requests.

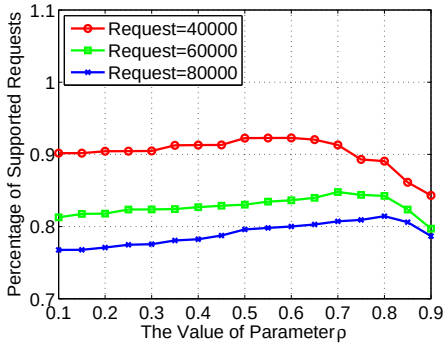

Fig. 9. The percentage of supported requests in D-Routing with different values of parameter ρ.

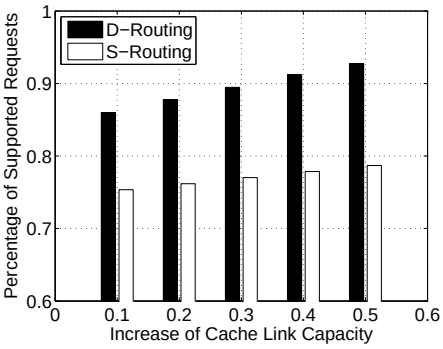

Fig. 10. The percentage of supported requests in collaborative caching with different increases of cache link capacities.

If we further investigate the percentage of supported traffic in different levels in Fig. 7, we find that both middle level and bottom level contribution in D-Routing can maintain a stable performance with different amounts of incoming requests, which clearly reveals the stability of our proposed collaborative caching mechanism.

The utilization of intra-links at the middle level and inter-links between the middle and bottom level reflect the effectiveness of the proposed mechanism with D-Routing. Fig. 8 shows that both middle level and bottom level cooperation are sufficiently good initially with over 90% utilization. When demands increase over the entire system, larger amounts of requests received at different levels lead to the full utilization of link capacities for both downstream content retrieval and middle level cache cooperation.

We then investigate the impact of parameter ρ in Fig. 9. The amount of requests per bottom level server is adjusted from 40000 to 80000. The increasing amount of requests also reflects a decreasing trend of the bandwidth provisioned in the system. We first observe that the percentage of fulfilled requests is gradually decreasing with an increasing input scale. Meanwhile, the optimal value of ρ is also "shifting," such that a larger amount of requests (80000) achieves the optimal result when $\rho = 0.8$ while a smaller amount of requests (40000) achieves the optimal value with $\rho = 0.6$. This practically proves our previous analysis that the selection of ρ should be negatively correlated to the bandwidth provisioned, in order to maintain a balance between the requirement of local demands and the adequate level of content heterogeneity among neighboring servers at the middle level.

Fig. 10 introduces the sensitivity analysis on cache capacity settings by adjusting the link capacity in the system. Although the performance of S-Routing also has nearly 1% improvement for each 10% of extra link capacities, the performance gap between S-Routing and D-Routing is still gradually increasing. The rationale is that the proposed D-Routing can successively explore the maximum potential of cache cooperation with expanded link capacities.

VI. Conclusion

In this paper, we have proposed a new collaborative caching mechanism based on the topology derived from a real-world system that provides IPTV services in a metropolitan network. In particular, we have focused on the exploration of the existing network infrastructure to better support new scenarios of massive content distribution. We have shown that dynamic request routing helps explore the capacity of a caching network. Our simulation results have demonstrated the effectiveness of our proposed mechanism, as compared to conventional static routing schemes.

References

[1] "Ubuntu Mirrors," https://wiki.ubuntu.com/Mirrors/.

[2] "China Telecom," http://en.chinatelecom.com.cn/.

[3] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Placement Algorithms for Hierarchical Cooperative Caching," in *Proc. ACM SODA*, Jan. 1999.

[4] L. Chen, M. Meo, and A. Scicchitano, "Caching Video Contents in IPTV Systems with Hierarchical Architecture," in *Proc. IEEE ICC*, Jun. 2009.

[5] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, "On the Scale and Performance of Cooperative Web Proxy Caching," in *Proc. ACM SOSP*, Dec. 1999.

[6] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of Web Caching Architectures Hierarchical and Distributed Caching," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, 2001.

[7] W. Li, E. Chan, G. Feng, D. Chen, and S. Lu, "Analysis and Performance Study for Coordinated Hierarchical Cache Placement Strategies," *Computer Communications*, vol. 33, no. 15, 2010.

[8] K. Li, H. Shen, F. Y. L. Chin, and S. Q. Zheng, "Optimal Methods for Coordinated Enroute Web Caching for Tree Networks," *ACM Transactions on Internet Technology*, vol. 5, no. 3, 2005.

[9] X. Jia, D. Li, H. Du, and J. Cao, "On Optimal Replication of Data Object at Hierarchical and Transparent Web Proxies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 8, 2005.

[10] L. Wang, V. Pai, and L. Peterson, "The Effectiveness of Request Redirection on CDN Robustness," in *Proc. USENIX OSDI*, Dec. 2002.

[11] N. Laoutaris, G. Zervas, A. Bestavros, and G. Kollios, "The Cache Inference Problem and its Application to Content and Request Routing," in *Proc. IEEE INFOCOM*, May 2007.

[12] T. Bektas, J.-F. Cordeau, E. Erkut, and G. Laporte, "Exact Algorithms for the Joint Object Placement and Request Routing Problem in Content Distribution Networks," *Computers and Operations Research*, vol. 35, no. 12, 2008.

[13] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," in *Proc. IEEE INFOCOM*, Mar. 2010.

[14] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal Content Placement for a Large-Scale VoD System," in *Proc. ACM CoNext*, Nov. 2010.

[15] I. Baev, R. Rajaraman, and C. Swamy, "Approximation Algorithms for Data Placement Problems," *SIAM J. on Computing*, vol. 38, no. 4, 2008.

[16] M. Hefeeda and O. Saleh, "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, 2008.