On Replica Placement for QoS-Aware Content Distribution

Xueyan Tang
School of Computer Engineering
Nanyang Technological University
Singapore 639798
asxytang@ntu.edu.sg

Jianliang Xu
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
xujl@comp.hkbu.edu.hk

***Abstract*— The rapid growth of time-critical information services and business-oriented applications is making quality of service (QoS) support increasingly important in content distribution. This paper investigates the problem of placing object replicas (e.g., web pages and images) to meet the QoS requirements of clients with the objective of minimizing the replication cost. We consider two classes of service models: replica-aware service and replica-blind service. In the replica-aware model, the servers are aware of the locations of replicas and can therefore direct requests to the nearest replica. We show that the QoS-aware placement problem for replica-aware services is NP-complete. Several heuristic algorithms for efficient computation of suboptimal solutions are proposed and experimentally evaluated. In the replica-blind model, the servers are not aware of the locations of replicas or even their existence. As a result, each replica only serves the requests flowing through it under some given routing strategy. We show that there exist polynomial optimal solutions to the QoS-aware placement problem for replica-blind services. Efficient algorithms are proposed to compute the optimal locations of replicas under different cost models.**

I. Introduction

Replication techniques are widely employed to improve the performance of large-scale content distribution systems such as CDNs [1]. By geographically multiplying the source of information, the requested contents can be brought much closer to the clients, thereby reducing both the access latency and network traffic. Replication also offers the potential to improve system scalability by distributing the load across multiple servers.

In general, a client would experience shorter access latency if a replica of the requested object (e.g., a web page or an image) is placed in its closer proximity. Therefore, the effectiveness of replication, to a large extent, depends on the locations where the replicas are placed. With the rapid growth of time-critical information services and business-oriented applications, there is an increasing demand to support quality of service (QoS) in content distribution [2]. The desired level of performance can be specified in the form of service level agreements (SLAs) between the content/service providers and their users, e.g., the response time of requests from domain A for Nasdaq.com home should not exceed 1 second; 95% of the requests from domain B for CNN.com home should complete in less than 2 seconds [3]. This entails the consideration of QoS requirements in replica placement. It is desirable to allocate replicas in the network closer to the clients with higher QoS requirements. Unfortunately, most existing work on replica placement has focused on optimizing an average performance measure of the entire client community such as the mean access latency [4], [5], [6]. While an average performance measure may be important from the system's point of view, it does not differentiate the likely diverse performance requirements of the individuals. So far, to the best of our knowledge, there has been no study on QoS-aware replica placement.

In this paper, we investigate the problem of placing the replicas of an object in content distribution systems to meet the QoS requirements of clients with the objective of minimizing the replication cost. The QoS requirements are specified in the form of a general distance metric. The replication cost, on the other hand, is measured in terms of storage, consistency management, or a combination of both. We consider two classes of service models that lead to different problem formulations: replica-aware service and replica-blind service.

In the replica-aware model, the servers in the system are aware of the locations of replicas. By making use of this information, the servers are capable of directing requests to the nearest replica of the target object. We show that the QoS-aware placement problem for replica-aware services is NP-complete. Several heuristic algorithms are then proposed for efficient computation of suboptimal solutions. They are evaluated, via simulation experiments, against a super-optimal bound obtained from the solution of a relaxed linear program.

In the replica-blind model, the servers in the system are not aware of the locations of replicas or even their existence. As a result, request routing is independent of where the replicas of the target object are located. Each replica only serves the requests flowing through it under some given routing strategy which can be implemented at either the network level or the application level. We show that the QoS-aware placement problem for replica-blind services can be solved with polynomial time complexity. Efficient algorithms are proposed to compute the optimal locations of replicas under different cost models.

The rest of this paper is organized as follows. Section II summarizes the related work. Section III describes the system model and provides some basic definitions. Section IV presents

a formulation of the QoS-aware placement problem for replica-aware services, analyzes its complexity and proposes several heuristic solutions. The QoS-aware placement problem for replica-blind services is formulated and investigated in Section V. Finally, Section VI concludes the paper.

II. Related Work

Early work on replica placement had investigated the file allocation problem (FAP) in storage systems [7] and the database location problem (DBL) in computer networks [8]. They were transformed into mixed linear programming models. In these studies, the delivery of data updates to different replicas was assumed unicast-based. Wolfson *et al.* [9] adopted a multicast-based delivery model to reduce the network traffic of update transfers. They proposed polynomial-time algorithms to compute optimal replica placement strategies for some special networks. However, their problem formulation assumed a homogeneous network model where all the links were associated with the same communication cost. Furthermore, the storage costs of replicas were not considered in the cost model.

Recent research has studied replica placement on the Internet for efficient content distribution. The replication entity can be either a mirror/proxy server or an object replica. The former is referred to as the server placement problem and the latter is called the object placement problem. Most existing work on server placement has assumed all mirror/proxy servers are provided with the same contents, in which case the server placement problem is essentially the same as the object placement problem. Li *et al.* [10] and Krishnan *et al.* [4] developed polynomial optimal solutions to place a given number of servers in a tree network to minimize the average retrieval cost of all clients. The same problem for general topologies was shown to be NP-complete. Qiu *et al.* [5] experimentally compared several heuristic solutions and found a simple greedy algorithm performed the best. Jamin *et al.* [11] investigated the constrained mirror placement problem where mirrors were allowed to be placed at some subset of network nodes only. It was shown that placing more mirrors beyond a certain number offered little performance gain. Different from the above studies which explored the optimization of retrieval cost only, Xu *et al.* [12] and Jia *et al.* [13] further took the update cost into consideration. Cidon *et al.* [6], on the other hand, used the total storage and retrieval cost as the objective metric of optimization. They developed a distributed algorithm to compute the optimal locations to place object replicas. A more comprehensive cost model was adopted by Kalpakis *et al.* [14] who optimized three types of cost (i.e., retrieval, update, and storage costs) for replica placement in an integrated fashion. Korupolu *et al.* [15] took a different approach to account for the storage overhead. They investigated replica placement for multiple objects in the context of hierarchical caching under the constraint that each cache was equipped with limited storage space. A coordinated placement and replacement strategy for general cascaded caching architectures was proposed by Tang *et al.* [16]. However, most work described above aimed at maximizing the performance gain with a given budget of resources in terms of an average performance measure. To the best of our knowledge, none of the existing work has considered providing some level of performance guarantee in replica placement. Different from existing research, the objective of our study is to minimize the amount of resources required to achieve a certain level of service. We investigate the QoS-aware replica placement problem for a variety of service and cost models.

III. System Model and Basic Definitions

Consider an object hosted by a content distribution system whose servers are connected to form a network represented by a graph $G = (V, E)$, where V is the set of servers and $E \subset V \times V$ is the set of physical or logical links between the servers. A weight $s(v)$ is associated with each server $v \in V$, representing the cost of storing a copy of the object at v. Moreover, a distance $d(u, v)$ is associated with each edge $(u, v) \in E$, representing the communication cost of sending a request and the associated response for the object between u and v. Note that the term "communication cost" is used in a general sense in our model. It can be intepreted as different performance measures such as network delay, bandwidth consumption, and hop count. If an object transfer goes through multiple links from the source to the destination, the total communication cost is given by the sum of those on all intermediate links. To facilitate presentation, we shall extend the function $d(u, v)$ to all pairs of nodes $(u, v) \in V \times V$ by defining $d(u, v)$ as the total communication cost of the links on the shortest path between u and v.

The object is associated with an authoritative *origin server* in the network where the content provider makes the updates to the object. The object copy located at the origin server is called the *origin copy* (denoted by r) and an object copy at any remaining server is called a *replica*. We refer to the set of servers in $V - \{r\}$ where the replicas are placed as the *replication strategy* (denoted by R). The object is retrieved by the clients outside the network of servers. We assume each server receives requests from some community of clients (e.g., by statically configuring the clients, using DNS-based request direction, or intercepting requests in a transparent fashion [1]). If the object is replicated at the server receiving the client request, the response is generated locally. Otherwise, the server forwards the request to the other servers in the network and relays the response to the client.

In this paper, we consider *retrieval cost* as a measure of the performance perceived by the clients. The retrieval cost of a request is given by the communication cost involved in serving the request. Since the communication cost from a client to the associated server is independent of the replication strategy, for simplicity, this portion of cost is not included in our analytical model. We shall assume client requests originate from the associated servers. Every server in the network has some QoS requirement on retrieving the object for its clients. The QoS requirement of each server v is specified by an upperbound $q(v)$ on retrieval cost. If the object can be retrieved by v within a cost of $q(v)$, the QoS requirement is *satisfied*. Otherwise, the

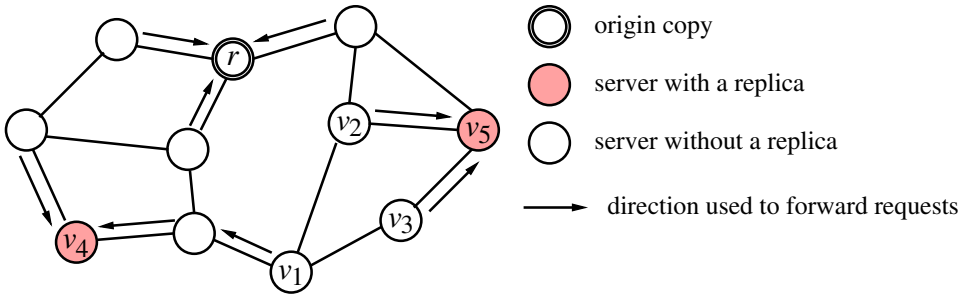


Fig. 1. Replica-Aware Service Model

QoS requirement is *violated*. The QoS requirements associated with different servers can be different.

The objective of the QoS-aware replica placement problem is to find a replication strategy of the object that satisfies the QoS requirements of all servers and involves the minimal replication cost. We identify two types of replication cost: *storage cost* and *update cost*. Given a replication strategy $R \subseteq V - \{r\}$, the storage cost of R refers to the cost of placing replicas at the servers in R and is given by

$$scost(R) = \sum_{v \in R} s(v).$$

The update cost, on the other hand, refers to the communication cost of keeping the replicas consistent with the authoritative origin copy. To allow for efficient delivery of object updates, it is assumed that all servers in the network are organized into a tree structure rooted at the origin server. We shall call it the *update distribution tree*, denoted by T. Object updates are delivered from the origin copy to all replicas via application-level multicast, in which each server receives updates from its parent and is responsible for further distributing the updates to its children [17], [18]. The total cost of update delivery depends on the locations of the lowest level replicas in the tree. Let μ be the update rate of the object. The update cost of R is then given by

$$ucost(R) = \mu \cdot \sum_{v \neq r \ \wedge \ T_v \cap R \neq \emptyset} d(v, p(v)),$$

where T_v is the subtree of T rooted at v, and $p(v)$ is the parent of v in T. Note that $T_v \cap R \neq \emptyset$ implies v is involved in the application-level multicast. Depending on the business model, the replication cost of R can take the form of the storage cost $scost(R)$, the update cost $ucost(R)$, or a combination of the two costs $\alpha \cdot scost(R) + (1 - \alpha) \cdot ucost(R)$, where $0 < \alpha < 1$ is a relative weight. In the following sections, we study the QoS-aware replica placement problem for different service models.

IV. QoS-Aware Placement for Replica-Aware Services

In the replica-aware service model, the servers in the system are aware of the replication strategy (e.g., by maintaining the object identifiers and the associated replication strategies in the form of directories) [5], [11], [19]. By making use of this information, the servers are capable of directing locally missed requests to the nearest replica of the target object. Figure 1 shows the request paths in an example system where all network links have the same communication cost. The requests originating from v_1, v_2 and v_3 are served by replicas v_4, v_5 and v_5 respectively.

By modeling the content distribution system as a general graph, the QoS-aware placement problems for replica-aware services are formulated as follows.

Definition 1: [The Placement Problems for Replica-Aware Services]

Given a network $G = (V, E)$, the origin copy $r \in V$ with an update rate μ, the storage cost $s(v)$ and the QoS requirement $q(v)$ for each node $v \in V$, the communication cost $d(u, v)$ for each link $(u, v) \in E$, and a relative weight α of update cost to storage cost. Let T be the update distribution tree rooted at r. The storage, update, and combined costs of a replication strategy $R \subseteq V - \{r\}$ are defined as

$$scost(R) \ = \ \sum_{v \in R} s(v),$$

$$ucost(R) \ = \ \mu \cdot \sum_{v \neq r \ \wedge \ T_v \cap R \neq \emptyset} d(v, p(v)),$$

and

$$sucost(R) = \alpha \cdot \sum_{v \in R} s(v) + (1 - \alpha)\mu \cdot \sum_{v \neq r \wedge T_v \cap R \neq \emptyset} d(v, p(v)),$$

respectively, where T_v is the subtree of T rooted at v, and $p(v)$ is the parent of v in T. The objectives of the *min-scost*, *min-ucost*, and *min-sucost* placement problems for replica-aware services are to find a replication strategy R with the minimal storage, update, and combined costs respectively, such that $R \cup \{r\}$ satisfies the QoS requirement of every node $v \in V$, i.e.,

$$\min_{w \in R \cup \{r\}} d(v, w) \leq q(v). \qquad \square$$

Note that the min-scost and min-ucost problems are special cases of the min-sucost problem with α values of 1 and 0 respectively. In the following, we first show that the placement problems for replica-aware services are NP-complete. Several

heuristic algorithms are then proposed and experimentally evaluated against the super-optimal yardstick obtained by solving the relaxed linear program formulation of the problems.

A. NP-Completeness Results

Theorem 1: The min-scost, min-ucost, min-sucost placement problems for replica-aware services are NP-complete.

Proof: Consider a candidate solution for an instance of the replica placement problem in its decision version with an integral bound K. Since the distance of the shortest path between each pair of nodes can be computed in polynomial time, examining whether the candidate solution satisfies the QoS requirements of all nodes has a polynomial time complexity. Moreover, the computation of the total cost and its comparison with the bound K can be performed in polynomial time, be it the storage, update, or combined cost. Therefore, the replica placement problems are in NP.

It can be shown that the replica placement problems are NP-complete by a polynomial reduction from the minimum set cover problem which is known to be NP-complete [20]. See [21] for details.

Hence, the theorem is proven. □

B. Heuristic Algorithms for Replica Placement

A brute-force solution to the replica placement problem is computationally expensive. There are a total of $2^{|V|-1}$ different replication strategies for an exhaustive search, where $|V|$ is the number of servers. The search space is huge even for small values of $|V|$. In this section, we present some heuristic algorithms for replica placement, all of which share the greedy approach. The performance of these algorithms is compared, via simulation experiments, with a super-optimal bound in Section IV-C. The results show that the proposed heuristics generally produce good solutions.

The first family of algorithms is called l-$Greedy$-$Insert$. l-Greedy-Insert starts with an empty replication strategy $R = \emptyset$ and continues to insert replicas into R until all QoS requirements are satisfied. At each step, the insertion alternative with the maximum *normalized benefit* is performed, where the normalized benefit is defined as the increase in the number of nodes whose QoS requirements are satisfied normalized by the increase in replication cost. l-Greedy-Insert allows l-level backtracking in the insertion process. In the first step, the set of $(l+1)$ replicas that maximizes the normalized benefit is inserted into R. In each subsequent step, l-Greedy-Insert examines all possibilities of replacing some l already assigned replicas with $(l+1)$ replicas. Note that the removed replicas and the inserted replicas can overlap. It is obvious that the first step has a time complexity of $O(|V|^{l+1})$ and each subsequent step has a worst case complexity of $O(|V|^{2l+1})$. There can be a total of $O(|V|)$ steps in the worst case. To calculate normalized benefits efficiently, the shortest-path distances between all pairs of nodes need to be computed in a preprocessing stage. This has a time complexity of $O(|V|^3)$. Therefore, the overall time complexity of l-Greedy-Insert is $O(|V|^3)$ for $l = 0$ and $O(|V|^{2l+2})$ for any $l > 0$.

The second family of algorithms is called l-$Greedy$-$Delete$. Different from l-Greedy-Insert, l-Greedy-Delete starts from a complete replication strategy $R = V - \{r\}$ and continues to remove replicas from R provided that no QoS requirement is violated. At each step, the removal alternative with the maximum reduction in replication cost is performed. l-Greedy-Delete also allows l-level backtracking in the removal process. In the first step, l-Greedy-Delete removes from R the set of $(l+1)$ replicas that maximizes the cost reduction without violating any QoS requirement. In each subsequent step, l-Greedy-Delete considers all possibilities of inserting l replicas into R and then removing $(l+1)$ replicas from the new R with no QoS requirement violated. The process continues until the set of feasible alternatives is empty. It is easy to see that the first step has a time complexity of $O(|V|^{l+1})$ and each subsequent step has a worst case complexity of $O(|V|^{2l+1})$. There are a total of $O(|V|)$ steps in the worst case. Similar to l-Greedy-Insert, l-Greedy-Delete also needs a preprocessing stage with time complexity $O(|V|^3)$ to compute the shortest-path distances between all pairs of nodes. Therefore, the overall time complexity of l-Greedy-Delete is $O(|V|^3)$ for $l = 0$ and $O(|V|^{2l+2})$ for any $l > 0$.

The selection of l in the above heuristics reflects a tradeoff between the time complexity and the quality of solution. Let $\mathcal{N}$ be the size of an optimal replication strategy. For l values in the range of 0 to $\mathcal{N}$ ($|V| - \mathcal{N}$), the l-Greedy-Insert (l-Greedy-Delete) heuristic with a larger l value generally produces a solution closer to the optimum at the cost of a higher computational complexity. An l value of 0 degenerates l-Greedy-Insert and l-Greedy-Delete to conventional greedy heuristics.

C. Performance Evaluation

To evaluate the performance of the above heuristics, we have experimentally compared them against a super-optimal bound. Note that the min-sucost replica placement problem can be written as the following 0-1 integer program, assuming the node set $V = \{r, v_1, v_2, \cdots, v_n\}$ where r is the origin copy:

minimize

$$\sum_{i>0} \big(\alpha \cdot s_i \cdot x_i + (1-\alpha) \cdot \mu \cdot d(v_i, p(v_i)) \cdot y_i\big),$$

subject to

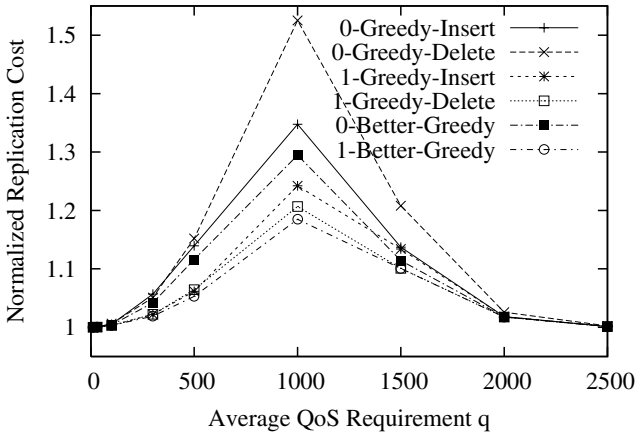
$$\forall i > 0 \wedge d(v_i, r) > q(v_i), \quad \sum_{d(v_i,v_j) \le q(v_i)} x_j \ge 1, \quad (1)$$

$$\forall\ i > 0, \quad y_i \ge x_i, \quad (2)$$

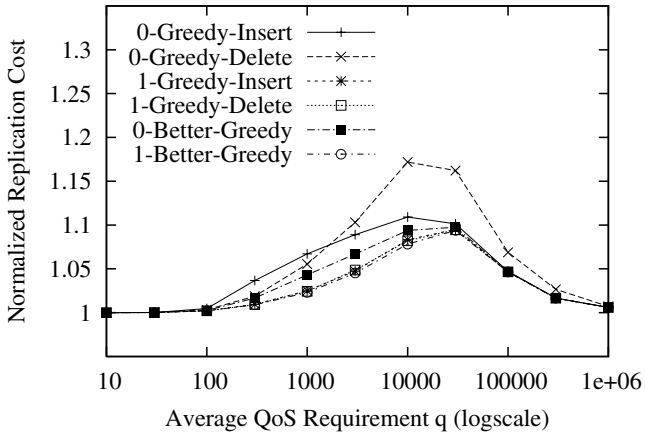
$$\forall\ i > 0\ \wedge\ p(v_j) = v_i, \quad y_i \ge y_j, \quad (3)$$

$$\forall\ i > 0, \quad x_i,\ y_i \in \{0, 1\}. \quad (4)$$

There are a total of $(2|V|-2)$ variables and a maximum of $(3|V|-3)$ constraints in the integer program. The 0-1 variable x_i indicates whether a replica is placed at node v_i and the 0-1 variable y_i indicates whether object updates need to be sent through the link $(v_i, p(v_i))$ in the update distribution tree T, where $p(v_i)$ is the parent of v_i in T. Constraint (1) ensures all



(a) Constant QoS Requirement Distribution q



(b) Uniform QoS Requirement Distribution $[0, 2q]$

Fig. 2. Performance for Different QoS Requirements

QoS requirements are satisfied, i.e., if the QoS requirement of a node cannot be met by the origin copy, it has to be satisfied by some replica. Constraints (2) and (3) ensure object updates are distributed to each replica. In our experiments, we relax the integer program to a regular linear program by replacing the last constraint (4) with $\forall\ i > 0,\ 0 \leq x_i,\ y_i \leq 1$, and compute the optimal solution to the regular linear program. Since the solution may not be integral (i.e., not feasible in practice), it provides a *super* bound on the optimal solution of the replica placement problem.

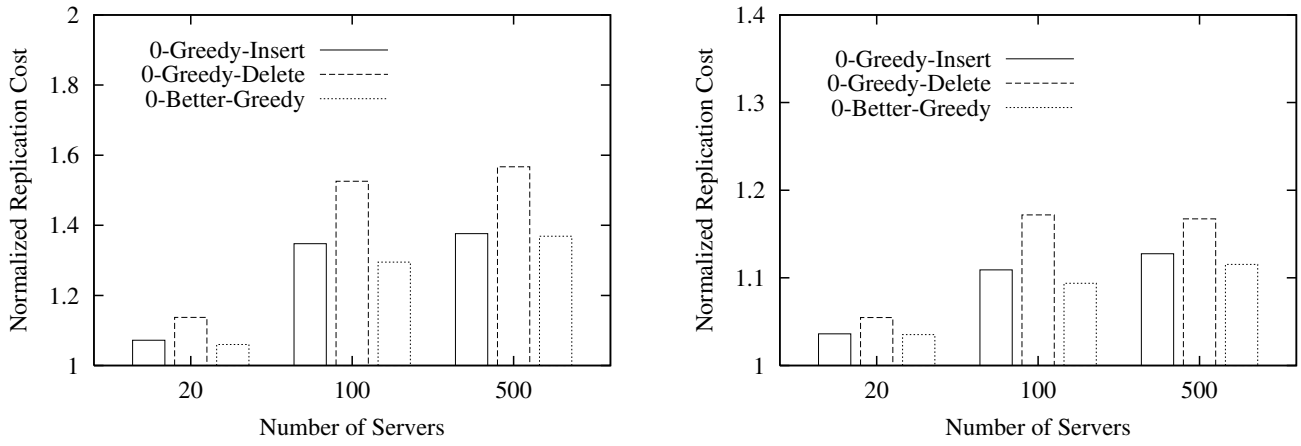
In our experiments, the network topology of the content distribution system was randomly generated using the Waxman model [22]. The model is designed to produce network topologies that resemble typical internetworks. It works by first placing a given number of N nodes on a square plane s distance units by s distance units. The links are then inserted to connect the nodes. A link is created between each pair of nodes (u, v) with probability $p(u, v) = \beta_1 \cdot e^{-d(u,v)/(\beta_2 \cdot L)}$, where $d(u, v)$ is the Euclidean distance between u and v, $L = \sqrt{2} \cdot s$ is the maximum distance between any two nodes, and β_1, β_2 are Waxman parameters. The cost of each link is given by the Euclidean distance between the two endpoints. The experiments were performed over a wide range of parameter settings. In our default parameter setting, N was set at 100, s was set one order of magnitude higher than N, β_1 was set at 0.1, and β_2 was set at 0.6. Under this setting, the average number of links in the networks generated was about 280, and the average communication cost of the links was about 450.

A server was assumed to be located at each node in the network. The origin copy was assumed to be located at a randomly selected node and was assigned an update rate of one per time unit. The shortest paths tree rooted at the origin copy was taken as the update distribution tree. The default storage cost at each node was set at 1,000. Given a mean value q, the QoS requirements of all nodes were assigned based on two different distributions: a constant q, and a uniform distribution in $[0, 2q]$. By default, the storage and update costs were assigned equal weights in the combined cost, i.e., the relative weight $\alpha = 0.5$.

The following algorithms described in Section IV-B were evaluated in our simulation study: 0-Greedy-Insert, 0-Greedy-Delete, 1-Greedy-Insert and 1-Greedy-Delete. In addition, two new algorithms called *0-Better-Greedy* and *1-Better-Greedy* were also included. The 0-Better-Greedy (1-Better-Greedy) algorithm outputs the lower-cost replication strategy between the solutions of 0-Greedy-Insert (1-Greedy-Insert) and 0-Greey-Delete (1-Greey-Delete). Note that the time complexities of 0-Better-Greedy and 1-Better-Greedy are asymptotically the same as the simple 0-Greedy and 1-Greedy heuristics respectively.

For each parameter setting, we randomly generated 1,000 different network topologies. The average performance of these 1,000 simulation runs is plotted for performance comparison. To quantify the relative performance difference, the replication costs of different algorithms are normalized with respect to the super-optimal bound.

Figures 2(a) and 2(b) show the normalized replication costs as a function of q for constant and uniform distributions of QoS requirements respectively. As evident from the superior performance of Better-Greedy heuristics over their simple Greedy peers, the Greedy-Insert and Greedy-Delete heuristics do not dominate one another. On the other hand, the 1-Greedy heuristics generally outperform the 0-Greedy heuristics. The normalized costs of 0-Better-Greedy and 1-Better-Greedy heuristics are consistently within 30% and 19% of the super-optimal bound respectively under constant distribution of QoS requirements. They are within 5% and 3% of the super-optimal bound respectively under uniform distribution



(a) Constant QoS Requirement Distribution q

(b) Uniform QoS Requirement Distribution $[0, 2q]$

Fig. 3. Performance for Different Numbers of Servers

of QoS requirements. When the QoS requirements of most nodes are low (i.e., $q \geq 2,000$ for constant distribution and $q \geq 30,000$ for uniform distribution), few replicas need to be placed in addition to the origin copy. In contrast, when the QoS requirements are high (i.e., $q \leq 100$), a replica should be placed at almost every node. In the above two cases, the performance of all heuristic algorithms approaches the super-optimal bound. Noticeable difference between 0-greedy and 1-greedy heuristics appears at moderate q values. The better performance of 1-greedy heuristics is achieved at the cost of a higher asymptotic complexity. Similar performance trends have also been observed in our experiments with different network connectivities and different relative weights α's of update cost to storage cost. The results are not shown here due to space limitation.

Figures 3(a) and 3(b) show the experimental results for different numbers of servers in the content distribution system (i.e., the number of nodes modeled in the network). In these experiments, the connectivity parameters β_1 and β_2 were set such that the average node degree was kept similar across networks. We did not simulate the 1-greedy heuristics for networks with 500 servers due to their high computational requirements. As seen from Figure 3, the normalized costs of the greedy heuristics generally increase with the number of servers. However, the increasing rate of normalized cost reduces with growing number of servers. The performance difference between networks with 500 and 100 servers is much smaller than that between networks with 100 and 20 servers. For content distribution systems with 500 servers, the normalized costs of 0-Better-Greedy are 1.37 and 1.12 under constant and uniform distributions of QoS requirements respectively. This demonstrates that the greedy heuristics generally produce close-to-optimal solutions with much lower computation complexity than a brute-force approach.

V. QoS-Aware Placement for Replica-Blind Services

In the replica-blind service model, the servers in the system are not aware of the replication strategy. Thus, request routing is independent of where the replicas of the target object are located. Examples of replica-blind service include static proxy hierarchies [12], [23], [24] and en-route content distribution architectures [4], [25], [26]. In these systems, each replica only serves the requests flowing through it under some given routing strategy which can be implemented at either the application level or the network level [1]. Regardless of the underlying routing mechanism used, the request/delivery paths between all nodes and a given origin server are represented by a tree topology rooted at the origin server. This tree is simultaneously used for the routing purposes of both object retrieval and update. Consider again the example system in Figure 1. Assuming all requests are routed through the shortest paths towards the origin server, the corresponding request paths are shown in Figure 4. As can be seen, the requests originating from v_1 and v_2 are served by the origin copy, and the requests from v_3 is served by replica v_5. Note that requests in a replica-blind serice model are not necessarily served by the physically nearest replica. They are satisfied by the nearest replica *along* the direction towards the origin copy.

The QoS-aware placement problems for replica-blind services are formulated as follows.

Definition 2: [The Placement Problems for Replica-Blind Services]

Given a tree $T = (V, E)$ rooted at the origin copy $r \in V$ with an update rate μ, the storage cost $s(v)$ and the QoS requirement $q(v)$ for each node $v \in V$, the communication cost $d(u, v)$ for each link $(u, v) \in E$, and a relative weight α of update cost to storage cost. The storage, update, and

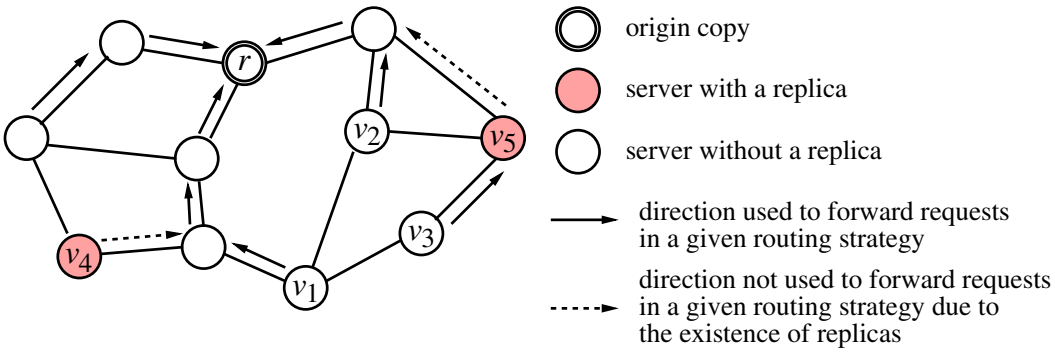


Fig. 4. Replica-Blind Service Model

combined costs of a replication strategy $R \subseteq V - \{r\}$ are defined as

$$scost(R) = \sum_{v \in R} s(v),$$

$$ucost(R) = \mu \cdot \sum_{v \neq r \ \wedge \ T_v \cap R \neq \emptyset} d(v, p(v)),$$

and

$$sucost(R) = \alpha \cdot \sum_{v \in R} s(v) + (1-\alpha)\mu \cdot \sum_{v \neq r \wedge T_v \cap R \neq \emptyset} d(v, p(v)),$$

respectively, where T_v is the subtree of T rooted at v, and $p(v)$ is the parent of v. The objectives of the *min-scost*, *min-ucost*, and *min-sucost* placement problems for replica-blind services are to find a replication strategy R with the minimal storage, update, and combined costs respectively, such that $R \cup \{r\}$ satisfies the QoS requirement of every node $v \in V$, i.e.,

$$d(v, l(v, R \cup \{r\})) \leq q(v),$$

where $l(v, R \cup \{r\})$ is the lowest ancestor of v in $R \cup \{r\}$. □

A. Optimal Placement with Minimal Storage Cost

This section presents a dynamic programming solution to the min-scost replica placement problem. As shown in Figure 5, we consider a more generalized problem of placing replicas in a subtree rooted at node x, assuming the lowest ancestor of x that has a replica is node y. This new problem is formally defined as follows.

Definition 3: Let node y be an ancestor of node x in tree T, and V_x be the set of nodes in the subtree of T rooted at x (see Figure 5). Assume an object replica is placed at y. The problem of finding a replication strategy $R(x, y) \subseteq V_x$ with the minimal storage cost

$$scost(R(x, y)) = \sum_{v \in R(x,y)} s(v),$$

such that $R(x, y) \cup \{y\}$ satisfies the QoS requirement of every node $v \in V_x$, i.e.,

$$d(v, l(v, R(x, y) \cup \{y\})) \leq q(v),$$

is referred to as the (x, y)-optimization problem. □

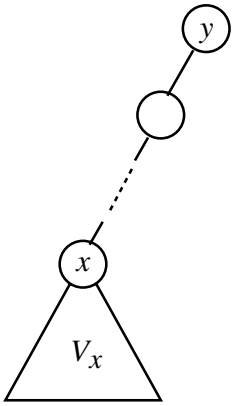


Fig. 5. The (x, y)-Optimization Problem

By creating a dummy parent p for the root r, an original tree $T = (V, E)$ can be transformed into a new tree $T^* = (V \cup \{p\}, E \cup \{(r, p)\})$, where $d(r, p) = 0$. It is easy to see that the min-scost replica placement problem in T is equivalent to the (r, p)-optimization problem in T^*. To develop a dynamic programming algorithm, Theorem 2 proves that an optimal solution to the (x, y)-optimization problem must contain optimal solutions to some subproblems.

Theorem 2: Consider three nodes x, y and z in tree T, where y is an ancestor of x, and z is a child of x (see Figure 6). Let V_z be the set of nodes in the subtree of T rooted at z. Let $R(x, y)$, $R(z, x)$ and $R(z, y)$ be optimal solutions to the (x, y)-, (z, x)- and (z, y)-optimization problems respectively.

(i) If $x \in R(x, y)$, the replication strategy $R'(x, y) = (R(x, y) - V_z) \cup R(z, x)$ is also an optimal solution to the (x, y)-optimization problem.

(ii) Otherwise, if $x \notin R(x, y)$, the replication strategy $R''(x, y) = (R(x, y) - V_z) \cup R(z, y)$ is also an optimal solution to the (x, y)-optimization problem.

Proof: See [21] for details. □

Theorem 2 implies the following properties.

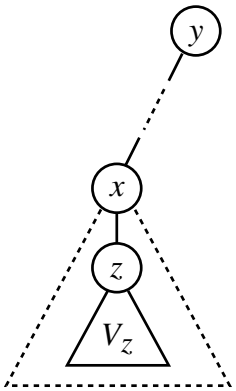


Fig. 6. Rationale of Dynamic Programming

Theorem 3: Let node y be an ancestor of node x in tree T, and $Z(x)$ be the set of x's children. Let $R(x,y)$ be an optimal solution to the (x,y)-optimization problem.

(i) If $x \in R(x,y)$, the replication strategy $\{x\} \cup \bigcup_{z\in Z(x)} R(z,x)$ is also an optimal solution to the (x,y)-optimization problem, where $R(z,x)$ is an optimal solution to the (z,x)-optimization problem.

(ii) Otherwise, if $x \notin R(x,y)$, the replication strategy $\bigcup_{z\in Z(x)} R(z,y)$ is also an optimal solution to the (x,y)-optimization problem, where $R(z,y)$ is an optimal solution to the (z,y)-optimization problem.

Proof: Suppose $Z(x) = \{z_1, z_2, \cdots, z_k\}$.

If $x \in R(x,y)$, by iteratively applying claim (i) of Theorem 2, the following replication strategies are all optimal solutions to the (x,y)-optimization problem:

$$\begin{aligned}
&(R(x,y) - V_{z_1}) \cup R(z_1,x),\\
&(R(x,y) - V_{z_1} - V_{z_2}) \cup R(z_1,x) \cup R(z_2,x),\\
&(R(x,y) - V_{z_1} - V_{z_2} - V_{z_3})\\
&\qquad \cup R(z_1,x) \cup R(z_2,x) \cup R(z_3,x),\\
&\qquad \cdots\cdots,\\
&(R(x,y) - \bigcup_{i=1}^{k} V_{z_i}) \cup (\bigcup_{i=1}^{k} R(z_i,x)). \qquad (5)
\end{aligned}$$

Since $x \in R(x,y)$, it follows that $R(x,y) - \bigcup_{i=1}^{k} V_{z_i} = \{x\}$. Thus, (5) $= \{x\} \cup \bigcup_{i=1}^{k} R(z_i,x)$ is an optimal solution to the (x,y)-optimization problem, and claim (i) is proven.

A similar proof can be done for claim (ii). Hence, the theorem is proven. □

Consider the tree $T^* = (V \cup \{p\}, E \cup \{(r,p)\})$ rooted at p that is induced from $T = (V,E)$. For each pair of nodes $x, y \in V \cup \{p\}$ where y is an ancestor of x, let $R(x,y)$ be an optimal solution to the (x,y)-optimization problem in T^*, and $min\text{-}scost(x,y)$ be the storage cost of $R(x,y)$. The (x,y)-optimization problem is trivial if x is a leaf in T^*. In this case, if $d(x,y) \le q(x)$, no replica needs to be placed at x; otherwise, if $d(x,y) > q(x)$, a replica should be placed at x. For each non-leaf node x in T^*, if $d(x,y) > q(x)$, a replica must be placed at x; otherwise, as seen from Theorem 3, the following two possibilities of $R(x,y)$ need to be compared: $\{x\} \cup \bigcup_{z\in Z(x)} R(z,x)$ and $\bigcup_{z\in Z(x)} R(z,y)$, the costs of which are given by $c_1(x,y) = s(x) + \sum_{z\in Z(x)} min\text{-}scost(z,x)$ and $c_2(x,y) = \sum_{z\in Z(x)} min\text{-}scost(z,y)$ respectively, where $Z(x)$ is the set of x's children in T^*. Therefore, the recurrences for dynamic programming are given by:

$$min\text{-}scost(x,y) = \begin{cases} 0 & \text{if } x \text{ is a leaf and } d(x,y) \le q(x),\\ s(x) & \text{if } x \text{ is a leaf and } d(x,y) > q(x),\\ \min\{c_1(x,y), c_2(x,y)\} & \text{if } x \text{ is not a leaf and } d(x,y) \le q(x),\\ c_1(x,y) & \text{if } x \text{ is not a leaf and } d(x,y) > q(x), \end{cases}$$

and

$$R(x,y) = \begin{cases} \emptyset & \text{if } x \text{ is a leaf and } d(x,y) \le q(x),\\ \{x\} & \text{if } x \text{ is a leaf and } d(x,y) > q(x),\\ \{x\} \cup \bigcup_{z\in Z(x)} R(z,x) & \text{if } x \text{ is not a leaf, } d(x,y) \le q(x),\\ & \text{and } c_1(x,y) \le c_2(x,y),\\ \bigcup_{z\in Z(x)} R(z,y) & \text{if } x \text{ is not a leaf, } d(x,y) \le q(x),\\ & \text{and } c_1(x,y) > c_2(x,y),\\ \{x\} \cup \bigcup_{z\in Z(x)} R(z,x) & \text{if } x \text{ is not a leaf and } d(x,y) > q(x). \end{cases}$$

Starting from the entries where x is a leaf, we can compute all $R(x,y)$'s and $min\text{-}scost(x,y)$'s by a post-order traversal of x in T^*. $R(r,p)$ is the optimal solution to the min-scost replica placement problem.

The space and time complexities of the dynamic programming solution are analyzed as follows. Since there are at most $O(|V|^2)$ R-entries and $min\text{-}scost$-entries to compute respectively, the worst case space complexity is given by $O(|V|^2)$. The time complexity to compute each entry $R(x,y)/min\text{-}scost(x,y)$ is $O(N_c(x))$, where $N_c(x)$ is the number of x's children. Note that given a node $x \in V$, there are a total of $N_a(x)$ R-entries/$min\text{-}scost$-entries in the form of $R(x,y)/min\text{-}scost(x,y)$, where $N_a(x)$ is the number of x's ancestors in T^*. Therefore, the total computation complexity of $R(r,p)$ is given by

$$\begin{aligned} O\Big(\sum_{x\in V} N_a(x) \cdot N_c(x)\Big) &\le O\Big(\sum_{x\in V} |V| \cdot N_c(x)\Big)\\ &= O\Big(|V| \cdot \sum_{x\in V} N_c(x)\Big)\\ &= O(|V|^2). \end{aligned}$$

B. Optimal Placement with Minimal Update Cost

This section presents an optimal solution to the min-ucost replica placement problem. Unlike storage costs, the update costs of different replicas are inter-related. Placing a new

replica does not increase the total update cost if some replicas have already been placed downstream to the new replica in the tree. As mentioned in Section III, the total update cost of a replication strategy depends on the locations of the most downstream replicas only. The closer the replicas to the origin copy, the lower the update cost. Therefore, the most downstream replicas should be selected such that each of them satisfies the QoS requirements of some nodes that are not satisfied by its parent. Theorem 4 presents an optimal replication strategy that produces the minimal update cost.

Theorem 4: The replication strategy

$$R^* = \{v \mid v \neq r \wedge \Delta cost(v) = \min_{w \in T_v} (q(w) - d(w,v)) < d(v,p(v))\}$$

is an optimal solution to the min-ucost placement problem for replica-blind services.

Proof: We first show that R^* satisfies the QoS requirement of every node $w \in V$. If $d(w,r) \leq q(w)$, the claim is straightforward. Otherwise, if $d(w,r) > q(w)$, since $d(w,w) = 0 \leq q(w)$, there exists one node v on the path between r and w such that $d(w,p(v)) > q(w)$ and $d(w,v) \leq q(w)$. Thus, $q(w) < d(w,p(v)) = d(w,v) + d(v,p(v))$, and equivalently, $q(w) - d(w,v) < d(v,p(v))$. By definition, $v \in R^*$, and it follows that $d(w, l(w, R^* \cup \{r\})) \leq d(w,v) \leq q(w)$. Therefore, R^* satisfies the QoS requirements of all nodes.

To prove the optimality of R^*, we show that for any replication strategy R satisfying all QoS requirements, $ucost(R) \geq ucost(R^*)$. In fact, for each node $v \neq r$, $T_v \cap R^* \neq \emptyset$ implies $T_v \cap R \neq \emptyset$. Let x be a node in $T_v \cap R^*$. According to the definition of R^*, there exists a node $w \in T_x$ such that $q(w) - d(w,x) < d(x,p(x))$. On the other hand, since R satisfies the QoS requirement of every node, we have $d(w, l(w, R \cup \{r\})) \leq q(w)$. Therefore, $d(w, l(w, R \cup \{r\})) \leq q(w) < d(w,x) + d(x,p(x)) = d(w,p(x))$. Note that $p(x)$ and $l(w, R \cup \{r\})$ are both ancestors of w. This implies $p(x)$ must be an ancestor of $l(w, R \cup \{r\})$, and thus, $l(w, R \cup \{r\}) \in T_x$. Since $x \in T_v$, we have $l(w, R \cup \{r\}) \in T_v$. Therefore, $l(w, R \cup \{r\}) \in T_v \cap R$ and $T_v \cap R \neq \emptyset$. It follows from Definition 2 that

$$\begin{aligned} ucost(R^*) &= \mu \cdot \sum_{v \neq r \ \wedge \ T_v \cap R^* \neq \emptyset} d(v,p(v)) \\ &\leq \mu \cdot \sum_{v \neq r \ \wedge \ T_v \cap R \neq \emptyset} d(v,p(v)) \\ &= ucost(R). \end{aligned}$$

Hence, the theorem is proven. □

It is obvious that if $v \in R^*$, all non-root ancestors of v are in R^*. Therefore, the optimal solution R^* induces a connected subgraph in tree T.

The values of $\Delta cost(v)$ in Theorem 4 can be computed in an iterative fashion by a post-order traversal of v in T with the following recurrences:

$$\Delta cost(v) = \begin{cases} q(v) & \text{if } v \text{ is a leaf,} \\ \min\{q(v), \min\limits_{z \in Z(v)} (\Delta cost(z) - d(z,v))\} & \\ & \text{otherwise,} \end{cases}$$

where $Z(v)$ is the set of v's children. The computational complexity of $\Delta cost(v)$'s is $O(|V|)$. On obtaining these values, the optimal replication strategy can be computed based on Theorem 4 in $O(|V|)$ time. Thus, the computation of R^* has a total time complexity of $O(|V|)$.

C. Optimal Placement with Minimal Combined Cost

Finally, we consider the min-sucost replica placement problem. It can be solved by a similar dynamic programming algorithm to that of the min-scost problem. The corresponding (x,y)-optimization problem is defined as follows.

Definition 4: Let node y be an ancestor of node x in tree T, and V_x be the set of nodes in the subtree of T rooted at x (see Figure 5). Assume an object replica is placed at y. The objective of the (x,y)-optimization problem is to find a replication strategy $R(x,y) \subseteq V_x$ satisfying the QoS requirement of every node $v \in V_x$, i.e.,

$$d(v, l(v, R(x,y) \cup \{y\})) \leq q(v),$$

with the minimal combined cost

$$\begin{aligned} & sucost(R(x,y)) \\ = \ & \alpha \cdot \sum_{v \in R(x,y)} s(v) + (1-\alpha)\mu \cdot \Big(\gamma \cdot d(x,y) \\ & + \sum_{v \in V_x - \{x\} \ \wedge \ T_v \cap R(x,y) \neq \emptyset} d(v,p(v))\Big), \end{aligned}$$

where

$$\gamma = \begin{cases} 0 & \text{if } R(x,y) = \emptyset, \\ 1 & \text{if } R(x,y) \neq \emptyset. \end{cases} \quad \square$$

The same conclusions of Theorems 2 and 3 can be proven for the above definition of (x,y)-optimization problem. Detailed proofs are omitted in this paper due to space limitation. Consider the tree $T^* = (V \cup \{p\}, E \cup \{(r,p)\})$ rooted at p. For each pair of nodes $x, y \in V \cup \{p\}$ where y is an ancestor of x, let $R(x,y)$ be an optimal solution to the (x,y)-optimization problem in T^* and $min\text{-}sucost(x,y)$ be the combined cost of $R(x,y)$. Similar to the analysis in Section V-A, for each leaf x in T^*, if $d(x,y) \leq q(x)$, the optimal solution $R(x,y) = \emptyset$; otherwise, if $d(x,y) > q(x)$, the optimal solution $R(x,y) = \{x\}$. For each non-leaf node x in T^*, if $d(x,y) > q(x)$, a replica must be placed at x; otherwise, if $d(x,y) > q(x)$, the following two possibilities of $R(x,y)$ should be compared: $\{x\} \cup \bigcup_{z \in Z(x)} R(z,x)$ and

$\bigcup_{z \in Z(x)} R(z, y)$, the costs of which are given by

$$c_1(x, y) = \alpha \cdot s(x) + (1 - \alpha)\mu \cdot d(x, y) + \sum_{z \in Z(x)} min\text{-}sucost(z, x),$$

and

$$c_2(x, y) = \begin{cases} 0 & \text{if } \bigcup_{z \in Z(x)} R(z, y) = \emptyset, \\ (1 - \alpha)\mu \cdot d(x, y) & \\ \quad + \sum_{z \in Z(x) \wedge R(z,y) \neq \emptyset} \big(min\text{-}sucost(z, y) & \\ \quad - (1 - \alpha)\mu \cdot d(x, y)\big) & \\ & \text{if } \bigcup_{z \in Z(x)} R(z, y) \neq \emptyset, \end{cases}$$

respectively, where $Z(x)$ is the set of x's children in T^*.

The recurrences for dynamic programming, not shown here due to space limitation, are similar to the ones in Section V-A. The space and time complexities of the dynamic programming algorithm are both given by $O(|V|^2)$.

VI. Conclusion

We have investigated the minimal cost replica placement problem for QoS-aware content distribution. The problem has been formulated under two classes of service models (replica-aware service and replica-blind service) and three different cost models (storage cost, update cost, and their combination). In replica-aware services, the content distribution system is modeled as a general graph. The associated replica placement problems are proven to be NP-complete. Several heuristic algorithms have been proposed and experimentally evaluated against a super-optimal bound obtained from the relaxed linear program. The results show that the proposed heuristics perform close to the super-optimal bound. In replica-blind services, the delivery paths with respect to a given origin server are represented by a tree topology. It is shown that the optimal solution to the associated replica placement problem for minimal update cost can be computed with a time complexity linear to the number of servers. There also exist polynomial optimal solutions to the associated replica placement problems for minimal storage and combined costs. Dynamic programming algorithms with time complexities square to the number of servers have been proposed for these two problems.

Acknowledgment

Jianliang Xu's work was supported by a grant from the Hong Kong Baptist University (Grant No. FRG/02-03/II-34).

References

[1] M. Rabinovich and O. Spatscheck, *Web Caching and Replication*. Addison-Wesley, 2002.

[2] M. Colajanni, P. S. Yu, and V. Cardellini, "Scalable web-server systems: Architectures, models and load balancing algorithms," in *Tutorial Presented at ACM SIGMETRICS'00*, June 2000.

[3] D. A. Menasce, "QoS issues in web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72–75, November/December 2002.

[4] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–582, Oct. 2000.

[5] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proceedings of IEEE INFOCOM'01*, Apr. 2001, pp. 1587–1596.

[6] I. Cidon, S. Kutten, and R. Soffer, "Optimal allocation of electronic content," in *Proceedings of IEEE INFOCOM'01*, Apr. 2001, pp. 1773–1780.

[7] L. W. Dowdy and D. V. Foster, "Comparative models of the file assignment problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287–313, June 1982.

[8] M. L. Fisher and D. S. Hochbaum, "Database location in computer networks," *Journal of the ACM*, vol. 27, no. 4, pp. 718–735, Oct. 1980.

[9] O. Wolfson and A. Milo, "The multicast policy and its relationship to replicated data placement," *ACM Transactions on Database Systems*, vol. 16, no. 1, pp. 181–205, Mar. 1991.

[10] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the Internet," in *Proceedings of IEEE INFOCOM'99*, Mar. 1999, pp. 1282–1290.

[11] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the Internet," in *Proceedings of IEEE INFOCOM'01*, Apr. 2001, pp. 31–40.

[12] J. Xu, B. Li, and D. L. Lee, "Placement problems for transparent data replication proxy services," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1383–1398, Sept. 2002.

[13] X. Jia, D. Li, X. Hu, W. Wu, and D. Du, "Placement of web-server proxies with consideration of read and update operations on the Internet," *The Computer Journal*, vol. 46, no. 4, July 2003.

[14] K. Kalpakis, K. Dasgupta, and O. Wolfson, "Optimal placement of replicas in trees with read, write, and storage costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 6, pp. 628–637, June 2001.

[15] M. R. Korupolu and M. Dahlin, "Coordinated placement and replacement for large-scale distributed caches," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 6, pp. 1317–1329, November/December 2002.

[16] X. Tang and S. T. Chanson, "Coordinated management of cascaded caches for efficient content distribution," in *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE)*, Mar. 2003, pp. 37–48.

[17] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proceedings of ACM SIGMETRICS'00*, June 2000, pp. 1–12.

[18] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," in *Proceedings of IEEE INFOCOM'02*, June 2002, pp. 1366–1375.

[19] M. Rabinovich, J. Chase, and S. Gadde, "Not all hits are created equal: Cooperative proxy caching over a wide area network," *Computer Networks and ISDN Systems*, vol. 30, no. 22–23, pp. 2253–2259, Nov. 1998.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[21] X. Tang and J. Xu, "On replica placement for QoS-aware content distribution" (extended version), available at http://www.ntu.edu.sg/home/asxytang/papers/replica.pdf.

[22] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[23] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical Internet object cache," in *Proceedings of the 1996 USENIX Annual Technical Conference*, Jan. 1996, pp. 153–163.

[24] 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, pp. 404–418, Aug. 2001.

[25] P. Rodriguez and S. Sibal, "SPREAD: scalable platform for reliable and efficient automated distribution," *Computer Networks*, vol. 33, no. 1–6, pp. 33–49, June 2000.

[26] X. Tang and S. T. Chanson, "Coordinated en-route web caching," *IEEE Transactions on Computers*, vol. 51, no. 6, pp. 595–607, June 2002.