QoS-Aware Replica Placement for Content Distribution

Xueyan Tang, *Member*, *IEEE*, and Jianliang Xu, *Member*, *IEEE*

Abstract—The rapid growth of new information services and business-oriented applications entails the consideration of quality of service (QoS) in content distribution. This paper investigates the QoS-aware replica placement problems for responsiveness QoS requirements. We consider two classes of service models: replica-aware services and replica-blind services. In replica-aware services, the servers are aware of the locations of replicas and can therefore optimize request routing to improve responsiveness. We show that the QoS-aware placement problem for replica-aware services is NP-complete. Several heuristic algorithms for fast computation of good solutions are proposed and experimentally evaluated. In replica-blind services, 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.

Index Terms—Content distribution, replication, placement, quality of service, dynamic programming, NP-complete.

1 Introduction

The rapid growth of new information services and business-oriented applications is increasing the demand to support quality of service (QoS) in content distribution [6], [16]. Responsiveness, which refers to how fast the users can access the requested information, is an important type of QoS requirements needed by a wide range of applications. For example, shareholders have responsiveness requirements on stock information services to assist them with their business, and drivers have such requirements on traffic information services to help them keep away from congested sections on highways. The desired level of performance can be specified in the form of service level agreements (SLAs) between the content/service providers and their customers, e.g., the response time of requests from domain A for Nasdaq.com home should not exceed 1 second; 95 percent of the requests from domain B for CNN.com home should complete in less than 2 seconds [16].

In this paper, we address the challenges introduced by responsiveness QoS requirements in content replication—a widely employed technique to improve the performance of large-scale content distribution systems such as CDNs [19]. The effectiveness of replication, to a large extent, depends on the locations where the replicas are placed. 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, 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 improving some average performance measure of the entire client community such as the mean access latency [14], [17], [5]. Although an average measure may be important under some circumstances, it neither provides any level of performance guarantee nor differentiates the likely diverse QoS requirements of the individuals. Optimizing an average performance measure does not necessarily satisfy the performance requirements of all users. So far, to the best of our knowledge, there has been no study on QoS-aware replica placement.

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, and a combination of both. We consider two classes of service models that lead to different problem formulations: replica-aware services and replica-blind services.

In replica-aware services, the servers in the system are aware of the locations of replicas. By making use of this information, the servers are capable of optimizing request routing to improve responsiveness. 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 good solutions. They are evaluated, via simulation experiments, against a super-optimum yardstick obtained from the solution of a relaxed linear program.

In replica-blind services, 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 the replicas. 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 there exist polynomial optimal solutions to the QoS-aware

- *X. Tang is with the School of Computer Engineering, Nanyang Technological University, Nanyang Ave., Singapore 639798. E-mail: asxytang@ntu.edu.sg.*
- *J. Xu is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. E-mail: xujl@comp.hkbu.edu.hk.*

Manuscript received 5 Mar. 2004; revised 12 Aug. 2004; accepted 4 Dec. 2004; published online 22 Aug. 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0064-0304.

placement problem for replica-blind services. 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 2 summarizes the related work. Section 3 describes the system model and provides some basic definitions. Section 4 presents a formulation of the QoS-aware placement problem for replica-aware services and proposes several heuristic solutions. The QoS-aware placement problem for replica-blind services is formulated and investigated in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Early work on replica placement had investigated the file allocation problem (FAP) in storage systems [8] and the database location problem (DBL) in computer networks [9]. 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. [26] 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. Moreover, the storage costs of replicas were not considered in their 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 called the server placement problem and the latter is referred to as 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. [15] and Krishnan et al. [14] developed polynomial optimal solutions to place some given number of servers in a tree network to minimize the average retrieval cost of all clients. The same problem for general network topologies was shown to be NP-complete. Qiu et al. [17] experimentally compared several heuristic solutions and found a simple greedy algorithm performed the best. Cronin et al. [7] investigated the constrained mirror placement problem where the 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. [27] and Jia et al. [11] further took the update cost into consideration. Cidon et al. [5], on the other hand, used the total storage and retrieval cost as the target 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. [12] who optimized three types of costs (i.e., retrieval, update, and storage costs) for replica placement in an integrated fashion. However, most work described above aimed at maximizing the performance gain in terms of some average measure with a given budget of resources. To the best of our knowledge, no 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 desired level of service. We investigate the QoS-aware replica placement problem for a variety of service and cost models. In parallel to an earlier version of our work [23], Karlsson and Karamanolis [13] conducted a complementary study on the lower bound of replication cost to meet specific performance goals.

3 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. Different servers may have different storage costs. Moreover, a distance $d(u, v)$ is associated with each edge $(u, v) \in E$, representing the communication cost of sending a request for the object and the associated response 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 latency 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 distance of the shortest path between u and v.

The object is associated with an authoritative *origin server* in the network where the content provider makes 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 group of clients (e.g., by statically configuring the clients, using DNS-based request direction, or intercepting requests in a transparent fashion [19]). If the object is replicated at the server receiving the request, the response is generated locally. Otherwise, the server forwards the request to some other server in the network and relays the response to the client.

The access latency of a client request is affected by factors including network latency (in terms of round-trip time), server load, and network load. Similar to other studies [7], [13], in this paper, we focus on the network latency factor and consider the communication cost involved in serving requests (called *retrieval cost*) as a measure of the responsiveness perceived by the clients. The motivation is that a busy server or a bottleneck link that has to handle a large number of requests can always be better provisioned to meet the capacity requirements (e.g., by using a server cluster [3] or upgrading the bandwidth). However, the network latency cannot be improved by simply adding hardware resources. Since the communication cost from a client to the

TABLE 1
Summary of Notations

Notation	Definition
$s(v)$	storage cost at node v
$q(v)$	QoS requirement of node v
$d(u,v)$	distance between nodes u and v
$p(v)$	parent of node v
T_v	subtree rooted at node v
r	authoritative origin copy
μ	object update rate
R	replication strategy
$scost(R)$	storage cost of replication strategy R
$ucost(R)$	update cost of replication strategy R
α	relative weight of storage and update costs

associated server receiving its request 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 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 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, i.e.,

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

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). The updates are delivered from the origin copy to all replicas via application-level multicast, in which each server receives the updates from its parent and is responsible for further distributing the updates to its children [10], [28]. 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

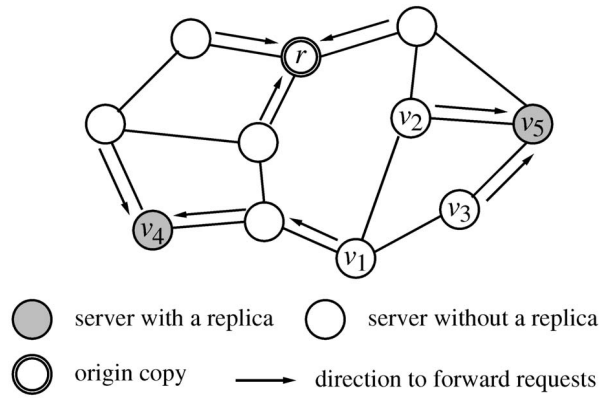


Fig. 1. Replica-aware service model.

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

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 multicast. We shall not consider the cost of creating replicas explicitly in our model. Note that the cost is essentially the same as that of delivering an object update. Therefore, the cost of creating replicas can be modeled in the update cost by treating the creation as a "first update." Depending on the business model, the replication cost of R can take the form of storage cost $scost(R)$, update cost $ucost(R)$, or a combination of the two costs

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

where $0 < \alpha < 1$ is a relative weight. Table 1 summarizes the notations used in this paper. In the following sections, we study the QoS-aware replica placement problem for different service models. Note that, although the problem is formulated for a single object, our discussion is applicable to different granularities of replication. If the basic unit of replication consists of a group of objects, the proposed solutions can be adapted by setting the QoS requirement at each node to the highest among the QoS requirements of all objects if they are different and setting the update rate to the average rate of all objects. The selection of replication granularity reflects a trade-off between maintenance overhead and system performance, yet it is an issue beyond the scope of this paper.

4 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 object identifiers and the associated replication strategies in the form of directories). By making use of this information, the servers are capable of directing locally missed requests to the nearest replica of the target object [17], [7], [18]. Fig. 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 distance $d(u, v)$ for each link $(u, v) \in E$, the update distribution tree T rooted at r, and a relative weight α of update cost to storage cost. 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.,*

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

where $d(v, w)$ is the distance of the shortest path between v and w.

Note that the min-scost and min-ucost problems are special cases of the min-sucost problem with α values 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 a super-optimum yardstick obtained by solving the relaxed linear program formulation of the problems.

4.1 NP-Completeness Results

Theorem 1. *The min-scost, min-ucost, and 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. The computation of total cost and its comparison with the bound K can also be performed in polynomial time, be it the storage, update, or combined cost. Therefore, the replica placement problems are in NP.

We show 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 [29]. The minimum set cover problem is defined as follows: Given a finite set S and a collection $\mathcal{F}$ of subsets of S where $\bigcup_{F \in \mathcal{F}} F = S$, the objective of the minimum set cover problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ such that $\bigcup_{F \in \mathcal{C}} F = S$.

Let P be an instance of the minimum set cover problem. Assume there are n elements in S and m subsets of S in $\mathcal{F}$: $S = \{s_1, s_2, \cdots, s_n\}$ and $\mathcal{F} = \{F_1, F_2, \cdots, F_m\}$. We first construct a graph $G = (V, E)$ with a node r, m x-type nodes, and n y-type nodes: $V = \{r, x_1, x_2, \cdots, x_m, y_1, y_2, \cdots, y_n\}$. Each x-type node corresponds to a subset in $\mathcal{F}$ and each y-type node corresponds to an element in S. All x-type nodes are connected to r and a y-type node is connected to an x-type node if and only if the element belongs to the subset accordingly, i.e,

$$E = \{(r, x_i) \mid \forall\, i\} \cup \{(x_i, y_j) \mid \forall\, i, j,\ s_j \in F_i\}.$$

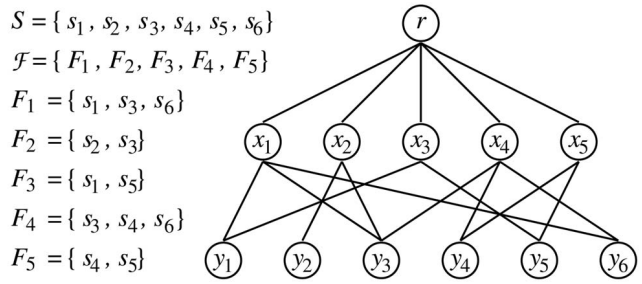


Fig. 2. Example instances P and Q of the minimum set cover problem and the replica placement problem, respectively.

G is essentially the combination of a star and a bipartite graph (see Fig. 2). An instance Q of the replica placement problem is then constructed on graph G by setting $s(x_i) = 1$, $q(x_i) = 1$, and $d(r, x_i) = 1$ for each x-type node, $s(y_i) = 2 \cdot m$ and $q(y_i) = 2 \cdot m$ for each y-type node, and $d(x_i, y_j) = 2 \cdot m$ for each edge $(x_i, y_j) \in E$. Moreover, μ is set to 1. It is obvious that the instance Q is constructed in time polynomial to the size of instance P. In the following, we show that, for any integral bound K, there exists a set cover of size at most K for instance P if and only if there exists a replication strategy of cost at most K for instance Q.

It follows from $\bigcup_{F \in \mathcal{F}} F = S$ that the replication strategy $R^* = \{x_1, x_2, \cdots, x_m\}$ satisfies all QoS requirements. The storage and update costs of R^* are both given by m. Since the replication strategies containing at least one y-node have storage and update costs of at least $2 \cdot m$, they cannot be optimal solutions to the min-scost and min-ucost placement problems. Among the remaining strategies (i.e., those without any y-node), there exists a one-to-one correspondence between the strategies satisfying all QoS requirements and the subsets of $\mathcal{F}$ covering all elements of S. This is because all x-node QoS requirements are satisfied by the origin copy r and placing a replica at x_i satisfies the QoS requirement of y_j if and only if $s_j \in F_i$. Note that the size of each subset of $\mathcal{F}$ is numerically the same as the cost of the associated replication strategy. Thus, there exists a solution to the minimum set cover problem P with a size of at most K if and only if there exists a solution to the min-scost/min-ucost replica placement problem Q with a cost of at most K. Therefore, the min-scost and min-ucost replica placement problems are NP-complete. Since they are special cases of the min-sucost replica placement problem, it follows that the latter is also NP-complete.

Hence, the theorem is proven. □

4.2 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-optimum yardstick in Section 4.3. The results show that the proposed heuristics generally produce close-to-optimal 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 selected, 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.

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 selected. 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.

For l-Greedy-Insert and l-Greedy-Delete, the first step of insertion or removal 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. Moreover, to calculate normalized benefits (for l-Greedy-Insert) and cost reductions (for l-Greedy-Delete) 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 and 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 ranges of $[0, \mathcal{N}]$ and $[0, |V| - \mathcal{N}]$, respectively, the l-Greedy-Insert and l-Greedy-Delete heuristics with a larger l value generally produce 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.

4.3 Performance Evaluation

To evaluate the performance of the above heuristics, we have experimentally compared them against a super-optimum. 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:

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

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

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

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

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

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 QoS requirements are satisfied: 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) guarantee object updates are distributed to all replicas: The link $(v_i, p(v_i))$ is involved in update delivery if a replica is placed at v_i or if any link between v_i and its children are involved in update delivery. A bound on the optimal solution to an integer program is given by the optimal solution to its linear programming relaxation [24]. In our experiments, we relax the integer program to a regular linear program by replacing the last constraint (4) with $\forall i > 0, 0 \le x_i, y_i \le 1$ and compute the optimal solution to the latter. 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 three different models: the Waxman model [25], the Random model [1], and the Tiers model [2]. All three models work 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. The cost of each link is given by the Euclidean distance between the two endpoints. In the Waxman model, a link is created between each pair of nodes (u, v) with the probability $p(u, v) = \beta_1 \cdot e^{-e(u,v)/(\beta_2 \cdot L)}$, where $e(u, v)$ 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 Random model inserts a link between each pair of nodes (u, v) with a fixed probability p. In the Tiers model, the degree of each node is randomly assigned with a specified upper bound m (i.e., the maximum node degree). The links are inserted by first computing a minimum spanning tree connecting the nodes. Each node is then connected to some other nodes in increasing order of distance to satisfy the assigned degree. Interested readers are referred to [25], [1], and [2] for details of these topology models.

The experiments were performed over a wide range of parameter settings. In the default parameter setting, N and s were set at 100 and 1,000, respectively. The connectivity-related parameters of each model were set to keep the total number of links similar across models. Under the default setting, the average number of links in the 100-node networks generated using the three topology models was 315.

A server was assumed to be located at each node in the network. The authoritative origin copy was assumed to be

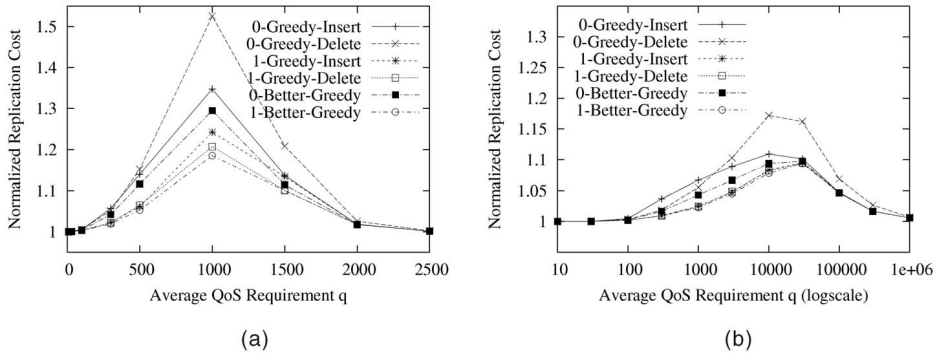


Fig. 3. Performance for different QoS requirements (Waxman model, homogeneous storage costs). (a) Constant QoS requirement distribution q. (b) Uniform QoS requirement distribution $[0, 2q]$.

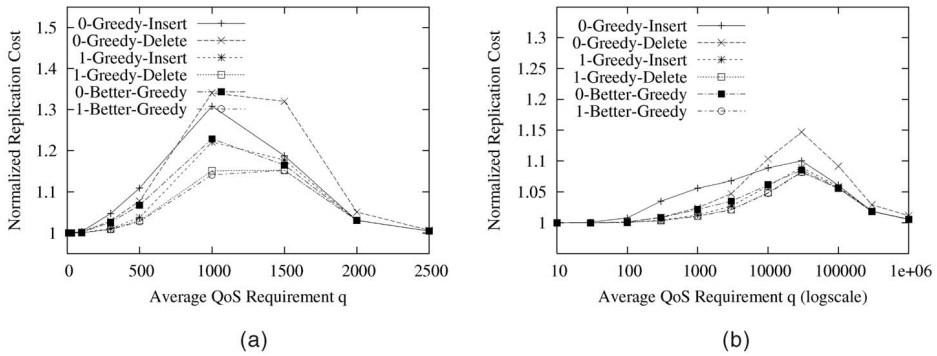


Fig. 4. Performance for different QoS requirements (Waxman model, heterogeneous storage costs). (a) Constant QoS requirement distribution q. (b) Uniform QoS requirement distribution $[0, 2q]$.

located at a randomly selected node (server) 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 distribution 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 4.2 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 $0\text{-}Better\text{-}Greedy$ and $1\text{-}Better\text{-}Greedy$ 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-optimum.

The experimental results for different topology models showed similar performance trends. Due to space limitation, we shall only report the results of the Waxman topology model. Figs. 3 and 4 show the normalized replication costs as a function of q when the storage costs at all nodes were set at 1,000 and were uniformly distributed in [0, 2,000], respectively. It is evident from the superior performance of Better-Greedy heuristics over their simple Greedy peers that the Greedy-Insert and Greedy-Delete heuristics do not dominate each other. Moreover, as expected, the 1-Greedy heuristics generally outperform the 0-Greedy heuristics. These performance trends are consistent over different distributions of QoS requirements and storage costs examined. For all situations reported, the costs of 0-Better-Greedy and 1-Better-Greedy heuristics are within 30 percent and 19 percent of the super-optimum, respectively, under constant distribution of QoS requirements. They are within 10 percent and 9 percent of the super-optimum under uniform distribution of QoS requirements. When the QoS requirements of most nodes are low (i.e., $q \geq 2,000$ for constant distribution and $q \geq 300,000$ 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-optimum. Some 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 of update cost to storage cost. The results are not shown here due to space limitation.

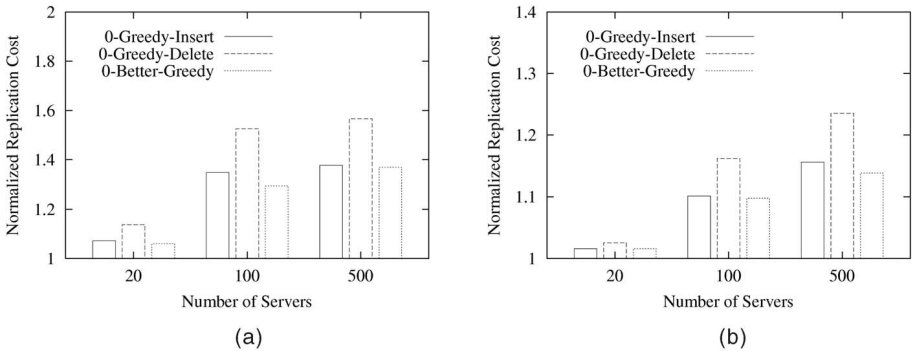


Fig. 5. Performance for different numbers of servers (Waxman model, homogeneous storage costs). (a) Constant QoS requirement distribution q. (b) Uniform QoS requirement distribution $[0, 2q]$.

Fig. 5 shows the experimental results for different numbers of servers in the content distribution system (i.e., the number of nodes modeled in the network) when the storage costs at all nodes were set at 1,000. In these experiments, the connectivity-related parameters of the topology models were set such that the average node degree was kept similar across networks. The average QoS requirement q was set to the values that lead to the highest normalized costs in Fig. 3. We did not simulate the 1-greedy heuristics due to their high computational requirements for 500-server systems. As seen from the figures, the normalized costs of the greedy heuristics increase with the number of servers. However, the increase is mild. In general, the performance difference between systems with 500 and 100 servers is much smaller than that between systems with 100 and 20 servers.

The above experimental results demonstrate that the greedy heuristics are capable of producing good solutions with much lower computation complexity than a brute-force approach.

5 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 services include static proxy hierarchies [27], [4], [21] and en-route content distribution architectures [14], [20], [22]. In these systems, each replica only serves the requests flowing through it under some given routing strategy, which is implemented at either the application or the network level [19]. Regardless of the underlying routing mechanism, the request/delivery paths between all nodes and a given origin server can be represented by a tree topology rooted at the origin server. This tree is used for the routing purposes of both object retrieval and update. Consider again the example system in Fig. 1. Assume all requests are routed through the shortest paths toward the origin server as shown by the arrows in Fig. 6. The requests originating from v_1 and v_2 are served by the origin copy and the requests from v_3 are served by replica v_5. Note that, in replica-blind services, the requests are not necessarily served by the physically nearest replica. They are satisfied by the nearest replica *along* the direction towards the origin copy.

Since the requests are routed on the links in a tree only, a general topology can be simplified to the tree topology for the purpose of replica placement. 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 distance $d(u, v)$ for each link $(u, v) \in E$, and a relative weight α of update cost to storage cost. 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 $R \cup \{r\}$ if $v \notin R \cup \{r\}$, $l(v, R \cup \{r\}) = v$ if $v \in R \cup \{r\}$, and $d(v, l(v, R \cup \{r\}))$ is the distance of the path between v and $l(v, R \cup \{r\})$.

5.1 Optimal Placement with Minimal Storage Cost

This section presents a dynamic programming solution to the min-scost replica placement problem. As shown in Fig. 7, we consider a more generalized problem of placing replicas in a subtree rooted at node x, assuming the lowest

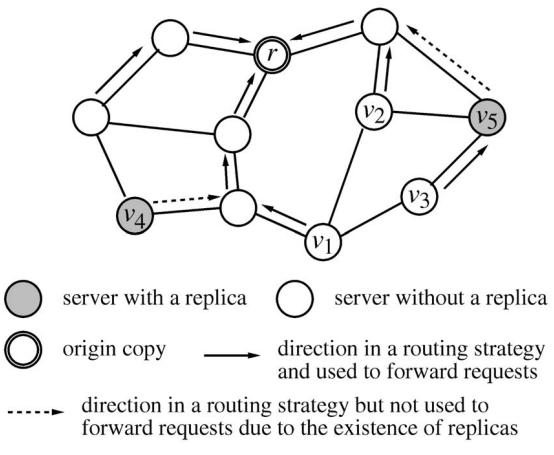


Fig. 6. Replica-blind service model.

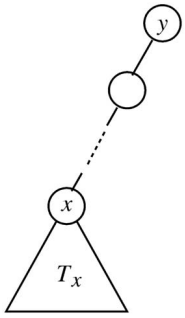


Fig. 7. The (x, y)-optimization problem.

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 T_x be the set of nodes in the subtree of T rooted at x (see Fig. 7). Assume an object replica is placed at y. The problem of finding a replication strategy, i.e., a subset*[1] *$R(x, y) \subseteq T_x$ 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 $v \in T_x$, i.e.,

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

is referred to as 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 Fig. 8). Let T_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.*

1. *If $x \in R(x, y)$, the replication strategy $R'(x, y) = (R(x, y) - T_z) \cup R(z, x)$ is also an optimal solution to the (x, y)-optimization problem.*
2. *Otherwise, if $x \notin R(x, y)$, the replication strategy $R''(x, y) = (R(x, y) - T_z) \cup R(z, y)$ is also an optimal solution to the (x, y)-optimization problem.*

Proof. To prove claim 1, we first show that, if $x \in R(x, y)$, then $R'(x, y) \cup \{y\}$ satisfies the QoS requirement of every node $v \in T_x$. This is because, if $v \in T_z$, it follows from Definition 3 that $d(v, l(v, R(z, x) \cup \{x\})) \leq q(v)$. Since $R(z, x) \subseteq R'(x, y)$ and $x \in R(x, y) - T_z \subseteq R'(x, y)$, we have $R(z, x) \cup \{x\} \subseteq R'(x, y) \cup \{y\}$ and, thus,

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

1. We include x and y as part of the notation for $R(x, y)$ to emphasize an optimal solution to the generalized replica placement problem given x and y.

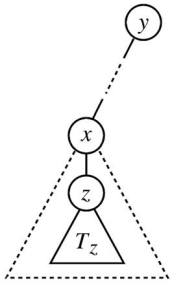


Fig. 8. Rationale of dynamic programming.

On the other hand, if $v \notin T_z$, the lowest ancestor $l(v, R(x, y) \cup \{y\})$ cannot be in T_z. This implies

$$l(v, R(x, y) \cup \{y\}) \in (R(x, y) - T_z) \cup \{y\} \subseteq R'(x, y) \cup \{y\}.$$

Therefore,

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

where the second inequality holds based on Definition 3.

Note that $R(x, y)$ is an optimal solution to the (x, y)-optimization problem. Thus,

$$scost(R(x, y)) \ \leq \ scost(R'(x, y)). \tag{5}$$

Next, we prove that $(R(x, y) \cap T_z) \cup \{x\}$ satisfies the QoS requirements of all nodes in T_z. In fact, for each node $v \in T_z$, since $x \in R(x, y)$ and z is a child of x, the lowest ancestor $l(v, R(x, y) \cup \{y\})$ must be in $T_z \cup \{x\}$. Hence, $l(v, R(x, y) \cup \{y\}) \in (R(x, y) \cap T_z) \cup \{x\}$. Therefore, it follows that

$$\begin{aligned} &d(v, l(v, (R(x, y) \cap T_z) \cup \{x\})) \\ &\quad \leq d(v, l(v, R(x, y) \cup \{y\})) \leq q(v). \end{aligned}$$

Since $R(z, x)$ is an optimal solution to the (z, x)-optimization problem, we have

$$scost(R(z, x)) \ \leq \ scost((R(x, y) \cap T_z)).$$

Note that $R'(x, y)$ can be divided into two disjoint subsets: $R(x, y) - T_z$ and $R(z, x)$. Thus,

$$\begin{aligned} scost(R'(x, y)) &= scost(R(x, y) - T_z) + scost(R(z, x)) \\ &\leq scost(R(x, y) - T_z) + scost(R(x, y) \cap T_z) \\ &= scost(R(x, y)). \end{aligned} \tag{6}$$

Combining (5) and (6), we have

$$scost(R'(x, y)) = scost(R(x, y)).$$

Therefore, $R'(x, y)$ is also an optimal solution to the (x, y)-optimization problem.

To prove claim 2, we start by showing that, if $x \notin R(x, y)$, then $R''(x, y) \cup \{y\}$ satisfies the QoS requirements of all nodes in T_x. For each node $v \in T_x$, if $v \in T_z$, based on Definition 3, the inequality

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

holds. Since $R(z, y) \subseteq R''(x, y)$, we have $R(z, y) \cup \{y\} \subseteq R''(x, y) \cup \{y\}$ and, thus,

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

On the other hand, if $v \notin T_z$, the lowest ancestor $l(v, R(x,y) \cup \{y\})$ cannot be in T_z. It follows that $l(v, R(x,y) \cup \{y\}) \in (R(x,y) - T_z) \cup \{y\} \subseteq R''(x,y) \cup \{y\}$. Therefore,

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

The optimality of $R(x,y)$ to the (x,y)-optimization problem implies that

$$scost(R(x,y)) \ \leq \ scost(R''(x,y)). \tag{7}$$

Next, we prove that $(R(x,y) \cap T_z) \cup \{y\}$ satisfies the QoS requirements of every node $v \in T_z$. Since $x \notin R(x,y)$, the lowest ancestor $l(v, R(x,y) \cup \{y\})$ must be in $T_z \cup \{y\}$. Thus, $l(v, R(x,y) \cup \{y\}) \in (R(x,y) \cap T_z) \cup \{y\}$. As a result, we have

$$\begin{aligned} &d(v, l(v, (R(x,y) \cap T_z) \cup \{y\})) \\ &\quad \leq d(v, l(v, R(x,y) \cup \{y\})) \leq q(v). \end{aligned}$$

It follows from the optimality of $R(z,y)$ to the (z,y)-optimization problem that

$$scost(R(z,y)) \ \leq \ scost((R(x,y) \cap T_z)).$$

Since $R''(x,y)$ can be divided into two disjoint subsets, $R(x,y) - T_z$ and $R(z,y)$, we have

$$\begin{aligned} scost(R''(x,y)) &= scost(R(x,y) - T_z) + scost(R(z,y)) \\ &\leq scost(R(x,y) - T_z) + scost(R(x,y) \cap T_z) \\ &= scost(R(x,y)). \end{aligned} \tag{8}$$

Combining (7) and (8), we have

$$scost(R''(x,y)) = scost(R(x,y)).$$

Thus, $R''(x,y)$ is also an optimal solution to the (x,y)-optimization problem.

Hence, the theorem is proven. □

Theorem 2 implies the following properties:

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.*

1. *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.
2. *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 1 of Theorem 2, the following replication strategies are all optimal solutions to the (x,y)-optimization problem:

$$\begin{aligned} &(R(x,y) - T_{z_1}) \cup R(z_1, x), \\ &(R(x,y) - T_{z_1} - T_{z_2}) \cup R(z_1,x) \cup R(z_2,x), \\ &(R(x,y) - T_{z_1} - T_{z_2} - T_{z_3}) \cup R(z_1,x) \cup R(z_2,x) \cup R(z_3,x), \\ &\cdots\cdots, \\ &\left(R(x,y) - \bigcup_{i=1}^{k} T_{z_i}\right) \cup \left(\bigcup_{i=1}^{k} R(z_i,x)\right). \end{aligned} \tag{9}$$

Since $x \in R(x,y)$, it follows that

$$R(x,y) - \bigcup_{i=1}^{k} T_{z_i} = \{x\}.$$

Thus, (9) $= \{x\} \cup \bigcup_{i=1}^{k} R(z_i,x)$ is an optimal solution to the (x,y)-optimization problem, and claim 1 is proven.

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

$$\begin{aligned} &(R(x,y) - T_{z_1}) \cup R(z_1, y), \\ &(R(x,y) - T_{z_1} - T_{z_2}) \cup R(z_1,y) \cup R(z_2,y), \\ &(R(x,y) - T_{z_1} - T_{z_2} - T_{z_3}) \cup R(z_1,y) \cup R(z_2,y) \cup R(z_3,y), \\ &\cdots\cdots, \\ &(R(x,y) - \bigcup_{i=1}^{k} T_{z_i}) \cup (\bigcup_{i=1}^{k} R(z_i,y)). \end{aligned} \tag{10}$$

Since $x \notin R(x,y)$, we have $R(x,y) - \bigcup_{i=1}^{k} T_{z_i} = \emptyset$. Therefore, (10) $= \bigcup_{i=1}^{k} R(z_i,y)$ is an optimal solution to the (x,y)-optimization problem, and claim 2 is proven.

Hence, the theorem is proven. □

Consider the tree $T^* = (V \cup \{p\}, E \cup \{(r,p)\})$ 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 $d(x,y) \leq q(x)$, no replica needs to be placed at x; otherwise, if $d(x,y) > q(x)$, a replica should be placed at x. For each nonleaf 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:

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

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 postorder 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 $R(x,y)$ and $min\text{-}scost(x,y)$ 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 $N_a(x)$ entries in the forms of $R(x,y)$ and $min\text{-}scost(x,y)$, respectively, where $N_a(x)$ is the number of x's ancestors in T^*. Therefore, the total computation complexity of dynamic programming is given by

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

5.2 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 *interrelated*. 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 3, 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 placed 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*

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

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 $d(w,r) \le q(w)$, the claim is straightforward. Otherwise, if $d(w,r) > q(w)$, consider the distances from w to all its ancestors. Since $d(w,w) = 0 \le q(w)$ and $d(w,r) > q(w)$, there exists a w's ancestor v such that $d(w,v) \le q(w)$ and $d(w,p(v)) > q(w)$. 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 $d(w, l(w, R^* \cup \{r\})) \le d(w,v) \le q(w)$. 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, $ucost(R) \ge ucost(R^*)$. In fact, for each node $v \ne r$, $T_v \cap R^* \ne \emptyset$ implies $T_v \cap R \ne \emptyset$. To show this, 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 $d(w, l(w, R \cup \{r\})) \le q(w)$. Therefore,

$$d(w, l(w, R \cup \{r\})) \le q(w) < d(w,x) + d(x,p(x)) = d(w,p(x)).$$

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 $T_v \cap R \ne \emptyset$. It follows from the definition of $ucost$ (see Section 3) that

$$ucost(R^*) = \mu \cdot \sum_{v \ne r \wedge T_v \cap R^* \ne \emptyset} d(v,p(v)) \le \mu \cdot \sum_{v \ne r \wedge T_v \cap R \ne \emptyset} d(v,p(v)) = ucost(R).$$

Hence, the theorem is proven. □

If $v \in R^*$, for any nonroot ancestor x of v, we have

$$\begin{aligned} &\min_{w \in T_x}(q(w) - d(w,x)) \\ &\le \min_{w \in T_v}(q(w) - d(w,x)) \\ &= \min_{w \in T_v}(q(w) - d(w,v)) - d(v,x) \\ &\le \min_{w \in T_v}(q(w) - d(w,v)) - d(v,p(v)) \\ &< 0 \\ &< d(x,p(x)). \end{aligned}$$

This implies all nonroot ancestors of v are in R^*. Therefore, the optimal solution R^* induces a connected subgraph in tree T.

Note that the values of $\Delta cost(v) = \min_{w \in T_v}(q(w) - d(w,v))$ can be computed in an iterative fashion by a postorder traversal of v in T with the following recurrences:

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

where $Z(v)$ is the set of v's children. Hence, the computational complexity of all $\Delta cost(v)$'s is $O(\sum_{v \in V - \{r\}} N_c(v)) = O(|V|)$, where $N_c(v)$ is the number of v's children. On obtaining the $\Delta cost(v)$'s, 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 $O(|V|)$.

5.3 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 in Section 5.1. 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 T_x be the set of nodes in the subtree of T rooted at x (see Fig. 7). Assume an object replica is placed at y. The objective of the (x, y)-optimization problem is to find a replication strategy, i.e., a subset $R(x, y) \subseteq T_x$ satisfying the QoS requirement of every node $v \in T_x$, i.e.,*

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

with the minimal combined cost

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

where

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

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 $T^* = (V \cup \{p\}, E \cup \{(r, p)\})$ 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 $min\text{-}sucost(x, y)$ be the combined cost of $R(x, y)$. Similar to the analysis in Section 5.1, 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 nonleaf node x in T^*, if $d(x, y) > q(x)$, a replica must be placed at x; otherwise, if $d(x, y) \leq q(x)$, 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

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

and

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

respectively, where $Z(x)$ is the set of x's children in T^*. The recurrences for dynamic programming are given by:

$$min\text{-}sucost(x, y) = \begin{cases} 0 & \text{if } x \text{ is a leaf and } d(x, y) \leq q(x), \\ \alpha \cdot s(x) + (1 - \alpha)\mu \cdot d(x, y) & \text{if } x \text{ is a leaf and } d(x, y) > q(x), \\ \min\{c_1(x, y), c_2(x, y)\} & \text{if } x \text{ is not a leaf and } d(x, y) \leq q(x), \\ c_1(x, y) & \text{if } x \text{ is not a leaf and } d(x, y) > q(x) \end{cases}$$

and

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

All $R(x, y)$s and $min\text{-}sucost(x, y)$s can be computed by a postorder traversal of x in T^* starting from the entries where x is a leaf. $R(r, p)$ is the optimal solution to the min-sucost replica placement problem. The space and time complexities of the dynamic programming algorithm are both given by $O(|V|^2)$.

6 Conclusion

We have investigated the QoS-aware replica placement problem for 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. We have shown that the replica placement problems for replica-aware services are NP-complete. Two families of heuristic algorithms have been proposed and experimentally evaluated. The results show that they perform close to a super-optimum computed from the relaxed linear program. In replica-blind services, the delivery paths with respect to a given origin server form a tree structure. In this case, the optimal solution to the 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 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.

Acknowledgments

Xueyan Tang's work was supported by grants from Nanyang Technological University (Grant Nos. CE-SUG 1/04 and SUG 12/04) and Jianliang Xu's work was supported by grants from the Research Grants Council of the Hong Kong SAR, China (Project Nos. HKBU 2115/05E and HKBU FRG/03-04/II-19).

REFERENCES

[1] GT Internetwork Topology Models (GT-ITM), http://www.cc.gatech.edu/projects/gtitm/, 2000.

[2] K.L. Calvert, M.B. Doar, and E.W. Zegura, "Modeling Internet Topology," *IEEE Comm. Magazine,* vol. 35, no. 6, pp. 160-163, June 1997.

[3] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu, "The State of the Art in Locally Distributed Web-Server Systems," *ACM Computing Surveys,* vol. 34, no. 2, pp. 263-311, June 2002.

[4] A. Chankhunthod, P.B. Danzig, C. Neerdaels, M.F. Schwartz, and K.J. Worrell, "A Hierarchical Internet Object Cache," *Proc. USENIX Ann. Technical Conf.,* pp. 153-163, Jan. 1996.

[5] I. Cidon, S. Kutten, and R. Soffer, "Optimal Allocation of Electronic Content," *Proc. IEEE INFOCOM '01,* pp. 1773-1780, Apr. 2001.

[6] M. Colajanni, P.S. Yu, and V. Cardellini, "Scalable Web-Server Systems: Architectures, Models and Load Balancing Algorithms," *Tutorial Presented at ACM SIGMETRICS '00,* June 2000.

[7] E. Cronin, S. Jamin, C. Jin, A.R. Kurc, D. Raz, and Y. Shavitt, "Constrained Mirror Placement on the Internet," *IEEE J. Selected Areas in Comm.,* vol. 20, no. 7, pp. 1369-1382, Sept. 2002.

[8] 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.

[9] M.L. Fisher and D.S. Hochbaum, "Database Location in Computer Networks," *J. ACM,* vol. 27, no. 4, pp. 718-735, Oct. 1980.

[10] Y. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM SIGMETRICS '00,* pp. 1-12, June 2000.

[11] 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 J.,* vol. 46, no. 4, pp. 378-390, July 2003.

[12] K. Kalpakis, K. Dasgupta, and O. Wolfson, "Optimal Placement of Replicas in Trees with Read, Write, and Storage Costs," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 6, pp. 628-637, June 2001.

[13] M. Karlsson and C. Karamanolis, "Choosing Replica Placement Heuristics for Wide-Area Systems," *Proc. 24th IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 350-359, Mar. 2004.

[14] P. Krishnan, D. Raz, and Y. Shavitt, "The Cache Location Problem," *IEEE/ACM Trans. Networking,* vol. 8, no. 5, pp. 568-582, Oct. 2000.

[15] B. Li, M.J. Golin, G.F. Italiano, X. Deng, and K. Sohraby, "On the Optimal Placement of Web Proxies in the Internet," *Proc. IEEE INFOCOM '99,* pp. 1282-1290, Mar. 1999.

[16] D.A. Menasce, "QoS Issues in Web Services," *IEEE Internet Computing,* vol. 6, no. 6, pp. 72-75, Nov./Dec. 2002.

[17] L. Qiu, V.N. Padmanabhan, and G.M. Voelker, "On the Placement of Web Server Replicas," *Proc. IEEE INFOCOM '01,* pp. 1587-1596, Apr. 2001.

[18] 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, nos. 22-23, pp. 2253-2259, Nov. 1998.

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

[20] P. Rodriguez and S. Sibal, "SPREAD: Scalable Platform for Reliable and Efficient Automated Distribution," *Computer Networks,* vol. 33, nos. 1-6, pp. 33-49, June 2000.

[21] P. Rodriguez, C. Spanner, and E.W. Biersack, "Analysis of Web Caching Architectures: Hierarchical and Distributed Caching," *IEEE/ACM Trans. Networking,* vol. 9, no. 4, pp. 404-418, Aug. 2001.

[22] X. Tang and S.T. Chanson, "Coordinated En-Route Web Caching," *IEEE Trans. Computers,* vol. 51, no. 6, pp. 595-607, June 2002.

[23] X. Tang and J. Xu, "On Replica Placement for QoS-Aware Content Distribution," *Proc. IEEE INFOCOM '04,* Mar. 2004.

[24] V.V. Vazirani, *Approximation Algorithms.* Springer-Verlag, 2001.

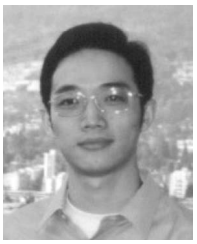
[25] B.M. Waxman, "Routing of Multipoint Connections," *IEEE J. Selected Areas in Comm.,* vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[26] O. Wolfson and A. Milo, "The Multicast Policy and Its Relationship to Replicated Data Placement," *ACM Trans. Database Systems,* vol. 16, no. 1, pp. 181-205, Mar. 1991.

[27] J. Xu, B. Li, and D.L. Lee, "Placement Problems for Transparent Data Replication Proxy Services," *IEEE J. Selected Areas in Comm.,* vol. 20, no. 7, pp. 1383-1398, Sept. 2002.

[28] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," *Proc. IEEE INFOCOM '02,* pp. 1366-1375, June 2002.

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



Xueyan Tang received the BEng degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 1998 and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an assistant professor in the School of Computer Engineering at Nanyang Technological University, Singapore. He has served as a program committee member of IEEE INFOCOM '04 and WWW '05. He is a co-editor of a book entitled *Web Content Delivery* to be published by Springer. His research interests include the Web and Internet (particularly caching, replication, and content delivery), mobile and pervasive computing (especially data management and delivery), wireless sensor networks, peer-to-peer networks, and distributed systems. He is a member of the IEEE.

Jianliang Xu received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, in 1998 and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2002. He is an assistant professor in the Department of Computer Science at Hong Kong Baptist University. His current research interests lie in mobile/pervasive computing, wireless sensor networks, and distributed systems. He has served as a session chair and program committee member for a number of international conferences including IEEE INFOCOM '04, IEEE MDM '04, and ACM SAC '04. He is a coeditor of a book entitled *Web Content Delivery* to be published by Springer. He is an executive committee member of the ACM Hong Kong Chapter. He is a member of the IEEE.