Multimedia Object Placement for Transparent Data Replication

Keqiu Li, Hong Shen, Francis Y.L. Chin, and Weishi Zhang

Abstract—Transparent data replication is a promising technique for improving the system performance of a large distributed network. Transcoding is an important technology which adapts the same multimedia object to diverse mobile appliances; thus, users' requests for a specified version of a multimedia object could be served by a more detailed version cached according to transcoding. Therefore, it is particularly of theoretical and practical necessity to determine the proper version to be cached at each node such that the specified objective is achieved. In this paper, we address the problem of multimedia object placement for transparent data replication. The performance objective is to minimize the total access cost by considering both transmission cost and transcoding cost. We present optimal solutions for different cases for this problem. The performance of the proposed solutions is evaluated with a set of carefully designed simulation experiments for various performance metrics over a wide range of system parameters. The simulation results show that our solution consistently and significantly outperforms comparison solutions in terms of all the performance metrics considered.

Index Terms—Web caching, multimedia, object placement, transcoding, transparent data access, optimization.

1 Introduction

The World Wide Web has become the most successful application on the Internet since it provides a simple way to access a wide range of information and services. However, due to the dramatic growth in demand, considerable access latency is often experienced in retrieving Web objects from the Internet, and popular Web sites are suffering from overload. An efficient way to overcome such deficiencies is Web caching, by which multiple copies of the same object are stored in geographically dispersed caches. An overview of Web caching can be found in [29]. As many mobile appliances are divergent in size, weight, I/O capabilities, network connectivity, and computing power, differentiated devices should be employed to satisfy their diverse requirements. In addition, different presentation preferences from users make this problem more serious. Transcoding, used to transform a multimedia object from one form to another, frequently through trading off object fidelity for size, is a technology that can meet these needs [5], [8], [25], [28].

Transparent data replication [13], [27], [31] has been advocated by both academic and industrial communities because of its low management overheads. Early studies on data replication [7], [30] showed that considerable management overheads were incurred for identifying the optimal locations for the replica before each request was served. For transparent data replication, caches are placed transparently ON both the servers and the clients. Each cache intercepts any request that passes through its associated node and either satisfies the request by sending the requested object to the client or forwards it upstream along the path to the server until it can be satisfied. In [9], [17], [27], [31], the authors have studied the problem of Web object caching (placement and replacement) from different points of view. Here, a Web object can be viewed as a single-version multimedia object. It has also been shown that different versions of the same multimedia object cannot be simply treated as different objects due to the aggregate effect of caching multiple versions of the same multimedia object [6]. Therefore, the solutions that are only applicable for Web objects cannot be simply applied to solve the same problem for multimedia objects. We also studied the problem of multimedia object caching in [16], where the cost loss caused by cache replacement is considered as an input for deriving an optimal solution. Since the cache replacement problem is NP hard, it is difficult to evaluate the cost loss; thus, the solution is firmly dependent on the accuracy of the cost loss evaluation. In this paper, we address the problem of multimedia object placement for transparent data replication, i.e., to determine the exact version of the same multimedia object to be placed at EACH node in the network such that the total access cost is minimized, which is of great importance for cache content initialization. Obviously, our solution is completely independent OF cache replacement, which can be solved by applying the most widely used *LRU* algorithm if necessary. Although dynamic programming is also applied to derive an optimal solution in this paper, the essence of the algorithm is completely novel and there is not any overlap with existing solutions. An example of the problem to be solved in this paper is shown as follows:

Example. Consider a simple network which has only four nodes, v_0, v_1, v_2, and v_3, and a multimedia object which has two versions (A_1 and A_2) as shown in the left part of Fig. 1. In this example, two cases with different access frequencies are also shown in the right part of Fig. 1. We

- *K. Li and W. Zhang are with the College of Computer Science and Technology, Dalian Maritime University, 1 Linghai Road, Dalian, 116026, China. E-mail: keqiu_01@163.com, wszhang@dmu.edu.cn.*
- *H. Shen is with the School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia. E-mail: hong@cs.adelaide.edu.au.*
- *F.Y.L. Chin is with the Department of Computer Science and Information Systems, University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: chin@cs.hku.hk.*

Manuscript received 11 Oct. 2005; revised 22 Feb. 2006; accepted 17 Mar. 2006; published online 27 Dec. 2006.
Recommended for acceptance by K. Nakano.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0428-1005.

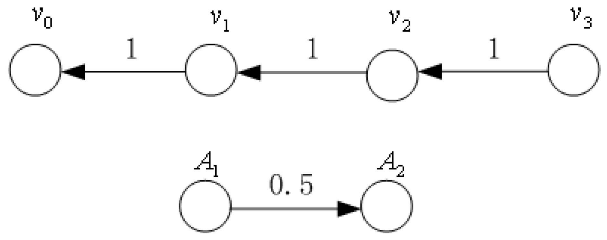


Fig. 1. A simple example.

assume that the transmission cost on each edge is 1 and the transcoding cost from A_1 to A_2 is 0.5.

We can easily obtain the following results as shown in Table 1: The column denotes the placement decision and the relevant access cost for different cases. For instance, the fourth column denotes the decision is placing A_2, A_1, A_2 at nodes v_1, v_2, and v_3, respectively, for Case 1 and the access cost is 2.5. Let us consider the last instance of Case 2 in Table 1, i.e., placement A_1, A_1, and A_2 at nodes v_1, v_2, and v_3, respectively. The access costs for requesting versions A_1, A_1, and A_2 at nodes v_1, v_2, and v_3 are zero, while the access cost for version A_2 at node v_1 is $1 \times 0.5 = 0.5$ (transcoding from version A_1 at node v_1), the access cost for version A_2 at node v_2 is $2 \times 0.5 = 1$ (transcoding from version A_1 at node v_2), and, finally, the access cost for version A_1 at node v_3 is $3 \times 1 = 3$ (served by version A_1 at node v_2). So, the total access cost is 4.5.

Obviously, the access cost for each case is different with different placement. So, the same version should not be simply cached at each node that the request passes by as it returns to the client.

From the above example, we can conclude that the placement of different versions of a multimedia object at different nodes has a significant influence on network performance. Therefore, the study of the multimedia object placement strategies for transparent data replication has made significant contributions to both theory and practice. In this paper, we address the problem of multimedia object placement for transparent data replication, i.e., determining exactly which version should be placed at each node such that the total access cost is minimized. The main contributions of this paper are summarized as follows:

- We present a model for the problem of multimedia object placement for transparent data replication, formulated as an optimization problem. In our model, multimedia object placement decisions are made based on both transcoding and transmission cost.
- We propose dynamic programming-based solutions to compute the optimal versions to be cached for different cases.
- We give an extensive and detailed analysis on the proposed solutions and show that our solutions are optimal and low-cost.
- We evaluate our model on various performance metrics through extensive simulation experiments. The implementation results show that our solution consistently and significantly outperforms existing solutions.

The rest of this paper is organized as follows: Section 2 introduces some preliminary concepts used for latter analysis. In Section 3, we formulate the problem of multimedia object placement for transparent data replication followed by related work. Section 4 presents optimal solutions for the different cases considered. The simulation model and performance evaluation are described in Sections 5 and 6, respectively. Section 7 summarizes our work and concludes the paper.

TABLE 1
Access Costs with Different Placement

CASE	1			2		
v_1	A_1	A_2	A_2	A_1	A_2	A_1
v_2	A_1	A_2	A_1	A_1	A_2	A_1
v_3	A_1	A_2	A_2	A_1	A_2	A_2
Access Cost	1.5	6	2.5	5.5	12	4.5

2 Preliminary Concepts

To facilitate our analysis later in this paper, we introduce some preliminary concepts in this section. Object transcoding is described in Section 2.1, and the transparent data replication model is depicted in Section 2.2.

2.1 Web Object Transcoding

Transcoding is used to transform a multimedia object from one form to another, frequently trading off object fidelity for size for the prevailing operating environments. The relationship among different versions of a multimedia object can be expressed by a weighted transcoding graph [6]. An example of such a graph is shown in Fig. 2.

In Fig. 2, we can see that the original version A_1 can be transcoded to each of the less detailed versions A_2, A_3, A_4, and A_5. It should be noted that not every A_i can be transcoded to A_j since it is possible that A_i does not contain enough content information for the transcoding from A_i to A_j. In the example, transcoding cannot be executed between A_4 and A_5 due to insufficient content information. The transcoding cost of a multimedia object from A_i to A_j is denoted by $t(A_i, A_j)$. Obviously, $t(A_i, A_i) = 0$. The number beside each edge in Fig. 2 is the transcoding cost from one version to another. For example, $t(A_1, A_2) = 6$ and $t(A_3, A_4) = 8$. If a version cannot be transcoded from another version, we consider the transcoding cost as infinity. For instance, $t(A_2, A_1) = \infty$ and $t(A_4, A_5) = \infty$. If version A_j can be transcoded from

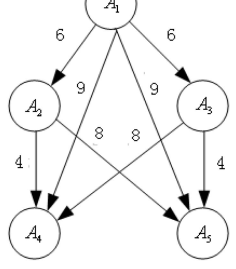


Fig. 2. A weighted transcoding graph.

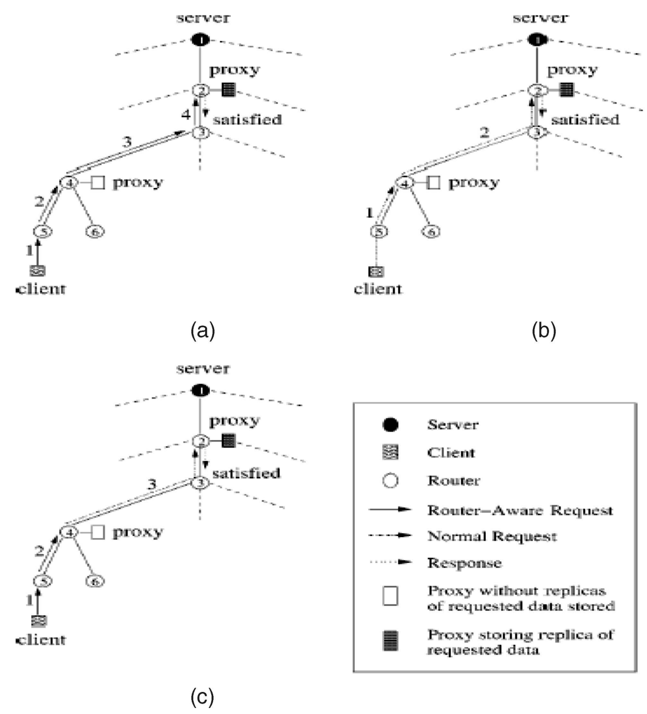


Fig. 3. Transparent data replication model. (a) En-route model. (b) Hierarchical model. (c) Hybrid model.

version A_i through version A_k with $i < k < j$, then $t(A_i, A_j) \leq t(A_i, A_k) + t(A_k, A_j)$ (triangularity property) because version A_k is always an option if the transcoding cost $t(A_i, A_j)$ is too large. For example, $t(A_1, A_4) \leq \min\{t(A_1, A_2) + t(A_2, A_4), t(A_1, A_3) + t(A_3, A_4)\}$. $\Phi(A_i)$ is the set of all the versions that can be transcoded from A_i, including A_i. For example, $\Phi(A_1) = \{A_1, A_2, A_3, A_4, A_5\}$, $\Phi(A_2) = \{A_2, A_4, A_5\}$, and $\Phi(A_4) = \{A_4\}$.

2.2 Transparent Data Replication Model

As mentioned in previous sections, transparent data replication is a promising technique for improving the system performance of a large distributed network, which can overcome the management overheads that are incurred in early studies for identifying the optimal locations for the replica before each request is served. In general, there are two basic approaches for transparent data replication: en-route caching [13], [27] and hierarchical caching [23], [24]. Similar to [31], we address the problem of multimedia object placement on a hybrid transparent data replication model by combining both en-route and hierarchical caching. In this model, transcoding proxies are organized in a hierarchical manner as in hierarchical caching, and each request from a client is routed on the access path from the client to the server as in en-route caching. Fig. 3 shows an example of such a transparent data replication model [31].

3 Problem Formulation and Related Work

In this section, we first introduce the problem formulation followed by related work.

3.1 Problem Formulation

The network topology in this paper is modeled as a graph $G = (V, E)$, where $V = \{v_0, v_1, \cdots, v_n\}$ is the set of nodes or vertices, and E is the set of edges or links. For a multimedia object O, we assume that it has m versions: $A_1, A_2, \cdots, A_m$. For each version of object O, we associate each link $(u, v) \in E$ with a nonnegative cost $L_k(u, v)$, which is defined as the cost of sending a request for version A_k and the relevant response over the link (u, v). In particular, $L_k(u, u) = 0$. If a request goes through multiple network links, the cost is the sum of the cost on all these links. The cost in our analysis is calculated from a general point of view. It can be different performance measures such as delay, bandwidth requirement, and access latency, or a combination of these measures. Let $f_{i,j}$ be the access frequency of version A_j from node v_i.

Now, we start to formulate the problem of multimedia object placement for data transparent replication (*MOP problem*). Consider the snapshot when a request for a specified version of a multimedia object is being served (see Fig. 4). Here, v_0 denotes the content server which contains all versions of object O. v_n is the client and $v_1, v_2, \cdots, v_{n-1}$ are the nodes on the path from the client to the server. We can see that a request for a version of a multimedia object from a node can be satisfied either by this node or by upstream nodes (transcoding if necessary) until it arrives at the server at which no transcoding is necessary. Therefore, the total access cost can be decomposed into two parts: transcoding cost and transmission cost. Our objective is to find the exact version of a multimedia object to be placed at each node on the path from v_1 to v_n so that the total access cost is minimized. Note that all requests at node v_0 can be satisfied at zero cost. If we denote A_{d_i} $(d_i \in \{1, 2, \cdots, m\})$[1] as the version cached at node v_i, then the total access cost of caching $A_{d_1}, A_{d_2}, \cdots, A_{d_n}$, denoted by $C(X)$, is defined as follows:

$$C(X) = \sum_{i=1}^{n} \sum_{j=1}^{m} f_{i,j} \min_{0 \leq k \leq i} \{L_j(v_i, v_k) + T(A_{d_k}, A_j)\}, \tag{1}$$

where $X = (A_{d_1}, A_{d_2}, \cdots, A_{d_n})$ and

$$T(A_{d_k}, A_j) = \begin{cases} 0 & \text{if } k = 0 \\ t(A_{d_k}, A_j) & \text{if } k \neq 0. \end{cases}$$

Obviously, our objective is to obtain $X^* = (A_{d_1^*}, A_{d_2^*}, \cdots, A_{d_n^*})$ such that $C(X^*) = \min_X \{C(X)\}$.

Before we solve the MOP problem based on the cost function as given in (1), we can make the following assumptions:

- *Assumption* 1. $L_j(v_i, v_k) = (i - k)L$ for all $1 \leq j \leq m$ as there are $i - k$ links on the path between nodes v_i and v_k, and the cost on each link for each version is L.
- *Assumption* 2. The transcoding graph is a linear array and the transcoding cost between any two adjacent versions is constant, i.e.,

  $$(A_i, A_j) = \sum_{k=i}^{j-1} t(A_k, A_{k+1}) = (j - i)^+ T,$$

  where $x^+ = x$ if $x \geq 0$, else $x^+ = \infty$.
- *Assumption* 3. $(\delta - 1)T \leq L$, and $\delta T > L$ for some positive integer δ.

If there does not exist δ such that *Assumption* 3 can be satisfied, i.e., $L \gg T$ or $T \gg L$. Obviously, these are two

1. A_{d_0} is a virtual version such at $T(A_{d_0}, A_j) = 0$ for $1 \leq j \leq m$.

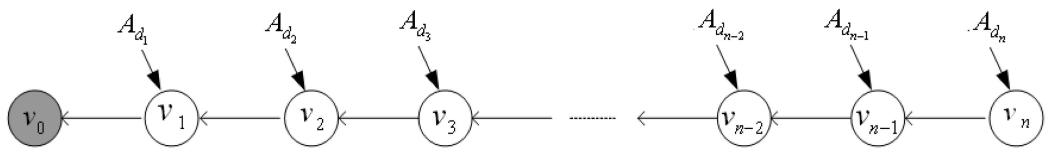


Fig. 4. System model for multimedia object caching.

special cases. If $L \gg T$, then version A_{d_i} should be cached so that no transmission cost is necessary to incur, where $d_i = \min\{j | f_{i,j} > 0, 1 \le j \le m\}$. If $T \gg L$, this case is not trivial and is equivalent to the en-route caching problem of caching m objects on a linear network of n nodes, where the transcoding cost is prohibited.

With the above assumptions, the MOP problem can be simplified as follows:

$$C(X) = \sum_{i=1}^{n} \sum_{j=1}^{m} f_{i,j} \min\left\{ \min_{1 \le k \le i} \{(i-k)L + (j-d_k)^{+} T\}, iL \right\}. \tag{2}$$

3.2 Related Work

Transparent data replication has been advocated by both academic and industrial communities because of its low management overheads. The performance of transparent data replication mainly depends on two issues, i.e., cache location/proxy placement and cache content management.

A lot of work has been done for solving the problem of cache location/proxy placement for transparent data replication. In [14], the authors addressed the problem of proxy placement for linear topology. In [10], [11], [15], [18], the authors studied the problem of proxy placement for tree networks from different points of consideration. In [13], the authors addressed the problem of cache location from a general point of view and analyzed several cases for different network topologies. In [19], [20], we addressed the problem of transcoding proxy[2] placement for tree and linear topologies, respectively.

In this paper, we concentrate on the second issue, i.e., cache content management. Currently, two kinds of objects, i.e., Web objects and multimedia objects, are studied in cache content management for transparent data replication. Since a multimedia object has at least two different versions, from this point of view, a Web object can be viewed as a single-version multimedia object. In [6], the authors have shown that it is not applicable to view different versions of the same multimedia object as different Web objects since the aggregate profit of caching multiple versions of the same multimedia object is not the simple summation of their individual profits. The current research on cache content management for transparent data replication can be classified from different points of view. If it is classified according to the kinds of objects, then it can be classified into two groups, i.e., Web object-based research and multimedia object-based research. If it is classified according to the kinds of network topologies, then it can also be classified into two groups, i.e., linear topology-based research and tree topology-based research. If it is classified according to the kinds of actions, then it can be classified into three groups, i.e., object placement-based research, object replacement-based research, and object content update-based research. If it is classified according to the kinds of cache storages, then it can be classified into two groups, i.e., limited storage-based research and unlimited storage-based research. Table 2 clearly illustrates the differences among the current research for solving the problem of cache content management for transparent data replication, where we denote Web Object by *WO*, Multimedia Object by *MO*, Linear Topology by *LT*, Tree Topology by *TT*, Object Placement by *OP*, Object Replacement by *OR*, Object Content Update by *OCU*, Limited Cache Storage by *LCS*, and Unlimited Cache Storage by *UCS*. In Table 2, "Y" means that the item on the column is addressed in the paper on the row.

From Table 2, we can see that in [9], [12], [17], [21], [27], [31], the authors have studied the problem of Web object caching (placement and replacement) from different points of view. Here, a Web object can be viewed as a single-version multimedia object. It has also been shown that different versions of the same multimedia object cannot be simply treated as different objects due to the aggregate effect of caching multiple versions of the same multimedia object [6]. Therefore, the solutions that are only applicable for Web objects can not be simply applied to solve the same problem for multimedia objects. We also studied the problem of multimedia object caching in [16], where the cost loss caused by cache replacement is considered as an input for deriving an optimal solution. Since the cache replacement problem is NP hard, it is difficult to evaluate the cost loss; thus, the solution is firmly dependent on the accuracy of the cost loss evaluation. In this paper, we address the problem of multimedia object placement for transparent data replication, i.e., to determine the exact version of the same multimedia object to be placed at EACH node in the network such that the total access cost is minimized, which is of great importance for cache content initialization. Obviously, our solution is completely independent of the cache replacement, which can be solved by applying the most widely used LRU algorithm if necessary. Although dynamic programming is also applied to derive an optimal solution in this paper, the essence of the algorithm is completely novel and there is not any overlap with existing solutions.

Table 3 shows the comparison of our research on transparent data replication.

4 Dynamic Programming-Based Solutions

In this section, we first consider the case of $n = 1$, i.e., there is only one node, and then discuss the case of $n > 1$.

2. A transcoding proxy is a proxy with the functionality of transcoding.

TABLE 2
Classification of Current Research

Current Research	Object Type		Network Topology		Action			Storage	
	WO	*MO*	*LT*	*TT*	*OP*	*OR*	*OCU*	*LCS*	*UCS*
This Paper		Y	Y		Y			Y	Y
[16]		Y		Y	Y	Y		Y	
[27]	Y		Y		Y	Y		Y	
[9]	Y			Y	Y		Y	Y	Y
[12,17,21]	Y			Y	Y	Y		Y	
[31]	Y			Y	Y		Y		Y

TABLE 3
Comparison of Our Research

Our Research	Problem Description
Proxy Placement	
[18]	Proxy Placement for Tree Networks
[19]	Transcoding Proxy Placement for Tree Networks
[20]	Transcoding Proxy Placement for Linear Topology
Cache Content Management	
[17,21]	Web Object Caching for Tree Networks and Autonomous Systems
[16]	Multimedia Object Caching for Tree Networks
This Paper	Multimedia Object Placement for Linear Topology

4.1 The Case of $n = 1$

Before presenting the optimal solutions, we give a brief explanation of the significance for solving the MOP problem for the case of $n = 1$. For the cache replacement problem, a crucial thing is to determine the objects to be removed so that the cost loss is minimized and the free space is enough to accommodate the new object. The following solutions are of great importance in deciding the objects to be removed for transcoding-enabled cache replacement problem.

First, we begin by computing the access cost of caching only one version A_k at node v_1 with $1 \le k \le m$. Intuitively, all the requests for version A_i with $i < k$ will be handled by server v_0, while some of the requests for A_i with $i \ge k$, depending on the transcoding cost and the transmission cost, will be taken care of by transcoding from version A_k. Based on the cost function defined by (2), the total access cost of caching only version A_k at node v_1 is computed as follows:

$$C(A_k) = \sum_{i=1}^{k-1} f_{1,i} L + \sum_{i=k}^{m} f_{1,i} \min\{(i-k)T, L\}. \tag{3}$$

Since version A_k is cached at node v_1, we can see (with Assumption 3) that δ is such a parameter that the request for version A_i will be served by the local node if $0 < i - k < \delta$, and the request for version A_i will be served by the server if $i - k \ge \delta$.

Based on (3), $C(A_k)$ can be further defined as follows:

$$C(A_k) = \begin{cases} \sum_{i=1}^{k-1} f_{1,i} L + \sum_{i=k}^{k+\delta-1} f_{1,i}(i-k)T \\ + \sum_{i=k+\delta}^{m} f_{1,i} L & \text{if } k+\delta \le m \\ \sum_{i=1}^{k-1} f_{1,i} L + \sum_{i=k}^{m} f_{1,i}(i-k)T & \text{if } k+\delta > m. \end{cases} \tag{4}$$

It is easy to see that $C(A_1)$ can be calculated in $O(m)$ time.

$$C(A_{k+1}) = \begin{cases} C(A_k) + f_{1,k}L - \sum_{i=k+1}^{k+\delta-1} f_{1,i}T \\ +f_{1,k+\delta}((\delta-1)T - L) & \text{if } k+\delta \le m \\ C(A_k) + f_{1,k}L - \sum_{i=k+1}^{m} f_{1,i}T & \text{if } k+\delta > m \end{cases}$$
$$= C(A_k) + E(k),$$

where

$$E(k) = \begin{cases} f_{1,k}L - \sum_{i=k+1}^{k+\delta-1} f_{1,i}T \\ +f_{1,k+\delta}((\delta-1)T - L) & \text{if } k+\delta \le m \\ f_{1,k}L - \sum_{i=k+1}^{m} f_{1,i}T & \text{if } k+\delta > m \end{cases}$$

and

$$E(k+1) = \begin{cases} E(k) - f_{1,k}L + f_{1,k+1}(L+T) \\ -f_{1,k+\delta}(\delta T - L) + f_{1,k+\delta+1}(\delta-1) \\ T - f_{1,k+\delta+1}L & \text{if } k+\delta < m, \\ E(k) - f_{1,k}L + f_{1,k+1}(L+T) \\ -f_{1,k+\delta}(\delta T - L) & \text{if } k+\delta = m, \\ E(k) - f_{1,k}L + f_{1,k+1}(L+T) & \text{if } k+\delta > m. \end{cases}$$

Thus, each $C(A_2), C(A_3), \cdots, C(A_m)$ can be done in constant time. Therefore, the MOP problem can be solved in $O(m)$ time. With regard to the time complexity of solving the MOP problem, we have the following theorem:

Theorem 1. *Based on the cost function as given in (3), the MOP problem for $\{A_1, A_2, \cdots, A_m\}$ by caching only one version (i.e., $n = 1$) can be solved in $O(m)$ time.*

Proof. Since the cost function as given in (4) is equivalent to the cost function as given in (3) and the MOP problem based on the cost function as given in (4) can be solved in $O(m)$ time, the MOP problem based on the cost function as given in (3) can also be solved in $O(m)$ time. □

The second step is to extend the above solution to compute the optimal solution for caching two versions, A_{k_1} and A_{k_2}, at the same time at node v_1.

Suppose that A_{k_1} and A_{k_2} are the two optimal versions to be cached. The key observation here is that A_{k_1} is also an optimal solution for the problem with $\{A_1, A_2, \cdots, A_{k_2-1}\}$ if $k_1 < k_2$ because the requests for $\{A_{k_2}, A_{k_2+1}, \cdots, A_m\}$ can only be served by A_{k_2}. With regard to this observation, we have the following lemma:

Lemma 1. *Assume that A_{b_p} and A_{b_q} are the optimal solutions for the problem of caching only one version from the set of $\{A_1, A_2, \cdots, A_{p-1}\}$ and $\{A_1, A_2, \cdots, A_{q-1}\}$, respectively, then we have $b_p \le b_q$ if $p < q$.*

Proof. Without loss of generality, it is sufficient for us to prove that $b_p \le b_{p+1}$, where $1 \le b_p \le p-1$ and $1 \le b_{p+1} \le p$. The proof is by contradiction. Assume that we have $b_p > b_{p+1}$. As A_{b_p} is the optimal version to be cached, we have $C_{1,p}(A_{b_p}) < C_{1,p}(A_{b_{p+1}})$. Let $C_{1,p}(A_i)$ denote the access cost of caching A_i for the MOP problem with $\{A_1, A_2, \cdots, A_{p-1}\}$. From the definition of the access cost function $C_{1,p}$ as given in (3), adding A_p to the set $\{A_1, A_2, \cdots, A_{p-1}\}$ will increase both $C_{1,p}(A_{b_p})$ and $C_{1,p}(A_{b_{p+1}})$ by $f_{1,p}\min\{(p-b_p)T, L\}$ and $f_{1,p}\min\{(p-b_{p+1})T, L\}$, respectively. The increase to $C_{1,p}(A_{b_{p+1}})$ is no less than that to $C_{1,p}(A_{b_p})$ because $b_p > b_{p+1}$. So, we have $C_{1,p+1}(A_{b_p}) < C_{1,p+1}(A_{b_{p+1}})$, which contradicts the fact that $C_{1,p+1}(A_{b_{p+1}})$ is the minimum access cost of caching $A_{b_{p+1}}$ for the problem with $\{A_1, A_2, \cdots, A_{p-1}, A_p\}$. □

Based on Lemma 1, we can see that the feasible range of the optimal solution for the problem with $\{A_1, A_2, \cdots, A_q\}$ can be reduced if the optimal version for the problem with $\{A_1, A_2, \cdots, A_p\}$ has been obtained. So is the other case when the optimal solution for the problem with $\{A_1, A_2, \cdots, A_q\}$ is known; the feasible range of the optimal solution for the problem with $\{A_1, A_2, \cdots, A_p\}$ is also reduced. Therefore, we can find A_{b_p} and compute $C_{1,p}(A_p)$ by divide and conquer.

Let $D^{(k)}_{p,q}$ denote the minimum access cost of caching k versions for the MOP problem with $q-p$ versions, i.e., $A_p, A_{p+1}, \cdots, A_{q-1}$, where $1 \le p < q \le m$. Thus, $D^{(1)}_{1,p} = C_{1,p}(A_{b_p})$ and $D^{(1)}_{1,m+1} = \min_{1 \le k \le m}\{C_{1,m+1}(A_k)\}$. Based on Lemma 1, we have the following theorem on the time complexity of computing $D^{(1)}_{1,p}$ for $1 < p \le m$:

Theorem 2. *All the p MOP problems for $\{A_1, A_2, \cdots, A_p\}$, where $1 \le p \le m$, i.e., $D^{(1)}_{1,p}$ for $1 < p \le m$, can be computed in $O(m \log m)$ time.*

Proof. Assume that there exists an integer θ such that $m = 2^\theta$.

Based on Theorem 1, we can compute $D^{(1)}_{1,\frac{1}{2}m}$ in $O(m)$ time. Assume that $A_{b_{\frac{m}{2}}}$ is the optimal solution for the problem of caching only one version with $\{A_1, A_2, \cdots, A_{\frac{m}{2}-1}\}$; then, we can find the optimal solution for the problem of caching only one version for $\{A_1, A_2, \cdots, A_{\frac{m}{4}}\}$ in $O(m)$ time. Similarly, $D^{(1)}_{1,\frac{3m}{4}}$ can also be computed by solving the problem of caching only one version with $\{A_1, A_2, \cdots, A_{\frac{3m}{4}-1}\}$. As we have already computed $C_{1,\frac{m}{2}}(A_y)$, where $y = \min(b_{\frac{m}{2}}, \frac{m}{2}-1)$, we can base on this result to compute $C_{1,\frac{3m}{4}}(A_y)$ for $\{A_1, A_2, \cdots, A_{\frac{3m}{4}-1}\}$ (by adding at most $\frac{m}{4}$ terms to $C_{1,\frac{m}{2}}(A_{\frac{m}{2}-1})$. We then compute $C_{1,\frac{3m}{4}}(A_y), C_{1,\frac{3m}{4}}(A_{y+1}), \cdots, C_{1,\frac{3m}{4}}(A_{\frac{3m}{4}-1})$ in at most $O(\frac{3m}{4} - y)$ time. So, it takes at most $O(m)$ time to compute $D^{(1)}_{1,\frac{m}{4}}$ and $D^{(1)}_{1,\frac{3m}{4}}$. According to the similar decomposition, $D^{(1)}_{1,\frac{m}{8}}$, $D^{(1)}_{1,\frac{3m}{8}}$, $D^{(1)}_{1,\frac{5m}{8}}$, and $D^{(1)}_{1,\frac{7m}{8}}$ can all be solved in $O(m)$ time. To be precise, let $A_{z_1}, A_{z_2}, A_{z_3}$ be the optimal versions for $\{A_1, A_2, \cdots, A_{\frac{m}{4}-1}\}$, $\{A_1, A_2, \cdots, A_{\frac{m}{2}-1}\}$, and $\{A_1, A_2, \cdots, A_{\frac{3m}{4}-1}\}$, respectively. The first step is to compute $C_{1,\frac{m}{8}}(A_1)$, and then $C_{1,\frac{3m}{8}}(A_{z_1})$, $C_{1,\frac{5m}{8}}(A_{z_2})$, and $C_{1,\frac{7m}{8}}(A_{z_3})$ from $C_{1,\frac{m}{4}}(A_{z_1})$, $C_{1,\frac{m}{2}}(A_{z_2})$, and $C_{1,\frac{3m}{4}}(A_{z_3})$, respectively. As the computation of each item takes $O(\frac{m}{8})$ time, this step takes $O(m)$ time in total. Then, we can search the optimal solutions for $\{A_1, A_2, \cdots, A_{\frac{m}{8}-1}\}$, $\{A_1, A_2, \cdots, A_{\frac{3m}{8}-1}\}$, $\{A_1, A_2, \cdots, A_{\frac{5m}{8}-1}\}$, and $\{A_1, A_2, \cdots, A_{\frac{7m}{8}-1}\}$ in the ranges $(1, \min\{z_1, \frac{m}{8}-1\})$, $(z_1, \min\{z_2, \frac{3m}{8}-1\})$, $(z_2, \min\{z_3, \frac{5m}{8}-1\})$, and $(z_3, \frac{7m}{8}-1)$, respectively. Since each step takes constant time, all these searches take no more than $O(m)$ time in total. After repeating this process $\log m$ times, we can finish computing $D_{1,p}^{(1)}$ for $1 < p \leq m$. □

Now, we can accomplish the problem of caching two versions in the following three steps:

- *Step* 1: Evaluate $D_{1,p}^{(1)}$ for $1 < p \leq m$, where $D_{1,p}^{(1)}$ denotes the minimum access cost of caching only one version for the MOP problem with $p-1$ versions, i.e., $A_1, A_2, \cdots, A_{p-1}$. In particular, $D_{1,m+1}^{(1)} = \min_{1 \leq k \leq m} \{C_{1,m+1}(A_k)\}$.
- *Step* 2: Evaluate $\overline{D}_p$ for $2 \leq p \leq m$, where $\overline{D}_p$ is the access cost for versions $A_p, A_{p+1}, \cdots, A_m$ if A_p is cached at node v_1. $\overline{D}_p$ is defined as follows:

$$\overline{D}_p = \begin{cases} \sum_{i=p}^{p+\delta-1} f_{1,i}(i-p)T + \sum_{i=p+\delta}^{m} f_{1,i}L & \text{if } p+\delta \leq m \\ \sum_{i=p}^{m} f_{1,i}(i-p)T & \text{if } p+\delta > m. \end{cases}$$

- *Step* 3: Compute $D_{1,m}^{(2)}$, where $D_{1,m}^{(2)}$ is the minimum access cost of caching two versions for the problem with $\{A_1, A_2, \cdots, A_m\}$. $D_{1,m}^{(2)}$ is calculated as follows:

$$D_{1,m}^{(2)} = \min_{2 \leq p \leq m} \{D_{1,p}^{(1)} + \overline{D}_p)\}.$$

The following theorem shows that $D_{1,m}^{(2)}$ is the minimum access cost of caching two versions of the MOP problem:

Theorem 3. $D_{1,m}^{(2)}$ *is the minimum access cost of caching two versions for the MOP problem.*

Proof. Assume that

$$D_{1,m}^{(2)} = D_{1,p^*}^{(1)} + \overline{D}_{p^*} = \min_{2 \leq p \leq m} \{D_{1,p}^{(1)} + \overline{D}_p)\}.$$

It is obvious from the computation of $D_{1,m}^{(2)}$ that b_{p^*} and A_{p^*} are the two versions which achieve the minimum access cost of caching two versions, where $D_{1,p^*}^{(1)} = C_{1,p^*}(b_{p^*})$. □

The following theorem shows the time complexity of computing $D_{1,m}^{(2)}$:

Theorem 4. $D_{1,m}^{(2)}$ *can be computed in* $O(m \log m)$ *time.*

Proof. Since *Step* 1 can be computed in $O(m \log m)$ time (Theorem 2) and *Steps* 2 *and* 3 both take $O(m)$ time, the total time for computing $D_{1,m}^{(2)}$ is $O(m \log m)$. □

After we have calculated $D_{1,p}^{(1)}$ for $1 \leq p \leq m$ in *Step* 1, we can obtain $D_{1,p}^{(2)}$ for all $2 \leq p \leq m$ in another $O(m \log m)$ time by divide and conquer, where $D_{1,p}^{(2)}$ is the minimum access cost of caching only two versions for the problem with $p-1$ versions. The main idea is similar to Lemma 1 in the finding of $D_{1,p}^{(1)}$. Assume that $A_{b_{p_1}}$ and $A_{b_{p_2}}$ with $1 \leq b_{p_1} < b_{p_2} < p$ are the two optimal versions cached in node v_1 for $A_1, A_2, \cdots, A_{p-1}$ to achieve the optimal access cost $D_{1,p}^{(2)}$. Similarly, $A_{b_{q_1}}$ and $A_{b_{q_2}}$ with $1 \leq b_{q_1} < b_{q_2} < q$ are the two optimal versions cached in node v_1 for $A_1, A_2, \cdots, A_{q-1}$ to achieve the optimal access cost $D_{1,q}^{(2)}$. We can show with a similar argument with Lemma 1 that $b_{p_2} \leq b_{q_2}$ if $p < q$ and this property limits the range of searching for the optimal solutions. As in Theorem 2, the two optimal solutions in $D_{1,\frac{m}{2}}^{(2)}$ can be found in $O(m)$ time after knowing the optimal versions of $D_{1,p}^{(1)}$ for $1 < p \leq m$; then, $D_{1,\frac{m}{4}}^{(2)}$ and $D_{1,\frac{3m}{4}}^{(2)}$ in another $O(m)$ time; then, $D_{2,\frac{m}{8}}^{(2)}$, $D_{1,\frac{3m}{8}}^{(2)}$, $D_{1,\frac{5m}{8}}^{(2)}$, and $D_{1,\frac{7m}{8}}^{(2)}$ in another $O(m)$ time until $D_{1,p}^{(2)}$ for $2 < p \leq m$ are found after $\log m$ times. Therefore, the minimum access cost of caching three versions, denoted by $D_{1,m}^{(3)}$, can be computed similarly, i.e., $D_{1,m}^{(3)} = \min_{3 \leq p \leq m} \{D_{1,p}^{(2)} + \overline{D}_p)\}$, with at most $O(m \log m)$ time (similar to Theorem 5). Using the same idea, we can solve the problem of caching K versions in $O(Km \log m)$ time.

Let $D_{1,m}^{(K)}$ denote the minimum access cost of caching K versions from m versions, i.e., $A_1, A_2, \cdots, A_m$; then, we have the following theorem on the time complexity of computing $D_{1,m}^{(K)}$:

Theorem 5. $D_{1,m}^{(K)}$ *can be computed in* $O(Km \log m)$ *time.*

Proof. Based on the above analysis, we have

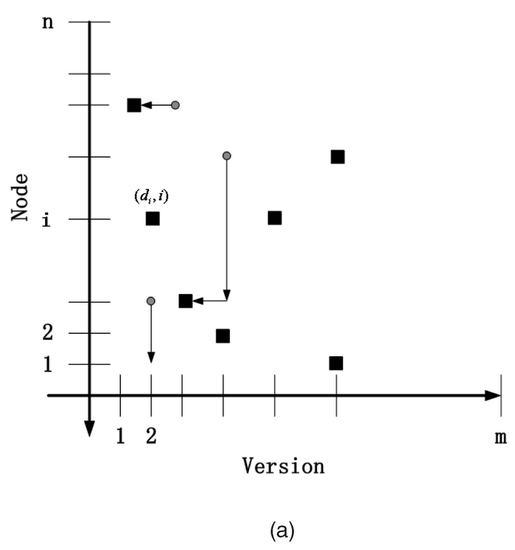
$$D_{1,m}^{(K)} = \min_{K \leq p \leq m} \{D_{1,p}^{(K-1)} + \overline{D}_p)\}.$$

Since $\overline{D}_p$ can all be computed in $O(m)$ time and we have showed that $D_{1,p}^{(1)}$ can be computed in $O(m \log m)$ time, we can easily prove that $D_{1,m}^{(K)}$ can be computed in $O(Km \log m)$ time by induction. Note that in the induction step, $D_{1,p}^{(K-1)}$ for $K-1 < p \leq m$ is computed in $O((K-1)m \log m)$ time. □

4.2 The Case of $n > 1$

When $n > 1$, the problem of multimedia object placement can be visualized as given in Fig. 5a. We can see that the requests can be served in one of the following ways: 1) A request is served by the exact versions at its local cache or one of the upstream caches. 2) A request is served by more detailed versions according to transcoding at its local cache or one of the upstream caches. 3) A request is served by the original server (no transcoding is executed since all versions are stored at the server).

In Fig. 5a, a square symbol at (d_i, i) indicates that version A_{d_i} is cached at node v_i and a dot indicates the request for a specified version from a node. Each node has exactly one such square symbol. A request for version A_j at node v_i might be either served at node v_i by version A_{d_i} if $j \geq d_i$ with transcoding cost $(j - d_i)T$, or at node v_{i-1} with additional transmission cost L. In the latter case, a new request for A_j is created at node v_{i-1}. This process can be generalized as $w(i,j) = \min\{(j-d_i)^+T, w(i-1,j) + L\}$ if $i \geq 1$; otherwise, $w(i,j) = 0$, where $w(i,j)$ is the access cost for request A_j at node v_i. In particular, if $L > mT$, then $w(i,j) = (j - d_i)T$ if $j \geq d_i$, i.e., transcoding is always performed if possible. Therefore, we can solve this problem using dynamic programming.



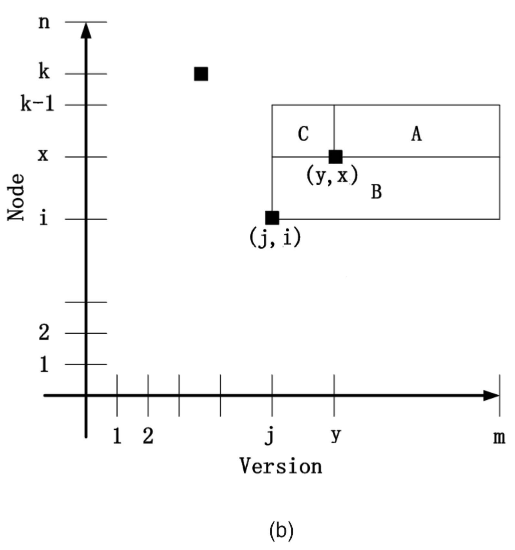


Fig. 5. Request flow and block definition for multimedia object placement.

Assume that version A_j is cached at node v_i, and v_k is the smallest vertex, $k > i$, with a cached version, say A_z, more detailed than A_j, i.e, $z \leq j$ (see Fig. 5b). Let $B_{i,j,k} = \{(\alpha, \beta) | i \leq \alpha \leq k-1, j \leq \beta \leq m\}$. We also assume that A_y is the most detailed version in Block $B_{i,j,k}$, which is cached at node v_x. Let $W(i,j,k)$ denote the minimum total access cost for serving all the requests in Block $B_{i,j,k}$. It is obvious that all the requests in Block C are served by version A_j at node v_i because the versions of all the requests in this block is more detailed than A_y, i.e., there does not exist a version in Block $B_{i,j,k}$ other than A_j that can provide the requested versions in this block since A_y is the most detailed version in Block $B_{i,j,k}$ besides A_j. Similarly, it is easy to see that the minimum total access cost for serving all the requests in Block A, i.e., $B_{x,y,k}$ and Block B, i.e., $B_{i,j,x+1}$ (see Fig. 5b), is $W(x,y,k) + W(i,j,y-1)$. With a similar method for partitioning Block $B_{i,j,k}$, Blocks A and B can be divided recursively until the minimum total access cost for serving all the requests in each block, i.e., $W(x,y,k)$ and $W(i,j,y-1)$, can be finally determined.

Based on the above observation, $W(i,j,k)$ is defined as follows:

$$W(i,j,k) = \begin{cases} \min_{i \leq x < k; j \leq y \leq m} \{W(i,j,y-1) + W(x,y,k) + \\ \sum_{x \leq \alpha < k; j \leq \beta \leq y} f_{\alpha,\beta}((\alpha - i)L + \\ (\beta - j)T) & (\text{for } 0 < i < k \leq n; \\ & 1 \leq j \leq m) \\ \min_{0 < x \leq k; 1 \leq y \leq m} \{W(0,1,y-1) \\ + W(x,y,k) + \sum_{x \leq \alpha < k; 1 \leq \beta \leq y} \beta L f_{\alpha,\beta} & (\text{for } i = 0, j = 1) \\ 0 & (\text{for } i = k). \end{cases} \tag{5}$$

Now, let us refer to the first equation in the recurrence formula above. The first term $W(i,j,y-1)$ is the total access cost for the requests in Block B and D, the second term is the total access cost for the requests in Block A, and the last term is the total access cost for the requests in Block C. The second equation is for the special case of $i = 0$ which denotes the original server, where transcoding is not necessary since all versions are stored there. To obtain the optimal solution, all possible values of i, j, and k must be checked. The following theorem shows the correctness of the above recurrence formula for $W(i,j,k)$:

Theorem 6. *Formula (5) is the correct recurrence formula for* $W(i,j,k)$.

Proof. Without loss of generality, we only need to prove the correctness of the first equation in (5) since the second equation can be easily derived in a similar way and the third equation is trivial.

Let

$$W'(i,j,k) = \min_{i \leq x < k; j \leq y \leq m} \{W(i,j,y-1) + W(x,y,k) + \sum_{x \leq \alpha < k; j \leq \beta \leq y} f_{\alpha,\beta}((\alpha - i)L + (\beta - j)T)$$

denote the value of the right side of the first equation.

We now prove that $W'(i,j,k)$ is the optimal access cost, i.e., $W'(i,j,k) = W(i,j,k)$. Suppose $A_{d_i^*}, A_{d_{i+1}^*}, \cdots, A_{d_k^*}$ is the optimal placement in Block $B_{i,j,k}$, i.e., $W(i,j,k) = C(A_{d_i^*}, A_{d_{i+1}^*}, \cdots, A_{d_k^*})$. Thus, we can always divide Block $B_{i,j,k}$ into four parts according to y^* (see Fig. 5b), where $y^* = \max_{j < y \leq m} \{d_y^*\}$ and version A_{y^*} is cached at node v_{x^*}. Therefore, we have

$$\begin{aligned} W(i,j,k) = C(A_{d_i^*}, A_{d_{i+1}^*}, \cdots, A_{d_{k-1}^*}) = W(i,j,y^*-1) \\ + W(x^*, y^*, k) + \sum_{x^* \leq \alpha < k; j \leq \beta \leq y^*} f_{\alpha,\beta}((\alpha - i)L \\ + (\beta - j)T) \geq \min_{i \leq x < k; j \leq y \leq m} \{W(i,j,y-1) \\ + W(x,y,k) + \sum_{x \leq \alpha < k; j \leq \beta \leq y} f_{\alpha,\beta}((\alpha - i)L \\ + (\beta - j)T). \end{aligned}$$

Now, we want to prove $W'(i,j,k) \geq W(i,j,k)$. Suppose there exists (x', y') such that

$$\begin{aligned} W'(i,j,k) &= \min_{i\le x<k; j\le y\le m}\{W(i,j,y-1)+W(x,y,k) \\ &\quad + \sum_{x\le\alpha<k; j\le\beta\le y} f_{\alpha,\beta}((\alpha-i)L+(\beta-j)T) \\ &= \min_{i\le x'<k; j\le y'\le m}\{W(i,j,y'-1)+W(x',y',k) \\ &\quad + \sum_{x'\le\alpha<k; j\le\beta\le y'} f_{\alpha,\beta}((\alpha-i)L+(\beta-j)T). \end{aligned}$$

Thus, Block $B_{\alpha,\beta}$ can be divided into four parts according to x' and y'. According to the definition of $W(i,j,k)$, we have

$$\begin{aligned} W(i,j,k) &\le \min_{i\le x'<k; j\le y'\le m}\{W(i,j,y'-1)+W(x',y',k) \\ &\quad + \sum_{x'\le\alpha<k; j\le\beta\le y'} f_{\alpha,\beta}((\alpha-i)L+(\beta-j)T) \\ &= \min_{i\le x<k; j\le y\le m}\{W(i,j,y-1)+W(x,y,k) \\ &\quad + \sum_{x\le\alpha<k; j\le\beta\le y} f_{\alpha,\beta}((\alpha-i)L+(\beta-j)T) \\ &= W'(i,j,k). \end{aligned}$$

Therefore, we have proved that $W'(i,j,k)=W(i,j,k)$. □

The original multimedia object placement problem, i.e., with the cost function based on (2), can be solved using dynamic programming with these recurrences. We can also see that the minimum access cost is $W(0,1,n)$. The detailed algorithm is given in Table 4.

Regarding to the time complexity of Algorithm 1, we have the following theorem:

Theorem 7. *Algorithm 1 can terminate in $O(n^3m^2)$ time, where n is the number of nodes and m is the number of versions.*

Proof. The work of Procedure $Block(i,j,k)$ is to compute $W(i,j,k)$. It is easy to see that $W(i,j,k)$ has n^2m different entries and each entry is computed only once (it simply returns the value if it was computed before). Consider the time complexity of computing an entry in $Block(i,j,k)$. It takes two layers of loops to compute an element in $Block(i,j,k)$. The outside *for-loop* on x iterates at most n times, and the inner *for-loop* on y iterates at most m times. Thus, it takes at most $O(n^2m)$ time of comparisons to compute an entry. Therefore, it takes $O(n^2m\cdot nm)=O(n^3m^2)$ time to compute all entries in $Block(i,j,k)$. □

TABLE 4
Algorithm for MOP Problem $(n>1)$

```
Main()
begin
  Call $Block(0,1,n)$
end

Procedure $Block(i,j,k)$
begin
  if $j=k$ then
    Exit;
  for x=i to k do
    for y=j to m do
      $Block(i,j,k)=Block(i,j,y-1)+Block(x,y,k)+BlockC(j)$
   Return $Block(i,j,k)$
end

Function $BlockC(j)$
begin
  if $j=0$ then
    $BlockC(j)=\sum_{x\le\alpha<k; j\le\beta\le y}\beta L f_{\alpha,\beta}$
  if $j>0$ then
    $BlockC(j)=\sum_{x\le\alpha<k; j\le\beta<y} f_{\alpha,\beta}[(\alpha-i)T+(\beta-j)L]$
   Return $BlockC(j)$
end
```

5 Simulation Model

In this section, the simulation model used for performance evaluation is described. We have performed extensive simulation experiments to compare our solution with existing solutions.

5.1 System Configuration

To the best of our knowledge, it is difficult to find true trace data in the open literature to execute such simulations. Therefore, we generated the simulation model from the empirical results presented in [1], [2], [3], [4], [6].

The network topology was randomly generated by the Tier program [4]. Experiments for many topologies with various parameters were conducted and the performance of our solution was found to be insensitive to topology changes. Here, only the experimental results for one topology are presented due to space limitations. The characteristics of this topology and the workload model are shown in Table 5, which were chosen from the open literature and are considered to be reasonable.

The WAN (Wide Area Network) is viewed as a backbone network to which no servers or clients are attached. Each MAN (Metropolitan Area Network) node is assumed to connect to a content server. Each MAN and WAN node is associated with an en-route cache. The objects generated are divided into two types: *text* and *multimedia*. Similar to the studies in [3], [26], cache size is described as the total relative size of all objects available in the content server. In our experiments, the object sizes are assumed to follow a Pareto distribution and the average object size is $6KB$. We also assume that each multimedia object has five versions and that the transcoding graph is as shown in Fig. 2 in Section 2.1. The sizes of each version are assumed to be 100 percent, 80 percent, 60 percent, 40 percent, and 20 percent of the original object size. The transcoding delay is determined as the quotient of the object size to the transcoding rate. In our experiments, the client at each

TABLE 5
Parameters Used in Simulation

Parameter	Value
Number of Nodes	400
Delay of WAN Links	Exponential Distribution ($\theta = 1.5Sec$)
Delay of MAN Links	Exponential Distribution ($\theta = 0.7Sec$)
Number of Web Objects	1000 objects per srever
Web Object Size Distribution	Pareto Distribution ($\mu = 6KB$)
Web Object Access Frequency	Zipf-Like Distribution ($\alpha = 0.7$)
Average Request Rate Per Node	$U(1,9)$ requests per second
Transcoding Cost	$50KB/Sec$

MAN node randomly generates the requests, and the average request rate of each node follows the distribution of $U(1,9)$, where $U(x,y)$ represents a uniform distribution between x and y. The access frequencies of both the content servers and the objects maintained by a given server follow a Zipf-like distribution [3], [22]. Specifically, the probability of a request for object O in server S is proportional to $1/(i^\alpha \cdot j^\alpha)$, where S is the ith most popular server and O is the jth popular object in S. The delay of both MAN links and WAN links follows an exponential distribution; the average delay for WAN links is 1.5 seconds and the average delay for WAN links is 0.7 seconds.

The cost for each link is calculated by the access delay. For simplicity, the delay caused by sending the request and the relevant response for that request is proportional to the size of the requested object. Here, we consider the average object sizes for calculating all delays, including the transmission delay and the transcoding delay. The cost function is taken to be the delay of the link, which means that the cost in our solution is interpreted as the access latency in our simulation.

We apply a "sliding window" technique, for estimating access frequency, to make our model less sensitive to transient workload [26]. Specifically, the access frequency is estimated by $N/(t - t_N)$, where N is the number of accesses recorded, t is the current time, and t_N is the Nth most recently referenced time (the time of the oldest reference in the sliding window). N is set to 2 in the simulation.

5.2 Existing Models

In addition to the solution proposed in Section 4.2, we also consider the following placement solutions for comparison purposes:

- SV: SV stores the same version of a multimedia object at each node when the request is sent back to the client from the server. For example, when the request is for A_2 of a multimedia object, the decision is to place A_2 at each node on the path from the client to the server.
- MV: MV stores the most referred version of a multimedia object at each node as the request is returned back to the client from the server. Specifically, if $i^* = \max_{1\leq j\leq m}\{f_{i,j}\}$, then version A_{i^*} is cached at node v_i.
- RV: RV randomly stores a version at each node.

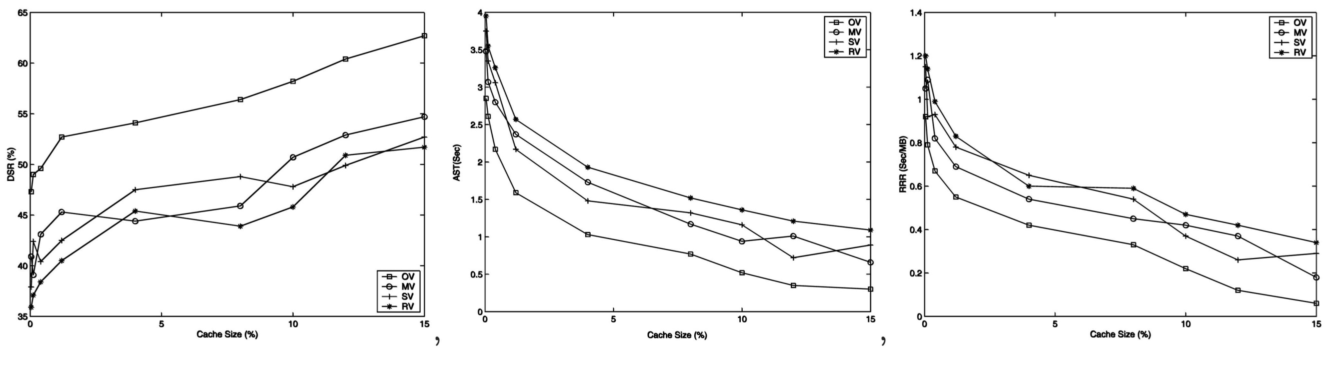
6 Performance Evaluation

In this section, we compare the performance results of our solution with those solutions introduced in Section 5.2 in terms of several performance metrics. The performance metrics we used in our simulation include delay-saving ratio (DSR), defined as the fraction of communication and server delays which is saved by satisfying the references from the cache instead of the server; average access latency (AST); request response ratio (RRR), defined as the ratio of the access latency of the target object to its size; object hit ratio (OHR), defined as the ratio of the number of requests satisfied by the caches as a whole to the total number of requests; and average server load (ASL), defined as the largest number of bytes served by the server per second. In the following figures, SV, MV, and RV denote the results for the three solutions introduced in Section 5.2, and OV denotes the optimal solution proposed in Section 4.2.

6.1 Impact of Cache Size

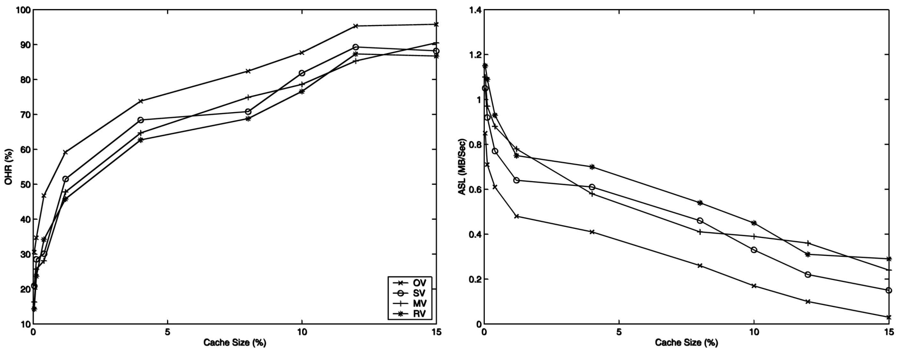
In this experiment set, we compare the performance results of different solutions across a wide range of cache sizes, from 0.04 percent to 15.0 percent.

The first experiment investigates DSR as a function of the relative cache size at each node and Fig. 6a shows the simulation results. As presented in Fig. 6a, we can see that our solution outperforms the others since it considers multimedia object placement by determining the optimal versions to be placed at each node, whereas existing solutions, including SV, MV, and RV, consider multimedia object placement heuristically or randomly. Specifically, the mean improvements of DSR over SV, MV, and RV are 4.3 percent, 17.9 percent, 19.8 percent, and 24.5 percent, respectively. Fig. 6b shows the simulation results of AST



(a) (b) (c)

Fig. 6. Experiments for DSR, AST, and RRR.



(a) (b)

Fig. 7. Experiments for OHR and ASL.

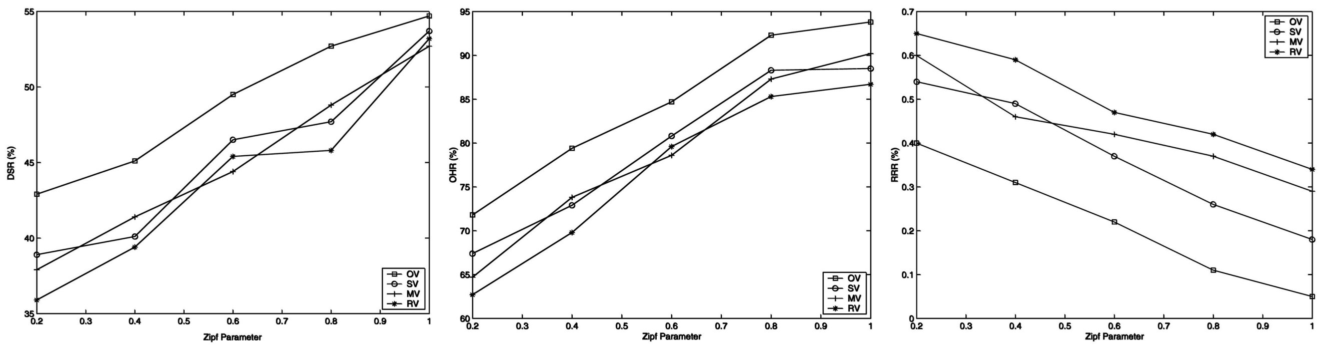


Fig. 8. Experiments for DSR, RRR, and OHR.

and RRR as a function of the relative cache size at each node; we describe the results of RRR as a function of the relative cache size at each node in Fig. 6c. Clearly, the lower the AST or the RRR, the better the performance. As we can see, all solutions provide steady performance improvement as the cache size increases. We can also see that OV significantly improves both AST and RRR compared to SV, MV, and RV, since our solution determines the optimal versions to be cached on the path from the client to the server, while the others place multiple versions of a multimedia object in a heuristic or random way. For AST to achieve the same performance as OV, the other solutions require two to eoght times as much cache size.

Fig. 7a shows the results of OHR as a function of the relative cache size for different solutions. By computing the optimal versions to be cached, we can see that our solution produces better results than the others, especially for smaller cache sizes. We can also see that OHR steadily improves as the relative cache size increases, which conforms to the fact that more requests will be satisfied by the caches as the cache size becomes larger. Fig. 7b shows the results of ASL as a function of the relative cache size. It can be seen that the ASL for our solution is lower than that for the other solutions. We can also see that ASL decreases as the relative cache size increases.

6.2 Impact of Object Access Frequency

This experiment set examines the impact of object access frequency distribution on the performance results of the various solutions. Fig. 8 shows the performance results of DSR, RRR, and OHR, respectively, for the values of Zipf parameter α from 0.2 to 1.0.

We can see that OV consistently provides the best performance over a wide range of object access frequency distributions. Specially, CV reduces or improves DSR by 30.4 percent, 24.4 percent, 21.3 percent, and 8.5 percent compared to SV, MV, and RV, respectively; the default cache size used here (4 percent) is fairly large in the context of Web caching due to the large network under consideration.

7 Conclusion

Transcoding is attracting increasing research interest in the environment of mobile appliances, and transparent data replication is receiving more and more attention since it is capable of high system scalability. In this paper, we addressed the problem of multimedia object placement for transparent data replication. We studied this problem for several cases with the objective of minimizing total access cost by combining both transmission cost and transcoding cost. A set of simulation experiments were conducted to study the performance of our proposed solutions. The simulation results showed that our solution can significantly improve network performance compared with existing solutions.

Acknowledgments

This work was supported by the Japan Society for the Promotion of Science (JSPS) under its General Research Scheme **B** (Grant No. 14380139), National Natural Science Foundation of China (Grand No. 60573087), and Hong Kong RGC grant (Grant No. HKU 7142/03E). Hong Shen is the corresponding author.

References

[1] C. Aggarwal, J.L. Wolf, and P.S. Yu, "Caching on the World Wide Web," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, no. 1, pp. 94-107, Jan. 1999.

[2] P. Barford and M. Crovella, "Generating Representive Web Workloads for Network and Server Performance Evaluation," *Proc. ACM SIGMETRICS '98*, pp. 151-160, 1998.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM '99*, pp. 126-134, 1999.

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

[5] S. Chandra, C. Ellis, and A. Vahdat, "Application-Level Differentiated Multimedia Web Services Using Quality Aware Transcoding," *IEEE J. Selected Areas in Comm.*, vol. 18, no. 12, pp. 2544-2565, 2000.

[6] C. Chang and M. Chen, "On Exploring Aggregate Effect for Efficient Cache Replacement in Transcoding Proxies," *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 6, pp. 611-624, June 2003.

[7] B.D. Davison, "Comparative Models of the File Assignment Problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287-313, 1982.

[8] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas, "Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing," *IEEE Personal Comm.*, vol. 5, no. 6, pp. 8-17, 1998.

[9] X. Jia, D. Li, H. Du, and J. Cao, "On Optimal Replication of Data Object at Hierarchical and Transparent Web Proxies," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 8, pp. 1-13, Aug. 2005.

[10] X. Jia, D. Li, X. Hu, and D. Du, "Optimal Placement of Web Proxies for Replicated Web Servers in the Internet," *The Computer J.*, vol. 44, no. 5, pp. 329-339, 2001.

[11] X. Jia, D. Li, X. Hu, W. Wu, and D. Du, "Placement of Web-Server Proxies with Consideration of Read and Update Cost on the Internet," *The Computer J.*, vol. 46, no. 4, pp. 378-390, 2003.

[12] A. Jiang and J. Bruck, "Optimal Content Placement for En-Route Web Caching," *Proc. Second Int'l Symp. Network Computing and Applications (NCA '03)*, pp. 9-16, 2003.

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

[14] B. Li, X. Deng, M.J. Golin, and K. Sohraby, "On the Optimal Placement of Web Proxies in the Internet: The Linear Topology," *Proc. Eighth IFIP Conf. High Performance Networking (HPN '98)*, pp. 21-25, 1998.

[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, 1999.

[16] K. Li and H. Shen, "Coordinated En-Route Multimedia Object Caching in Transcoding Proxies for Tree Networks," *ACM Trans. Multimedia Computing, Comm., and Applications (TOMCAPP)*, vol. 5, no. 3, pp. 289-314, 2005.

[17] K. Li, H. Shen, F. Chin, and S. Zheng, "Optimal Methods for Coordinated En-Route Web Caching for Tree Networks," *ACM Trans. Internet Technology (TOIT)*, vol. 5, no. 3, pp. 480-507, 2005.

[18] K. Li and H. Shen, "Optimal Methods for Proxy Placement in Coordinated En-Route Web Caching," *IEICE Trans. Comm.*, vol. E88-B, no. 4, pp. 1458-1466, 2005.

[19] K. Li and H. Shen, "Optimal Proxy Placement for Coordinated En-Route Transcoding Proxy Caching," *IEICE Trans. Information & Systems*, vol. E87-D, no. 12, pp. 2689-2696, 2004.

[20] K. Li and H. Shen, "Proxy Placement Problem for Coordinated En-Route Transcoding Proxy Caching," *Int'l J. Computer Systems, Science and Eng. (CSSE)*, vol. 19, no. 6, pp. 327-335, 2004.

[21] K. Li and H. Shen, "Optimal Methods for Object Placement in En-Route Web Caching for Tree Networks and Autonomous Systems," *Int'l J. High Performance Computing and Networking (IJHPCN)*, a conf. special issue of *GCC 2003*, vol. 3, no. 4, pp. 211-218, 2005.

[22] V.N. Padmanabhan and L. Qiu, "The Content and Access Dynamics of a Busy Site: Findings and Implications," *Proc. ACM SIGCOMM '00*, pp. 111-123, 2000.

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

[24] P. Rodriguez and S. Sibal, "Spread: Scalable Platform for Reliable and Efficient Distribution," *Computer Networks*, vol. 33, pp. 33-49, 2000.

[25] B. Shen, S.-J. Lee, and S. Basu, "Caching Strategies in Transcoding-Enabled Proxy Systems for Streaming Media Distribution Networks," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 375-386, 2004.

[26] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy Cache Algorithms: Design, Implementation, and Performance," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, no. 4, pp. 549-562, Apr. 1999.

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

[28] A. Vetro, C. Christopoulos, and H. Sun, "Video Transcoding Architectures and Techniques: An Overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18-29, 2003.

[29] J. Wang, "A Survey of Web Caching Schemes for the Internet," *ACM Computer Comm. Rev.*, vol. 29, no. 5, pp. 36-46, 1999.

[30] 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, 1991.

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

Keqiu Li received the Bachelor's and Master's degrees from the Department of Applied Mathematics at the Dalian University of Technology in 1994 and 1997, respectively. He received the PhD degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, in 2005. He is currently a professor in the College of Computer Science and Technology, Dalian Maritme University, China. His research interests include Internet technology, networking, multimedia applications, and bioinformatics.

Francis Y.L. Chin received the BASc degree from the University of Toronto, Canada, in 1972, and the MS, MA, and PhD degrees from Princeton University in 1974, 1975, and 1976, respectively. Since 1975, he has taught at the University of Maryland, Baltimore County, the University of California, San Diego, the University of Alberta, the Chinese University of Hong Kong, and the University of Texas at Dallas. He joined the University of Hong Kong (HKU) in 1985, where he is the chair of the Department of Computer Science and was the founding head of the department from its establishment until 31 December 1999. During 2001, Professor Chin was seconded to serve as the CEO (interim) of Hong Kong Domain Name Registration Company Limited. Since 2002, he has served as the associate dean of the graduate school. He has served on the program committees and as conference chairman of numerous international workshops and conferences. He is currently serving as managing editor of the *International Journal of Foundations of Computer Science* and is a member of the editorial boards of *Current Bioinforamtics*, *Information Processing Letters*, and the *Computer Processing of Oriental Languages*. Professor Chin current research interests are bioinformatics and algorithm studies.

Hong Shen received the BEng degree from the Beijing University of Science and Technology, the MEng degree from the University of Science and Technology of China, and the PhLic and PhD degrees from Abo Akademi University, Finland, all in computer science. He is a professor at the School of Computer Science at the University of Adelaide, Australia. Previously, he was a professor at the Japan Advanced Institute of Science and Technology and at Griffith University, Australia. Professor Shen has published more than 200 technical papers on algorithms, parallel and distributed computing, interconnection networks, parallel databases and data mining, multimedia systems, and networking. He has served in an editorial role for *Parallel and Distributed Computing Practice*, the *International Journal of Parallel and Distributed Systems and Networks*, *Parallel Algorithms and Applications*, the *International Journal of Computer Mathematics*, the *Journal of Supercomputing*, and the *Journal of Interconnection Networks*, on program committees, and as chair for various international conferences. Dr. Shen is a recipient of the 1991 National Education Commission Science and Technology Progress Award and the 1992 Sinica Academia Natural Sciences Award.

Weishi Zhang received the BS degree in computer science from Xi'an Jiaotong University, China, in 1984, and the MS degree in computer science from the Chinese Academy of Science, China, in 1986. He received the PhD degree in computer science from the University of Munich, Germany, in 1996. From 1986 to 1990, he was an assistant researcher at the Shenyang Institute of Computing, Chinese Academy of Science, China. From 1990 to 1992, he was a visiting scholar at Passau University, Germany. From 1992 to 1997, he was an assistant professor at the University of Munich, Germany. In 1997, he joined the Department of Computer Science, Dalian Maritime University, China, where he is currently a professor of computer science. His research interests include distributed computing, software engineering, software architecture, formal specification techniques, and program semantics models.