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
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 $A_2$, $A_1$, and $A_2$ at nodes $v_2$, $v_1$, 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 $v_1$, $v_1$, and $v_3$, respectively. The access costs for requesting versions $A_1$, $A_2$, and $A_3$ 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 $v_3$). 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 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.

### 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

---

**Table 1: Access Costs with Different Placement**

<table>
<thead>
<tr>
<th>CASE</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>$A_1$</td>
<td>$A_2$</td>
<td>$A_2$</td>
<td>$A_1$</td>
<td>$A_2$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$A_1$</td>
<td>$A_2$</td>
<td>$A_1$</td>
<td>$A_1$</td>
<td>$A_2$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$A_1$</td>
<td>$A_2$</td>
<td>$A_1$</td>
<td>$A_1$</td>
<td>$A_2$</td>
</tr>
<tr>
<td>Access Cost</td>
<td>1.5</td>
<td>6</td>
<td>2.5</td>
<td>5.5</td>
<td>12</td>
</tr>
</tbody>
</table>

---

Fig. 1. A simple example.

Fig. 2. A weighted transcoding graph.
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, $\Phi(A_1) = \{A_1, A_2, A_3, A_4, A_5\}$, $\Phi(A_2) = \{A_2, A_3, A_4\}$, 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 earlier 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].

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 $V = \{v_0, v_1, \ldots, v_n\}$ 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: $A_1, A_2, \ldots, A_m$. 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, $L_k(u, v) = 0$. 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 $f_{ij}$ 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 $v_1, v_2, \ldots, v_{n-1}$ 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 $A_{d_i}$ ($d_i \in \{1, 2, \ldots, m\}$) as the version cached at node $v_i$, then the total access cost of caching $A_{d_1}, A_{d_2}, \ldots, A_{d_n}$, denoted by $C(X)$, is defined as follows:

$$
C(X) = \sum_{i=1}^{n} \sum_{0 \leq k \leq m} f_{ij} \min\{L_k(v_i, v_k) + T(A_{d_i}, A_j)\},
$$

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

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

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

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 $\delta$.

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

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

Fig. 3. Transparent data replication model. (a) En-route model. (b) Hierarchical model. (c) Hybrid model.
special cases. If $L \gg T$, then version $A_d$ should be cached so that no transmission cost is necessary to incur, where $d_i = \min\{j|f_{i,j} > 0, 1 \leq j \leq m\}$. 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:

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

### 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 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 actions, 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.

---
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 $1 \leq k \leq m$. 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 $i \geq k$, 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:

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

Since version $A_k$ is cached at node $v_1$, we can see (with Assumption 3) that $\delta$ 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 $i - k \geq \delta$.

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

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

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

<table>
<thead>
<tr>
<th>Current Research</th>
<th>Object Type</th>
<th>Network Topology</th>
<th>Action</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WO</td>
<td>MO</td>
<td>LT</td>
<td>TT</td>
</tr>
<tr>
<td>This Paper</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>[16]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>[27]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>[9]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>[12,17,21]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>[31]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Thus, each $C(A_2), C(A_3), \ldots, C(A_m)$ 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 $\{A_1, A_2, \ldots, A_m\}$ 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. $\square$

The second step is to extend the above solution to compute the optimal solution for caching two versions, $A_{b_1}$ and $A_{b_2}$, at the same time at node $v_1$.

Suppose that $A_{b_1}$ and $A_{b_2}$ are the two optimal versions to be cached. The key observation here is that $A_{b_1}$ is also an optimal solution for the problem with $\{A_1, A_2, \ldots, A_{b_1} - 1\}$ if $k_1 < k_2$ because the requests for $\{A_{k_1}, A_{k_1 + 1}, \ldots, A_m\}$ can only be served by $A_{b_2}$. With regard to this observation, we have the following lemma:

**Lemma 1.** Assume that $A_{b_1}$ and $A_{b_2}$ are the optimal solutions for the problem of caching only one version from the set of $\{A_1, A_2, \ldots, A_{p-1}\}$ and $\{A_1, A_2, \ldots, A_{q-1}\}$, respectively, then we have $b_p \leq b_q$ if $p < q$.

**Proof.** Without loss of generality, it is sufficient for us to prove that $b_p \leq b_{p+1}$, where $1 \leq b_p \leq p - 1$ and $1 \leq b_{p+1} \leq p$. The proof is by contradiction. Assume that we have $b_p > b_{p+1}$. As $A_{b_1}$ is the optimal version to be cached, we have $C_{1,p}(A_{b_1}) < C_{1,p}(A_{b_{p+1}})$. Let $C_{1,p}(A_1)$ denote the access cost of caching $A_1$ for the MOP problem with $\{A_1, A_2, \ldots, A_{p-1}\}$. From the definition of the access cost function $C_{1,p}$ as given in (3), adding $A_2$ to the set $\{A_1, A_2, \ldots, A_{p-1}\}$ will increase both $C_{1,p}(A_{b_1})$ 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 $C_{1,p}(A_{b_1})$ is no less than that to $C_{1,p}(A_{b_{p+1}})$ because $b_p > b_{p+1}$. So, we have $C_{1,p+1}(A_{b_1}) < C_{1,p+1}(A_{b_{p+1}})$, which contradicts the fact that $C_{1,p+1}(A_{b_1})$ is the minimum access cost of caching $A_{b_1}$ for the problem with $\{A_1, A_2, \ldots, A_{p-1}, A_p\}$. $\square$

Based on Lemma 1, we can see that the feasible range of the optimal solution for the problem with $\{A_1, A_2, \ldots, A_m\}$ can be reduced if the optimal version for the problem with $\{A_1, A_2, \ldots, A_p\}$ has been obtained. So is the other case when the optimal solution for the problem with $\{A_1, A_2, \ldots, A_q\}$ is known; the feasible range of the optimal solution for the problem with $\{A_1, A_2, \ldots, A_m\}$ is also reduced. Therefore, we can find $A_{b_1}$ and compute $C_{1,p}(A_{b_1})$ by divide and conquer.

Let $D^{(k)}_{p,q}$ denote the minimum access cost of caching $k$ versions for the MOP problem with $p - q$ versions, i.e., $A_p, A_{p+1}, \ldots, A_{q-1}$, where $1 \leq p < q \leq m$. Thus, $D^{(1)}_{1,p} = C_{1,p}(A_{b_1})$ and $D^{(m+1)}_{1,m} = \min\{C_{1,m+1}(A_q)\}$. Based on Lemma 1, we have the following theorem on the time complexity of computing $D^{(k)}_{1,p}$ for $1 < k \leq m$:

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

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

Based on Theorem 1, we can compute $D^{(p)}_{1,p}$ in $O(m)$ time. Assume that $A_{b_1}$ is the optimal solution for the problem of caching only one version with $\{A_1, A_2, \ldots, A_{\frac{m}{2} - 1}\}$; then, we can find the optimal solution for the problem of caching only one version for $\{A_1, A_2, \ldots, A_{\frac{m}{2}}\}$ in $O(m)$ time. Similarly, $D^{(\frac{m}{2})}_{1,\frac{m}{2}}$ can also be computed by solving the problem of caching only one version with $\{A_1, A_2, \ldots, A_{\frac{m}{2}}\}$.

As we have already computed $C_{1,p}(A_q)$, we can base on this result to compute $C_{1,p}(A_q)$ for $\{A_1, A_2, \ldots, A_{\frac{m}{2}}\}$ (by adding at most $\frac{m}{2}$ terms to $C_{1,p}(A_{\frac{m}{2}})$). We then compute $C_{1,p}(A_q), C_{1,p}(A_{\frac{m}{2}}), \ldots, C_{1,p}(A_{\frac{m}{2}})$ in at most $O(m \log m)$ time. So, it takes at most $O(m)$ time to compute $D^{(\frac{m}{2})}_{1,\frac{m}{2}}$ and $D^{(\frac{m}{4})}_{1,\frac{m}{4}}$. According to the similar decomposition, $D^{(\frac{m}{4})}_{1,\frac{m}{4}}, D^{(\frac{m}{8})}_{1,\frac{m}{8}}, D^{(\frac{m}{16})}_{1,\frac{m}{16}}$, and $D^{(\frac{m}{32})}_{1,\frac{m}{32}}$ can all be solved in $O(m)$ time. To be precise, let $A_1, A_2, A_3$ be the optimal versions for $\{A_1, A_2, \ldots, A_{\frac{m}{2}}\}$, $\{A_1, A_2, \ldots, A_{\frac{m}{4}}\}$, and $\{A_1, A_2, \ldots, A_{\frac{m}{8}}\}$, respectively. The first step is to compute $C_{1,p}(A_1)$, and then $C_{1,p}(A_3)$, $C_{1,p}(A_5)$, and $C_{1,p}(A_7)$ from $C_{1,p}(A_1), C_{1,p}(A_3)$, and $C_{1,p}(A_5)$, respectively. As the computation of each item takes $O(\log \frac{m}{p})$ time, this step takes $O(m \log m)$ time in total. Then, we can search the optimal solutions for $\{A_1, A_2, \ldots, A_{\frac{m}{2}}\}$, $\{A_1, A_2, A_3, A_{\frac{m}{2}}\}$, and $\{A_1, A_2, A_3, A_5, A_{\frac{m}{4}}\}$ in the ranges $(1, \min\{\frac{m}{2}, \frac{m}{4} - 1\})$, $(\frac{m}{2}, \min\{\frac{m}{4}, \frac{m}{8} - 1\})$, and $(\frac{m}{4}, \min\{\frac{m}{8}, \frac{m}{16} - 1\})$, 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 \(D^{(1)}_{1,p}\) for \(1 < p \leq m\).

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

- **Step 1:** Evaluate \(D^{(1)}_{1,p}\) for \(1 < p \leq m\), where \(D^{(1)}_{1,p}\) 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, \(D^{(1)}_{1,m+1} = \min_{1 \leq i \leq m} \{C_1(A_i)\}\).

- **Step 2:** Evaluate \(D_p\) for \(2 \leq p \leq m\), where \(D_p\) is the access cost for versions \(A_p, A_{p+1}, \cdots, A_m\) if \(A_p\) is cached at node \(v_1\). \(D_p\) is defined as follows:

\[
D_p = \begin{cases} 
  \sum_{i=p}^{p+\delta-1} f_i(i-p)T + \sum_{i=p+\delta}^{m} f_iL & \text{if } p + \delta \leq m \\
  \sum_{i=p}^{m} f_i(i-p)T & \text{if } p + \delta > m.
\end{cases}
\]

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

\[
D^{(2)}_{1,m} = \min_{2 \leq p \leq m} \{D^{(1)}_{1,p} + D_p\}.
\]

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

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

**Proof.** Assume that

\[
D^{(2)}_{1,m} = D^{(1)}_{1,p'} + D_p = \min_{2 \leq p \leq m} \{D^{(1)}_{1,p} + D_p\}.
\]

It is obvious from the computation of \(D^{(1)}_{1,m}\) that \(b_p\) and \(A_p\) are the two versions which achieve the minimum access cost of caching two versions, where \(D^{(1)}_{1,p'} = C_{1,p'}(b_p)\).

The following theorem shows the time complexity of computing \(D^{(2)}_{1,m}\):

**Theorem 4.** \(D^{(2)}_{1,m}\) 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 \(D^{(2)}_{1,m}\) is \(O(m \log m)\).

After we have calculated \(D^{(1)}_{1,p}\) for \(1 \leq p \leq m\) in Step 1, we can obtain \(D^{(2)}_{1,p}\) for all \(2 \leq p \leq m\) in another \(O(m \log m)\) time by divide and conquer, where \(D^{(2)}_{1,p}\) 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 \(D^{(1)}_{1,p}\). Assume that \(A_{b_1}\) and \(A_{b_2}\) with \(1 \leq b_1 < b_2 < p\) 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 \(D^{(2)}_{1,p}\). Similarly, \(A_{b_1}\) and \(A_{b_2}\) with \(1 \leq b_1 < b_2 < q\) 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 \(D^{(2)}_{1,q}\).

We can show with a similar argument with Lemma 1 that \(b_m \leq b_{q'}\) if \(p < q\) and this property limits the range of searching for the optimal solutions. As in Theorem 2, the two optimal solutions in \(D^{(2)}_{1,p}\) can be found in \(O(m)\) time after knowing the optimal versions of \(D^{(1)}_{1,p}\) for \(1 < p \leq m\); then, \(D^{(2)}_{1,p}\) and \(D^{(2)}_{1,m}\) in another \(O(m)\) time; then, \(D^{(2)}_{1,2p}\), \(D^{(2)}_{1,4p}\), and \(D^{(2)}_{1,8p}\) in another \(O(m)\) time until \(D^{(2)}_{1,p}\) for \(2 < p \leq m\) are found after \(\log m\) times. Therefore, the minimum access cost of caching three versions, denoted by \(D^{(3)}_{1,m}\), can be computed similarly, i.e., \(D^{(3)}_{1,m} = \min_{3 \leq p \leq m} \{D^{(2)}_{1,p} + D_p\}\), 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 \(D^{(K)}_{1,m}\) 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 \(D^{(K)}_{1,m}\):

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

**Proof.** Based on the above analysis, we have

\[
D^{(K)}_{1,m} = \min_{K \leq p \leq m} \{D^{(K-1)}_{1,p} + D_p\}.
\]

Since \(D_p\) can all be computed in \(O(m)\) time and we have showed that \(D^{(1)}_{1,p}\) can be computed in \(O(m \log m)\) time, we can easily prove that \(D^{(K)}_{1,m}\) can be computed in \(O(Km \log m)\) time by induction. Note that in the induction step, \(D^{(K-1)}_{1,p}\) 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 \((d, i)\) indicates that version \(A_d\) 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 \(A_d\) if \(j \geq d\), with transcoding cost \(T(d - j)\), 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

\[
w(i, j) = \min_{j} \{w(i, j) + T, w(i, j) + L\} \quad \text{if} \quad i \leq 1; \quad \text{otherwise,} \quad w(i, j) = 0, \quad \text{where} \quad w(i, j)\text{ is the access cost for request } A_j\text{ at node } v_i\text{.}\]

In particular, if \(L > mT\), then

\[
w(i, j) = (d - j)T \quad \text{if} \quad j \geq d, \quad \text{i.e., transcoding is always performed if possible. Therefore, we can solve this problem using dynamic programming.}
\]
Assume that version $A_i$ is cached at node $v_i$, and $v_k$ is the smallest vertex, $k > i$, with a cached version, say $A_k$, more detailed than $A_i$, i.e., $z < j$ (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 $A_i$ is the most detailed version in Block $B_{i,j,k}$, which is cached at node $v_i$. 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 $A_i$ 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 $A_i$ that can provide the requested versions in this block since $A_y$ is the most detailed version in Block $B_{i,j,k}$ besides $A_i$. Similarly, it is easy to see that the minimum total access cost for serving all the requests in Block $A$, i.e., $B_{i,y,k}$ and Block $B$, i.e., $B_{i,j-1}$ (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:

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

(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 Blocks $B$ 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 $W'(i, j, k) = \min_{i \leq x < k, j \leq y \leq m} \{W(i, j, y - 1) + W(x, y, k) + \sum_{z < k, j \leq y} f_{\alpha, \beta}(\alpha - i)L + (\beta - j)T \}$ 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 $A_{x_k}, A_{x_{k+1}}, \cdots, A_{x_n}$ is the optimal placement in Block $B_{i,j,k}$, i.e., $W(i, j, k) = C(A_{x_k}, A_{x_{k+1}}, \cdots, A_{x_n})$. Thus, we can always divide Block $B_{i,j,k}$ into four parts according to $y'$ (see Fig. 5b), where $y' = \max \{ d^y \}$ and version $A_{y'}$ is cached at node $v_{y'}$. Therefore, we have

$$W(i, j, k) = C(A_{x_k}, A_{x_{k+1}}, \cdots, A_{x_n}) = W(i, j, y' - 1) + W(x', y', k) + \sum_{x' < k, j \leq y'} f_{\alpha, \beta}(\alpha - i)L + (\beta - j)T \geq \min_{i \leq x < k, j \leq y' \leq m} \{W(i, j, y' - 1) + W(x, y, k) + \sum_{z < k, j \leq y'} f_{\alpha, \beta}(\alpha - i)L + (\beta - j)T \}.
$$

Now, we want to prove $W'(i, j, k) \geq W(i, j, k)$. Suppose there exists $(x', y')$ such that
Thus, Block $B_{a,b}$ can be divided into four parts according to $x'$ and $y'$. According to the definition of $W(i,j,k)$, we have

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

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 $O(n^3m^2)$ time to compute all entries in $Block(i,j,k)$.  

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
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 $i^* = \max_{1 \leq j \leq m} \{f_{ij}\}$, then version $A_{i^*}$ is cached at node $v_i$.

- **RV**: RV randomly stores a version at each node.

### 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 $i^* = \max_{1 \leq j \leq m} \{f_{ij}\}$, 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 $AST$. 

### Table 5

Parameters Used in Simulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Nodes</td>
<td>400</td>
</tr>
<tr>
<td>Delay of WAN Links</td>
<td>Exponential Distribution ($\theta = 1.5\text{Sec}$)</td>
</tr>
<tr>
<td>Delay of MAN Links</td>
<td>Exponential Distribution ($\theta = 0.7\text{Sec}$)</td>
</tr>
<tr>
<td>Number of Web Objects</td>
<td>1000 objects per server</td>
</tr>
<tr>
<td>Web Object Size Distribution</td>
<td>Pareto Distribution ($\mu = 6\text{KB}$)</td>
</tr>
<tr>
<td>Web Object Access Frequency</td>
<td>Zipf-Like Distribution ($\alpha = 0.7$)</td>
</tr>
<tr>
<td>Average Request Rate Per Node</td>
<td>$U(1, 9)$ requests per second</td>
</tr>
<tr>
<td>Transcoding Cost</td>
<td>$50\text{KB/Sec}$</td>
</tr>
</tbody>
</table>
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 eight 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

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 Maritime 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 Bioinformatics, 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.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.