J Supercomput (2010) 52: 284–302
DOI 10.1007/s11227-009-0291-8

Coordinated multimedia object replacement in transcoding proxies

Keqiu Li · Yanming Shen · Kai Lin · Wenyu Qu

Published online: 19 April 2009

Abstract Finding replacement candidates for accommodating a new object is an important research issue in web caching. Due to the new emerging factors in the transcoding proxy and the aggregate effect of caching multiple versions of the same multimedia object, this problem becomes more important and complex as audio and video applications have proliferated over the Internet, especially in the environment of mobile computing systems. This paper addresses coordinated cache replacement in transcoding proxies. First, we propose an original model which determines cache replacement candidates on all candidate nodes in a coordinated fashion with the objective of minimizing the total cost loss for linear topology. We formulate this problem as an optimization problem and present a low-cost optimal solution for deciding cache replacement candidates. Second, we extend this problem to solve the same problem for tree networks. Finally, we conduct extensive simulations to evaluate the performance of our solutions by comparing with existing models.

Keywords Web caching · Multimedia object · Transcoding proxy · Cache replacement · Transcoding · Internet

1 Introduction

Web caching is an important technology for improving the services over Internet. Since the majority of web objects are static, caching them at various network com-

K. Li (✉) · Y. Shen · K. Lin
Department of Computer Science and Engineering, Dalian University of Technology, No 2, Linggong Road, Dalian 116024, China
e-mail: keqiu@dlut.edu.cn

W. Qu
School of Information Science and Technology, Dalian Maritime University, No 1, Linghai Road, Dalian 116026, China

ponents (e.g., client browser, proxy server) provides a natural way of decreasing network traffic. Moreover, web caching can also reduce users' access latency and alleviate server load.

A key factor that affects the performance of web caching is the cache replacement policy, which is a decision for evicting an object currently in the cache to make room for a new object. A number of cache replacement policies, which attempt to optimize various performance metrics, such as hit ratio, byte hit ratio, delay saving ratio, etc., have been proposed in the literature [4, 24]. However, all these polices are local replacement models that determine cache replacement candidates from the view of only a single node. Furthermore, they become inefficient in transcoding proxies due to the new emerging factors in the transcoding proxy (e.g., the additional delay caused by transcoding, different sizes, and reference rates for different versions of a multimedia object) and the aggregate effect of caching multiple versions of the same multimedia object. Although the authors have elaborated these issues in [13], they considered the cache replacement problem at only a single node. Cooperative caching, in which caches cooperate in serving each other's requests and making storage decisions, is a powerful paradigm to improve cache effectiveness [10, 11, 15, 21]. There are two orthogonal issues to cooperative caching: object location (i.e., finding nearby copies of objects) and object management (i.e., coordinating the caches while making storage decisions). The object location problem has been widely studied [14, 16, 30]. Efficient coordinated object management algorithms are crucial to the performance of a cooperative caching system, which can be divided into two type of algorithms: placement and replacement algorithms. There are a number of research on finding efficient solutions for cooperative object placement [19, 29, 32]. However, there is little work done on finding efficient solutions for cooperative object replacement. Due to the interrelationship among different versions of the same multimedia object, cooperative caching in transcoding proxies becomes more important and complicated. This is very significant for the performance of a cooperative caching system since when a updated version is to be cached, an efficient replacement policy should decide cache replacement candidates by considering the cooperation of all the nodes on the path from the server to the client. Another important point is that the replacement decision on each node should be beneficial, i.e., the profit gained by caching the new object should be no less than the profit lost by removing some objects from the cache to make room for the new object. As the transcoding proxy is attracting an increasing amount of attention in the environment of mobile computing, it is noted that new efficient cache replacement policies are required for these transcoding proxies. In this paper, we address coordinated cache replacement in transcoding proxies. The main contributions of this paper are summarized as follows:

- We propose an original model which determines cache replacement candidates among all candidate nodes in a coordinated fashion with the objective of minimizing the total cost loss for linear topology.
- We formulate this problem of an optimization problem and present a low-cost optimal solution for deciding cache replacement candidates.
- We further present an optimal solution for the same problem for tree networks.
- We evaluate our solution on various performance metrics through extensive simulation experiments with existing models.

The rest of this paper is organized as follows: Sections 2 and 3 introduce related work and preliminaries, respectively. We formulate the problem and present an optimal solution for this problem in Sect. 4. In Sects. 5 and 6, simulation model and performance evaluation are described, respectively. Finally, we conclude this paper in Sect. 7.

2 Related work

Cache replacement policies play an important role in the functionality of web caching. An overview of web caching replacement algorithms can be found in [4, 24]. All these policies can be generally classified into three categories as traditional replacement policies and their extensions [23], key-based replacement policies [31], function-based replacement policies [1, 12, 17, 25, 27], and transcoding technology-based replacement policies [13, 26, 28]. A common characteristic among them is that the replacement decisions are drawn from the view of only a single node. These single node-based algorithms become inefficient when cooperative caching is considered. This problem becomes more difficult and complicated in transcoding proxies due to the new emerging factors in the transcoding proxy and the aggregate effect of caching multiple versions of the same multimedia object. In [13], the authors proposed an efficient cache replacement algorithm for transcoding proxies by exploring the aggregate effect of caching multiple versions of the same multimedia object in the same cache at only a single node. Finding efficient coordinated cache replacement policies in transcoding proxies will greatly improve the performance of web caching as audio and video applications have proliferated on the Internet.

Cooperative caching has been widely discussed in the literature. These studies include analytical results [3, 18], simulation experiments [6, 16], and prototypes and products [2, 8]. Cooperative caching helps for two reasons [21]. First, cooperation permits a busy cache to utilize a nearby idle cache for serving a specified request. Moreover, cooperation balances the improved hit time achieved by increasing the replication of popular objects against the improved hit ratio achieved by reducing replication and storing more objects.

There is little work done on cooperatively determining replacement candidates when a new object is to be cached at a number of caches. In [21], the authors proposed a cooperative replacement algorithm that not only preserves the implicit coordination offered by greedy-dual, but also enables busy caches to utilize nearby idle caches. The problem addressed in this paper is from a different point of view, which considers the coordination among all the caches and the interrelationship among different versions of the same multimedia object.

3 Preliminaries

We first introduce multimedia object transcoding in Sect. 3.1, and then notations and definitions in Sect. 1.

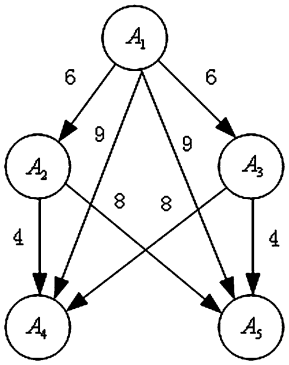


Fig. 1 An example of a weighted transcoding graph

3.1 Multimedia object transcoding

Transcoding is used to transform a multimedia object from one form to another, frequently trading off object fidelity for size, i.e., the process of converting a media file or object from one format to another. Transcoding is often used to convert video formats (i.e., Beta to VHS, QuickTime to MPEG). But it is also used to fit HTML files and graphics files to the unique constraints of mobile devices and other Web-enabled products. These devices usually have smaller screen sizes, lower memory, and slower bandwidth rates. In this scenario, transcoding is performed by a transcoding proxy server or device, which receives the requested document or file and uses a specified annotation to adapt it to the client.

The relationship among different versions of a multimedia object can be expressed by a weighted transcoding graph. An example of such a graph is shown in Fig. 1, where 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 our example, transcoding can not 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 $w(i, j)$. The number beside each edge in Fig. 1 is the transcoding cost from one version to another. For example, $w(1, 2) = 6$, and $w(3, 4) = 4$. $\phi(i)$ is the set of all the versions that can be transcoded from A_i, including A_i. For example, $\phi(1) = \{1, 2, 3, 4, 5\}$, $\phi(2) = \{2, 4, 5\}$, and $\phi(4) = \{4\}$. In this paper, we use W to denote a weighted transcoding graph.

3.2 Notations and definitions

We model the network as a graph $G = (V, E)$ in this paper, 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. We assume that every node is associated with a cache with the same size B and there are m multimedia objects maintained by the server. For multimedia object j, we assume that it has m_j versions $(O_{j,1}, O_{j,2}, \ldots, O_{j,m_j})$. To simplify the analysis in Sect. 4.2, we assume that all versions of the same multimedia object have the same size. Our analysis can be extended to the general case with the same methodology. Thus, each node can hold at most B objects. We denote the set of objects cached at node v_i by

Table 1 Parameters used in simulation

$V = \{v_0, v_1, \ldots, v_n\}$	Set of nodes in the network
O_k	Version k of object j
A^i	Set of different versions of object j cached at node v_i
$L_k(u, v)$	Cost of sending a request for version O_k over the link (u, v)
$r_{i,k}$	Request for O_k at node v_i
$f_{i,k}$	Frequency of $r_{i,k}$
$S(r_{i,k})$	Serving object for $r_{i,k}$

$Y^i = \{A^i_1, A^i_2, \ldots, A^i_m\}$, where $A^i_j \subseteq \{O_{j,k_1}, O_{j,k2}, \ldots, O_{j,k_j}\}$ is the set of different versions of object j cached at node v_i. Obviously, $Y = \{Y^1, Y^2, \ldots, Y^n\}$ is the set of all objects cached. For each version of object O_j, we associate each link $(u, v) \in E$ a nonnegative cost $L_{j,k}(u, v)$, which is defined as the cost of sending a request for version $O_{j,k}$ and the relevant response over the link (u, v). In particular, $L_{j,k}(u, u) = 0$. 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 $r_{i,j,k}$ denote the request for $O_{j,k}$ at node v_i and $f_{i,j,k}$ be the frequency of $r_{i,j,k}$.

For notational tidiness, we omit argument j in all parameters and functions throughout the following analysis since our analysis is based on a specific object. For example, O_k denotes version k of object j, A^i is the set of different versions of object j cached at node v_i, $L_k(u, v)$ denotes the cost of sending a request for version O_k over the link (u, v), $r_{i,k}$ denotes the request for O_k at node v_i, and $f_{i,k}$ denotes the frequency of $r_{i,k}$. In our analysis, we assume that the server holds all the m versions of object j, which is denoted by $Z_{0,1}, Z_{0,2}, \ldots, Z_{0,m}$. In our analysis, we assume that $L_k(v_{i_1}, v_{i_2}) = (i_1 - i_2)L$ for all $1 \le k \le m$ as there are $i_1 - i_2$ links on the path between node v_{i_1} and node v_{i_2}, and the cost on each link for each version of O_j is L. The transcoding graph is a linear array and the transcoding cost between any two adjacent versions is constant, i.e., $t(O_{k_1}, O_{k_2}) = \sum_{k=k_1}^{k_2-1} t(O_k, O_{k+1}) = (k_2 - k_1)^+ T$, where $x^+ = x$ if $x \ge 0$ else $x^+ = \infty$. we also assume that there exists some positive integer δ such that $(\delta - 1)T \le L$, and $\delta T > L$. If not, i.e., $L \gg T$ or $T \gg L$, then they are obviously two trivial cases.

For easy understanding, we summarize the notation used in this paper in Table 1.

4 Cooperative cache replacement in transcoding proxies

In this section, we first formulate the problem in Sect. 4.1, and then present an optimal solution for this problem in Sect. 4.2. Finally, we describe our cooperative cache replacement scheme in Sect. 4.3.

4.1 Problem formulation

Before formulating the problem, we give some explanation on how the requests are served. As shown in Fig. 2, a request goes along a routing path from the client

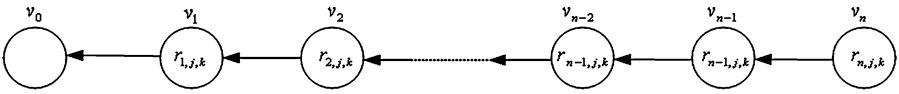


Fig. 2 System model for multimedia object caching

(node v_n) to the server (node v_0). Note that any request $r_{i,k}$ could find the service from $S(r_{i,k})$, where $S(r_{i,k})$ denotes the serving object for $r_{i,k}$. Assume that $S(r_{i,k}) = O_{k_1} \in A^{i_1}$ with $k_1 \leq k$ and $i_1 \leq i$, then there may be the following ways of serving $r_{i,k}$ by $O_{k_1} \in A^{i_1}$:

- O_{k_1} is first sent from node v_{i_1} to node v_i and then transcoded to O_k at node v_i.
- O_{k_1} is first transcoded to O_k at node v_{i_1} and then O_k is sent from node v_{i_1} to node v_i.
- O_{k_1} is first sent from node v_{i_1} to node v_{i_2}, transcoded to O_k at node v_{i_2}, and then O_k is sent from node v_{i_2} to node v_i.
- O_{k_1} is first sent from node v_{i_1} to node v_{i_2} and transcoded to O_{k_2} at node v_{i_2}, and then O_{k_2} is sent from node v_{i_2} to node v_{i_3} and transcoded to O_{k_3} at node v_{i_3}, then O_{k_3} is sent from node v_{i_3} to node v_i and transcoded to O_k at node v_i.
- $\vdots$

From the above analysis, we can see that when a new or updated version of a multimedia object to be cached (O_{i_0} for instance) is passing through each node between nodes $v_{i'}$ and v_i, it should be decided where O_{i_0} should be cached and which version should removed from the relevant cache to make room for it depending on how $r_{i,k}$ is served. Given X (i.e., the set of cached objects) and $O_{k'} \in A^{i'}$ ($i' \leq i$), then $d(r_{i,k}, O_{k'})$ is defined as follows:

$$d(r_{i,k}, O_{k'}) = (i - i')L + (k - k')^+ T \tag{1}$$

where $d(r_{i,k}, O_{k'})$ is the access cost of serving $r_{i,k}$ by $O_{k'}$ at node $v_{i'}$.

Now, we begin to formulate the problem addressed in this paper, i.e., determining where a new or updated version O_{i_0} should be cached among nodes $\{v_1, v_2, \ldots, v_n\}$ and which version of object j should be removed at that node to make room for O_{i_0} such that the total cost loss is minimized. Suppose that $P \subseteq V$ is the set of nodes at each of which $X_{i,k_i} \in A^i$ should be removed to make room for O_{i_0}, then this problem can be formally defined as follows:

$$L(P^*) = \min_{P \subseteq V} \{L(P)\} = \sum_{v_i \in P} \left(l(X_{i,k_i}) - g_i(O_{i_0})\right) \tag{2}$$

where $L(P)$ is the total relative cost loss, $l(X_{i,k_i})$ is the cost loss of removing X_{i,k_i} from node v_i, and $g_i(O_{i_0})$ is the cost saving of caching O_{i_0} at node v_i.

4.2 Dynamic programming-based solution

Before presenting the solution, we evaluate the two items, i.e., $l(X_{i,k_i})$ and $g_i(O_{i_0})$, shown in (2) in detail.

First, we begin with presenting a solution for finding the best way of serving $r_{i,k}$, i.e., finding $S(r_{i,k})$. Based on (1), the cost of serving $r_{i,k}$, denoted by $c(r_{i,k})$, is defined as follows:

$$c(r_{i,k}) = \min\left\{ \min_{O_{k'} \in A^{i'}, 1 \le i' \le i} d(r_{i,k}, O_{k'}), iL \right\} \tag{3}$$

Therefore, the object for serving $r_{i,k}$, denoted by $S(r_{i,k})$, is determined as follows:

$$S(r_{i,k}) = \begin{cases} O_{k^*} & \text{if } c(r_{i,k}) = d(r_{i,k}, O_{k^*}) \\ Z_{0,k} & \text{if } c(r_{i,k}) = iL \end{cases} \tag{4}$$

The following property will help us simplify the problem of finding the best way of serving $r_{i,k}$.

Theorem 1 *If both O_{k_1} and O_{k_2} are cached at node $v_{i'}$, then we have $d(r_{i,k}, O_{k_1}) < d(r_{i,k}, O_{k_2})$ for $k > k_1 > k_2$.*

Proof Based on the definition of $d(r_{i,k}, O_k)$, we have $d(r_{i,k}, O_{k_1}) = (i - i')L + (k - k_1)^+ T$ and $d(r_{i,k}, O_{k_2}) = (i - i')L + (k - k_2)^+ T$. Since $(k - k_1)^+ < (k - k_2)^+$, we have $d(r_{i,k}, O_{k_1}) < d(r_{i,k}, O_{k_2})$. Hence, the theorem is proven. □

From Theorem 1, we can see that for request $r_{i,k}$, we can consider only the least detailed version that can be transcoded to version k. Thus (3) can be simplified as follows:

$$c(r_{i,k}) = \min\left\{ \min_{1 \le i' \le i} d(r_{i,k}, O_{k''}), iL \right\} \tag{5}$$

where $O_{k''}$ is the least detailed version of object j cached at node $v_{i'}$ that can be transcoded to version k.

It is easy to see that the time complexity for computing $S(r_{i,k})$ is $O(\log n)$, where n is the number of nodes in the network. So, the total complexity for computing all $S(r_{i,k})$ ($1 \le i \le n$ and $1 \le k \le m$) is $O(mn \log n)$ since there are n nodes and object j has m different versions.

For each object $x \in X$, the set of requests served by x is expressed as $R(x) = \{r_{i,k} | S(r_{i,k}) = x\}$ and the total cost for the requests served by x is $C(x) = \sum_{r_{i,k} \in R(x)} f_{i,k} d(r_{i,k}, x)$.

Regarding to $R(x)$, we have the following property.

Property 1 *If $r_{i,k} \in R(x)$, then $r_{i',k'} \in R(x')$ $\forall i' \le i$ and $k' \le k$.*

Proof Suppose that $x \in A^{i_1} = O_{k_1}$, $x' \in A^{i_2} = O_{k_2}$ and there exists $i' \le i$ and $k' \le k$ such that $r_{i',k'} \in R(O_{i_2})$. Since $S(r_{i',k'}) = x'$, we have $d(r_{i',k'}, x') \le d(r_{i',k'}, x)$. Therefore, we have $(i' - i_2)L + (k' - k_2)T \le (i' - i_1)L + (k' - k_1)T$, i.e., $(i_2 - i_1)L + (k_2 - k_1)T \ge 0$. Therefore, we have $d(r_{i,k}, x) = (i - i_1)L + (k - k_1)T = (i - i_2)L + (k - k_2)T + (i_2 - i_1)L + (k_2 - k_1)T = d(r_{i,k}, x') + (i_2 - i_1)L + (k_2 - k_1)T \ge d(r_{i,k}, x')$. Obviously, this contradicts $r_{i,k} \in R(x)$. Hence, the property is proven. □

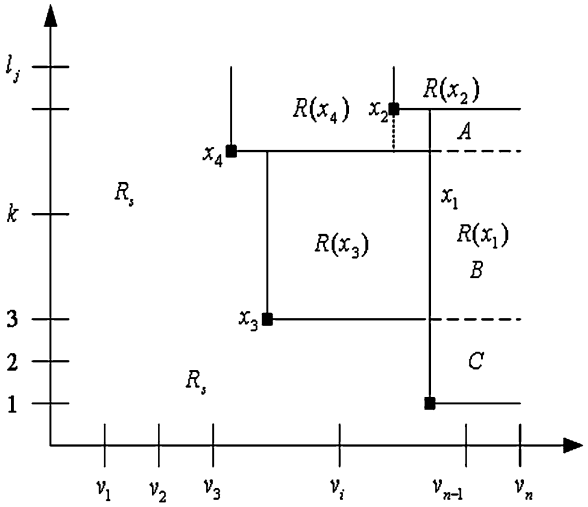


Fig. 3 Example for calculating $l(x)$

From Property 1, we can see that $R(x)$ should be a region that can be divided into several rectangular regions. This can be seen from Fig. 3. For example, $R(x_4)$ can be divided into two regions by the vertical broken line from x_2.

Regarding to calculating $l(X_{i,k_i})$, we first give the following theorem.

Theorem 2 *Suppose that only X_{i,k_i} is cached at node v_i, then we have*

$$l(X_{i,k_i}) = \sum_{r_{i,k} \in B_0} f_{i,k}\left[i \cdot L - d(r_{i,k}, X_{i,k_i})\right] + \sum_{i=1}^{n} \sum_{r_{i,k} \in B_i} f_{i,k}\left[d\left(r_{i,k}, X^i_{k_i}\right) - d(r_{i,k}, X_{i,k_i})\right] \tag{6}$$

where $B_0 = \{(\alpha, \beta) | (\alpha = i_0, \beta \in R_0 \cap R(X_{i,k_i}))\} \cap R(X_{i,k_i})$ *and* $B_i = \{(\alpha, \beta) | (\alpha = i_0, \beta \in R(X^i_{k_i}) \cap R(X_{i,k_i}))\} \cap R(X_{i,k_i})$.

Proof It is obvious that $B_i \cap B_j = \phi$ for $i \neq j$. This guarantees that each request's access cost is only calculated one time. Now, we prove the correctness of the calculation of $l(X_{i,k_i})$, i.e., the requests in B_i should be served by $X^i_{k_i}$. Suppose that there exists a request $r_{i',k'} \in B_i$ which is not served by $X^i_{k_i}$. Based on Property 1, we have all the requests in the region $B'_i = \{(\alpha, \beta) | i \leq \alpha \leq i_0, k_i \leq \beta \leq k_0\}$ will be not served by $X^i_{k_i}$. It is easy to see that $R(X^i_{k_i}) \cap B'_i \neq \phi$, i.e., there exist some requests in region $R(X^i_{k_i})$ that are not served by $X^i_{k_i}$. This obviously contradicts the fact that all the requests in region $R(X^i_{k_i})$ are served by $X^i_{k_i}$. Hence, the theorem is proven. □

For example, in Fig. 3, if x_1 is removed, $R(x_1)$ can be divided into three regions (i.e., A, B, and C), which will be served by x_4, x_3, and the server, respectively. Thus,

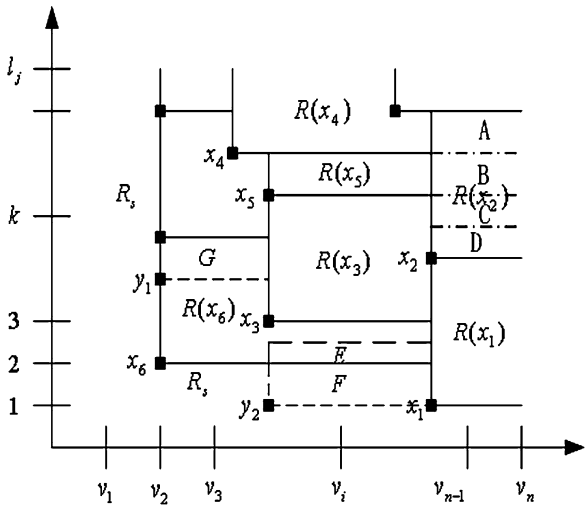


Fig. 4 Example for calculating $l(x)$

we have $l(x_1) = \sum_{r_{i,k} \in A} f_{i,k}[d(r_{i,k}, x_4) - d(r_{i,k}, x_1)] + \sum_{r_{i,k} \in B} f_{i,k}[d(r_{i,k}, x_3) - d(r_{i,k}, x_1)] + \sum_{r_{i,k} \in B} f_{i,k}[i \cdot L - d(r_{i,k}, x_1)]$.

In practice, the general case is that several versions of the same multimedia object are cached at node v_i at the same time (see Fig. 4). In this case, calculating $l(x)$ should also consider the mutual effect of the least more detailed cached version on the removed version since the requests served by the removed version could be satisfied by this detailed version. For example, when calculating $l(x_2)$, $R(x_2)$ might be divided into four parts A, B, C, and D which will be served by x_4, x_5, x_3, and x_1, respectively.

Taking into consideration the caching dependence along the path, calculating $l(X_{i,k_i})$ becomes more complex and it is so obvious to obtain an optimal solution.

Similarly, we can calculate the cost saving of caching O_{i_0} at node v_i. For example in Fig. 4, if $i_0 = y_1$, then $R(x_6)$ can be divided in to two parts: E and F; if $i_0 = y_2$, then $R(x_6)$ can also be divided in to two parts. So, we have $g(y_1) = \sum_{r_{i,k} \in G} f_{i,k}[d(r_{i,k}, y_1) - d(r_{i,k}, x_6)]$ and $g(y_2) = \sum_{r_{i,k} \in E} f_{i,k}[d(r_{i,k}, x_6) - d(r_{i,k}, y_1)] + \sum_{r_{i,k} \in F} f_{i,k}[i \cdot L - d(r_{i,k}, y_1)]$.

Now, we begin to present an optimal solution for the problem as defined in (2). In the following, we call the problem a k-optimization problem if we determine cache replacement candidates from nodes $\{v_1, v_2, \ldots, v_k\}$. Thus, the original problem (2) is an n-optimization problem. Theorem 3 shows an important property that the optimal solution for the whole problem must contain optimal solutions for some subproblems.

Theorem 3 *Suppose that* $X = \{X_{i_1,k_{i_1}}, X_{i_2,k_{i_2}}, \ldots, X_{i_\alpha,k_{i_\alpha}}\}$ *is an optimal solution for the* α*-optimization problem and* $X' = \{X_{i'_1,k_{i'_1}}, X_{i'_2,k_{i'_2}}, \ldots, X_{i'_\beta,k_{i'_\beta}}\}$ *is an optimal solution for the* $k_{i_\alpha} - 1$*-optimization problem. Then* $X^* = \{X_{i'_1,k_{i'_1}}, X_{i'_2,k_{i'_2}}, \ldots, X_{i'_\beta,k_{i'_\beta}}, X_{i_\alpha,k_{i_\alpha}}\}$ *is also an optimal solution for the* α*-optimization problem.*

Proof By definition, we first have $L(X^*) = l(X_{i'_1,k_{i'_1}}) + l(X_{i'_2,k_{i'_2}}) + \cdots + l(X_{i'_\beta,k_{i'_\beta}}) + l(X_{i_\alpha,k_{i_\alpha}}) = L(X') + l(X_{i_\alpha,k_{i_\alpha}}) \geq l(X_{i_1,k_{i_1}}) + l(X_{i_2,k_{i_2}}) + \cdots + l(X_{i_\beta,k_{i_\beta}}) + l(X_{i_\alpha,k_{i_\alpha}}) = L(X)$. On the other hand, since X is an optimal solution for the α-optimization problem, we have $L(X) \geq L(X^*)$. Therefore,we have $L(X) = L(X^*)$. Hence, the theorem is proven. □

Based on Theorem 3, an optimal solution for the n-optimization can be obtained by checking all possible removed candidates from node v_1 to node v_n in order. Therefore, it is east to get that the time complexity of this solution is $O(n^2 + mn \log n)$ based on our previous result that the complexity for computing all $S(r_{i,k})$ is $O(mn \log n)$, where n is the number of nodes in the network and m is the number of versions of object j.

Now, we can derive an optimal solution for (2) for tree networks by using the following two theorems. The proofs of Theorems 4 and 5 can be found in [20].

Theorem 4 *For tree T_r, if $C(r) = \{r_1, r_2, \ldots, r_m\}$, then we have $P^*_r = \bigcup_{i=1}^{m} P^*_{r,r_i}$.*

Theorem 5 *For tree $T_{r,w}$, if $C(w) = \{w_1, w_2, \ldots, w_t\}$, then we have*

$$P^*_{r,w} = \begin{cases} \bigcup_{i=1}^{t} P^*_{r,w_i}, & L(\bigcup_{i=1}^{t} P^*_{r,w_i}) \geq L(P^*_w \cup \{w\}) \\ P^*_w \cup \{w\}, & L(\bigcup_{i=1}^{t} P^*_{r,w_i}) < L(P^*_w \cup \{w\}) \end{cases}$$

By Theorem 4, we can see that the optimal solution for tree T_r can be decomposed into the combination of the solutions for subtrees $\{T_{r,w}, w \in C(r)\}$. By Theorem 5, we can see that the optimal solution for tree $T_{r,w}$ can be further divided until the division cannot be proceeded according to the relationship between $L(\bigcup_{i=1}^{t} P^*_{r,w_i})$ and $L(P^*_w \cup \{w\})$. Therefore, based on Theorems 4 and 5, a dynamic programming-based algorithm can be proposed to solve the problem formulated in (2).

4.3 Cooperative cache replacement scheme

Based on the previous analysis, we present the following cooperative cache replacement scheme. In our scheme, every cache maintains some information about the objects in the form of object descriptors. An object descriptor contains information that includes the object size and the access frequencies for all versions of the multimedia objects. When an updated version of a multimedia object is to be cached or a request for a version of a multimedia object arrives at the server, it should be cached at those nodes where a version of this object is cached and the cache replacement candidates are decided according to our proposed solution.

Since the cache contents change over time, the access frequency and the cost loss of an object with respect to a node must be refreshed from time to time. The access frequency can be estimated based on recent request history, which is locally available (e.g., by using a "sliding window" technique [27]). The cost loss is updated by the response messages. Specifically, a variable with an initial value of zero is attached to each object. At each intermediate node along the way, the variable is increased by the

cost of the last link the object has just traversed. The value is then used to update the cost loss of the object maintained by the associated cache. If the object is inserted into the cache, the node resets the value to zero before forwarding the object downstream. In this way, the updated cost loss is disseminated to all the caches on the way.

5 Simulation model

To the best of our knowledge, it is difficult to find true trace data in the open literature to simulate our model. Therefore, we generated the simulation model from the empirical results presented in [1, 5, 7, 9, 13, 19].

The network topology was randomly generated by the Tier program [9]. Experiments for many topologies with different parameters have been conducted and the relative performance of our model was found to be insensitive to topology changes. Here, only the experimental results for one topology was listed due to space limitations. The characteristics of this topology and the workload model are shown in Table 2, which are chosen from the open literature and are considered to be reasonable.

The Wide Area Network (WAN) is viewed as the backbone network to which no servers or clients are attached. Each Metropolitan Area Network (MAN) node is assumed to connect to a content server. Each MAN and WAN node is associated with an en route cache. Similar to the studies in [7, 12, 17, 27, 29], the 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 130 KB. We also assume that each multimedia object has at

Table 2 Parameters used in simulation

Parameter	Value
Number of WAN Nodes	200
Number of MAN Nodes	200
Delay of WAN Links	Exponential Distribution $p(x) = \theta^{-1} e^{-x/\theta}$
Delay of MAN Links	Exponential Distribution $p(x) = \theta^{-1} e^{-x/\theta}$
Number of Servers	100
Number of Web Objects	1000 objects per server
Web Object Size Distribution	Pareto Distribution $p(x) = \frac{ab^a}{a-1}$
Web Object Access Frequency	Zipf-Like Distribution $\frac{1}{i^\alpha}$
Average Request Rate Per Node	$U(1, 9)$ requests per second

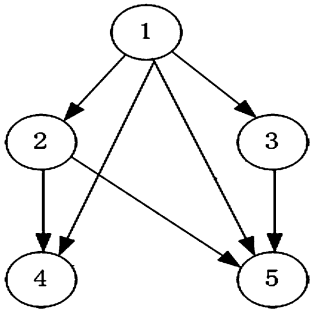


Fig. 5 Transcoding graph for simulation

most five versions and that the transcoding graph is as shown in Fig. 5. If an object has less than five versions, some nodes in the transcoding graph is null. For the case in which an object has only one version, then the transcoding graph degenerates to a node. The transcoding delay is determined as the quotient of the object size to the transcoding rate, which is assumed as 20 KB/Sec in the simulation. In the simulation, the client at each 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 [7, 22]. In the simulation, α is set as 0.7. 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, where the average delay for WAN links is 0.45 seconds and the average delay for WAN links is 0.06 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 transcoding delay. The cost function is taken to be the delay of the link, which means that the cost in our model is interpreted as the access latency in our simulation.

We apply a "sliding window" technique to estimate the access frequency to make our model less sensitive to transient workload [27]. Specifically, for each object O, $f(O, v)$ is calculated by $K/(t - t_K)$, where K is the number of accesses recorded, t is the current time, and t_K is the Kth most recently referenced time (the time of the oldest reference in the sliding window). K is set to 2 in the simulation. To reduce overhead, the access frequency is only updated when the object is referenced and at reasonably large intervals, e.g., several minutes, to reflect aging, which is also applied in [29].

In addition to the model presented in Sect. 4.2, we also consider the following caching models for comparison purposes.

LRU: *LRU* is a solution that evicts the web object which is requested the least recently. In *LRU*, different versions of the same multimedia object are considered as different objects.

AE [13]: *AE* is a solution that explores the aggregate effect of caching multiple versions of the same multimedia object in the cache.

LT: *LT* is a solution for cooperated multimedia object replacement in transcoding proxies in linear topology proposed in this paper.
TN: *TN* is a solution for cooperated multimedia object replacement in transcoding proxies in tree networks proposed in this paper.

6 Performance evaluation

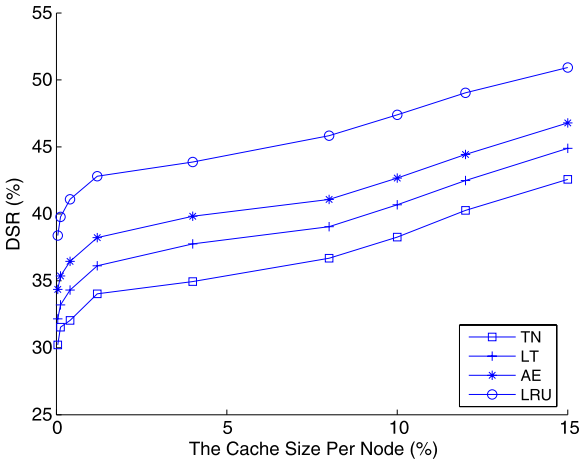
In this section, we compare the performance results of our solutions (proposed in Sect. 4.2) with those models introduced in Sect. 5 in terms of several performance metrics. The performance metrics we used in our simulation include delay-saving ratio (*DSR*), which is defined as the fraction of communication and server delays which is saved by satisfying the references from the cache instead of the server, average access latency (*AAL*), request response ratio (*RRR*), which is defined as the ratio of the access latency of the target object to its size, average hit ratio (*AHR*), which is defined as the ratio of the number of requests satisfied by the caches (including the requests satisfied by transcoding from the cached version) as a whole to the total number of requests, exact hit ratio (*EHR*), which is defined as the ratio of the number of requests satisfied by the exact version in the caches as a whole to the total number of requests, and highest server load (*HSL*), which is defined as the largest number of bytes served by the server per second. In the following figures, *LRU*, *AE*, *LT*, and *NT* denote the results for the solutions introduced in Sect. 5

6.1 Impact of cache size

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

The first experiment investigates *DSR* as a function of the relative cache size per node and Fig. 6 shows the simulation results. As presented in Fig. 6, we can see that our solutions outperform existing solutions since our coordinated cache replacement

Fig. 6 Experiment on *DSR*



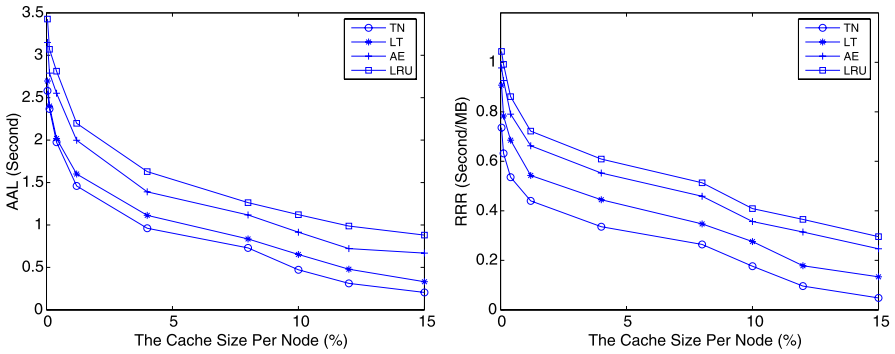


Fig. 7 Experiment on *AAL* and *RRR*

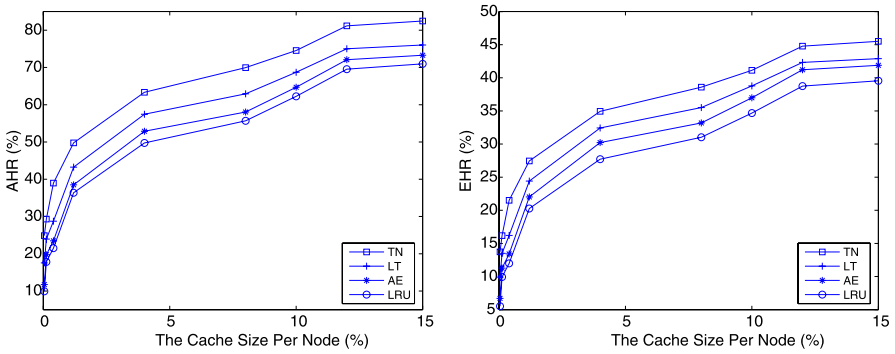


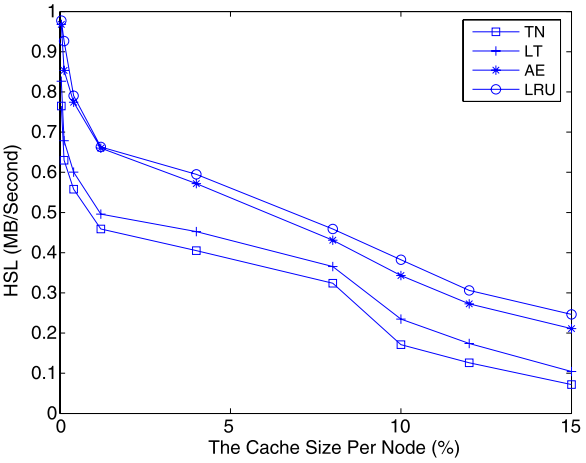
Fig. 8 Experiment for *AHR* and *EHR*

model determines the replacement candidates cooperatively among all the nodes for linear topology and tree networks, whereas existing solutions, including *LRU* and *AE*, decide cache replacement candidates locally, i.e., only from the view of a single node. We can also see that our proposed solution for tree networks outperforms the proposed solution for linear topology in that a linear topology can be viewed as a special case of tree networks when a tree has only one path from the server to the client.

Figure 7 shows the simulation results of *AAL* and *RRR* as a function of the relative cache size at each node. Clearly, the lower the *AAL* 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 both *TN* and *LT* significantly improve both *AAL* and *RRR* compared to *AE* and *LRU* since our proposed solutions determines the cache replacement candidates in an optimal and coordinated way, while the others decide the replacement candidates only by considering the situation of a single node.

Figure 8 shows the results of *EHR* and *AHR* as a function of the relative cache size for different solutions. By computing the optimal replacement candidates, we can see that the results for our solutions can greatly outperform those of the other solutions, especially for smaller cache sizes. We can also see that *AHR* and *EHR*

Fig. 9 Experiment for *HSL*



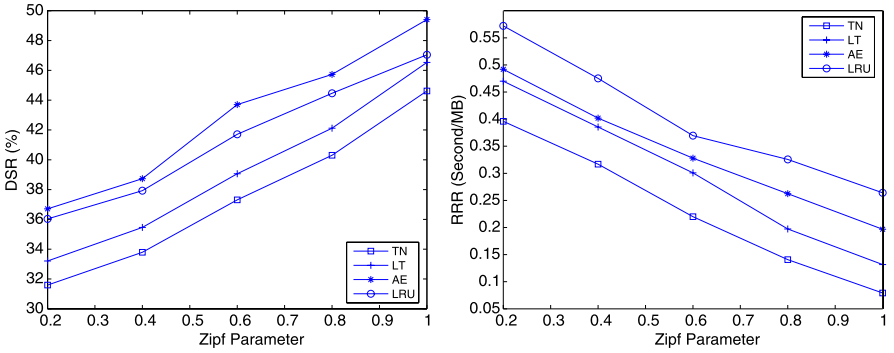


Fig. 10 Experiment for *DSR* and *RRR*

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.

Figure 9 also shows the results of *HSL* as a function of the relative cache size.

6.2 Impact of object access frequency

This experiment set examines the impact of object access frequency distribution on the performance results of different models. The relative cache size for this experiment set is 10%. Figures 10 and 11 shows the performance results of *DSR*, *RRR*, *AHR*, and *EHR*, respectively, for the values of Zipf parameter α from 0.2 to 1.0. We can see that both *TN* and *LT* consistently provide better performance over a wide range of object access frequency distributions.

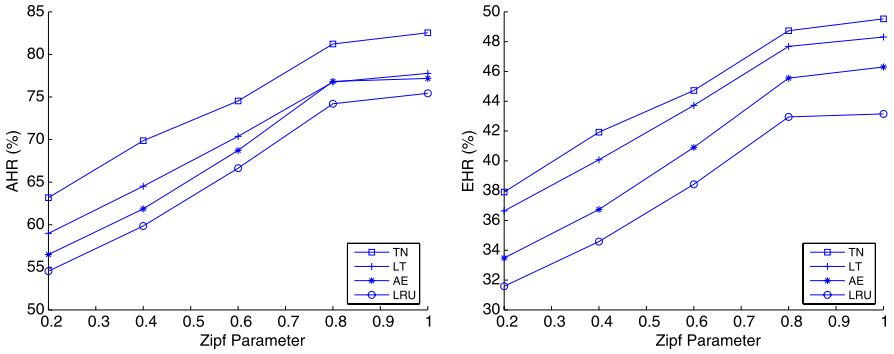


Fig. 11 Experiment for *AHR* and *EHR*

7 Conclusion

The transcoding proxy is attracting more and more attention since it plays an important role in the functionality of web caching. In this paper, we presented a coordinated cache replacement model in transcoding proxies where multimedia object placement and replacement policies are managed in a coordinated way. Our model is formulated as an optimization problem and the optimal solution is obtained using a low-cost dynamic programming-based solution. Extensive simulation experiments have been performed to compare the proposed coordinated cache replacement model with several existing models. The results show that our model effectively improves delay-saving ratio, average access latency, request response ratio, object hit ratio, and highest server load. The proposed coordinated cache replacement model considerably outperforms local cache replacement models that consider cache replacement at individual nodes only. Our model has wide applications for web caching, especially for the case in which multimedia objects are involved. However, it is a challenging task for our future research to solve this problem in the general case, i.e., the case in which different versions of the same multimedia object are of different sizes. The techniques of applying dynamic programming shown in this paper may serve as useful tools for deriving such solutions in the general case.

References

1. Aggarwal C, Wolf JL, Yu PS (1999) Caching on the world wide web. IEEE Trans Knowl Data Eng 11(1):94–107
2. Anderson TE, Dahlin MD, Neefe JN, Patterson DA, Rosselli DS, Wang RY (1995) Serverless network file systems. In: Proc of the 15th symposium on operating systems principles, pp 109–126
3. Awerbuch B, Bartal Y, Fiat A (1998) Distributed paging for general networks. J Algorithm 28:67–104
4. Balamash A, Krunz M (2004) An overview of web caching replacement algorithms. IEEE Commun Surv Tutor 6(2):44–56
5. Barford P, Crovella M (1998) Generating representive web workloads for network and server performance evaluation. In: Proc of ACM SIGMETRICS'98, pp 151–160
6. Blaze MA (1993) Caching in large-scale distributed file systems. Technical Report TR-397-92, Department of Computer Science, Princeton University

7. Breslau L, Cao P, Fan L, Phillips G, Shenker S (1999) Web caching and zip-like distributions: evidence and implications. In: Proc of IEEE INFOCOM'99, pp 126–134
8. Bowman CM, Danzig PB, Hardy DR, Manber U, Schwartz MF (1994) The harvest information discovery and access system. In: Proc of the 2nd international world wide web conference, pp 763–771
9. Calvert KL, Doar MB, Zegura EW (1997) Modelling Internet topology. IEEE Commun Mag 35(6):160–163
10. Canali C, Cardellini V, Colajanni M, Lancellotti R, Yu PS (2005) A two-level distributed architecture for efficient web content adaptation and Delivery. In: Proc of 2005 IEEE/IPSJ symposium on applications and the Internet, pp 132–139, Trento, Italy, Jan 2005
11. Canali C, Cardellini V, Colajanni M, Lancellotti R, Yu PS (2003) Cooperative architectures and algorithms for discovery and transcoding of multi-version content. In: Proc of 8th int'l workshop on web content caching and distribution, pp 205–221, Hawthorne, NY, Sept 2003
12. Cao P, Irani S (1997) Cost-aware WWW proxy caching algorithms. In: Proc of first USENIX symposium on Internet technologies and systems (USITS), pp 193–206
13. Chang C, Chen M (2003) On exploring aggregate effect for efficient cache replacement in transcoding proxies. IEEE Trans Parallel Distrib. Syst 14(6):611–624
14. Chankhunthod A, Danzig P, Neerdaels C, Schwartz M, Worrell K (1996) A hierarchical Internet object cache. In: Proc of the USENIX technical conference, pp 22–26
15. Dahlin MD, Wang RY, Anderson TE, Patterson DA (1994) Cooperative caching: using remote client memory to improve file system performance. In: Proc of first symp operating systems design and implementations, pp 267–280
16. Fan L, Cao P, Almeida J (1998) Summary cache: a scalable wide-area web cache sharing protocol. In: Proc of ACM SIGCOMM conference, pp 254–265
17. Jin S, Bestavros A (2001) Greeddual* web caching algorithm: exploiting the two sources of temporal locality in web request streams. Comput Commun 4(2):174–183
18. Leff A, Wolf JL, Yu PS (1993) Replication algorithms in a remote caching architecture. ACM Trans Parallel Distrib Syst 4(11):1185–1204
19. Li K, Shen H (2005) Coordinated en-route multimedia object caching in transcoding proxies for tree networks. ACM Trans Multimedia Comput Commun Appl (TOMCAPP) 5(3):289–314
20. Li K, Shen H, Chin F, Zheng S (2005) Optimal methods for coordinated en-route web caching for tree networks. ACM Trans Internet Technol 5(3):480–507
21. Korupolu MR, Dahlin M (2002) Coordinated placement and replacement for large-scale distributed caches. IEEE Trans Knowl Data Eng 14(6):1317–1329
22. Padmanabhan VN, Qiu L (2000) The content and access dynamics of a busy site: findings and implications. In: Proc of ACM SIGCOMM'00, pp 111–123, August 2000
23. O'Neil EJ, O'Neil PE, Weikum G (1999) An optimality proof of the LRU-K page replacement algorithm. J ACM 46(1):92–112
24. Podlipnig S, Boszormenyi L (2003) A survey of web cache replacement strategies. ACM Comput Surv 35(4):374–398
25. Psounis K, Prabhakar B (2002) Efficient randomized web-cache replacement schemes using samples from past eviction-times. IEEE/ACM Trans Netw 10(4):441–454
26. Shen B, Lee S-J, Basu S (2004) Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks. IEEE Trans Multimedia 6(2):375–386
27. Shim J, Scheuermann P, Vingralek R (1999) Proxy cache algorithms: design, implementation, and performance. IEEE Trans Knowl Data Eng 11(4):549–562
28. Singh A, Trivedi A, Ramamritham K, Shenoy P (2004) PTC: proxies that transcode and cache in heterogeneous web client environments. World Wide Web 7(1):7–28
29. Tang X, Chanson ST (2002) Coordinated en-route web caching. IEEE Trans Comput 51(6):595–607
30. Tewari X, Dahlin M, Vin HM, Kay JS (1999) Design considerations for distributed caching on the Internet. In: Proc of the 19th int'l conference distributed computing systems (ICDCS), pp 273–284
31. Williams S, Abrams M, Standbridge CR, Abdulla G, Fox EA (1996) Removal policies in network caches for world wide web documents. In: Proc of ACM SIGCOMM'96, pp 293–305
32. Xu J, Li B, Li DL (2002) Placement problems for transparent data replication proxy services. IEEE J Sel Areas Commun 20(7):1383–1398

Keqiu Li received the Bachelor and Master degrees from the Department of Applied Mathematics at the Dalian University of Technology in 1994 and 1997, respectively. He received the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology in 2005. Doctor Keqiu Li also has 2 years' postdoctoral experience in the University of Tokyo, Japan. He is currently Professor in the Department of Computer Science and Engineering, Dalian University of Technology, China. Doctor Keqiu Li has published more than 70 technical papers. He has served in an editorial role for several international journals. His research interests include Internet technology, computer networks, trusted computing.

Yanming Shen is Associate Professor in the Department of Computer Science and Engineering at Dalian University of Technology, China. He received the Ph.D. degree from Department of Electrical and Computer Engineering at the Polytechnic University and his B.Sc. degree in Automation from Tsinghua University, China in 2000. He was a summer intern with Avaya Labs in 2006, conducting research on IP telephony. His general research interests include packet switch design, peer-to-peer video streaming, and algorithm design, analysis and optimization.

Kai Lin is a lecturer at the School of Electronic and Information Engineering of Dalian University of Technology, China. He received his M.Sc. and Ph.D. degrees from the College of Information Science and Engineering of Northeastern University in China in 2005 and 2008, respectively. His research interests include wireless networks, ubiquitous computing, and embedded technology.

Wenyu Qu received her Bachelor and Master degrees both from Dalian University of Technology, China in 1994 and 1997, and her Doctoral degree from Japan Advanced Institute of Science and Technology in 2006. She is currently Professor at the School of Information Science and Technology, Dalian Maritime University, China. Wenyu Qu's research interests include mobile agent-based technology, distributed computing, computer networks, and grid computing. Wenyu Qu has published more than 40 technical papers in international journals and conferences. She is on the committee board for a couple of international conferences.