Transcoding Proxy Placement in En-Route Web Caching*

Keqiu Li and Hong Shen
Graduate School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Tatsunokuchi, Ishikawa, 923-1292, Japan

Email: {keqiu,shen}@jaist.ac.jp

Abstract

With the rapid growth of audio and video applications on the internet, caching media objects in transcoding proxies has become an important research topic in recent years. In this paper, we address the problem of finding the optimal locations for placing fixed number of transcoding proxies among the nodes in a network such that the specified objective is achieved. We present an original model for this problem, which makes transcoding proxy placement decisions on all the en-route nodes along the routing path in a coordinated way. In our model, proxy status information along the routing path of requests is used for optimally determining the locations for placing fixed number of transcoding proxies. We formulate this problem as an optimization problem and the optimal locations are obtained using a low-cost dynamic programming-based algorithm. We implement our algorithm and evaluate our model on different performance metrics through extensive simulation experiments. The implementation results show that our model significantly outperforms the random algorithm which places transcoding proxies among the nodes in a network randomly.

Key words: Transcoding caching, dynamic programming, optimization problem, World Wide Web.

1 Introduction

Transcoding is defined as a transformation that is used to convert a media object from one form to another, frequently trading off object fidelity for size. Transcoding caching is attracting more and more attention since it plays an important role in the functionality of caching [3, 10, 15].

Transcoding can be executed at various components in the network such as server, proxy, and client. For the case of client, it can preserve the original semantic of system architecture and transport protocols. However, it is extremely expensive when the clients are mobile users due to the limitation of connection bandwidth and power. For the case of server, it is not necessary to perform transcoding during the time between the client issuing a request and the server responsing to it; thus no additional transcoding delay will be incurred. At the same time, it will take too much storage space to keep all the versions of the same media object. Further, it is not flexible in dealing with the future change of clients' needs. For these reasons, it will be better to transcode the media objects in intermediate proxies. Many research has been focused on exploring the advantage of this approach [6, 9, 10], where an intermediate proxy is capable of transcoding the requested media object to a proper version according to the client's specification before it sends this media object to the client. We refer such an intermediate proxy as a transcoding proxy in this paper.

To obtain the full benefits of transcoding caching, different architectures have been employed [17, 20]. En-route caching is a new caching architecture developed recently

*This work was supported by Japan Society for the Promotion of Science (JSPS) under its General Research Scheme **B** Grant No. 14380139).

[13, 19] in which proxies are placed on the access path from the user to the server. Each en-route proxy intercepts any request that passes through its associated node, and either satisfies the request by sending the requested media object to the client or forwards the request upstream along the path to the server until it can be satisfied. Cooperative caching, in which proxies cooperate in serving each other's requests, is a powerful paradigm to improve transcoding caching effectiveness [8, 11, 12]. For web caching in transcoding proxies, proxy collaboration is more important since transcoding can be executed from one version of a media object to another.

For coordinated en-route web caching in transcoding proxies, an important issue is to find the optimal locations for placing K transcoding proxies among the en-route nodes in a network such that the specified objective is achieved. In this paper, we present an original model for this problem. In our model, transcoding proxy placement decisions are made on all the en-route nodes along the routing path in a coordinated way and proxy status information along the routing path of requests is used for optimally determining the locations for placing K transcoding proxies. We formulate this problem as an optimization problem and the optimal locations are obtained using a low-cost dynamic programming-based algorithm. We implement our algorithms and evaluate our model on different performance metrics through extensive simulation experiments. The implementation results show that our model significantly outperforms random algorithms that determine the locations for placing K transcoding proxies in a random way.

The rest of the paper is organized as follows. Section 2 formulates the problem of finding the optimal locations for placing K transcoding proxies among the en-route nodes. Section 3 presents a dynamic programming-based algorithm for solving our problem, describes and discusses a coordinated transcoding caching scheme for implementation. Section 4 and Section 5 describe the simulation model and discusses the performance results, respectively. Section 6 summaries our work and concludes the paper.

2 Problem Formulation

As introduced in previous sections, transcoding is a transformation that is used to convert a media object from one form to another, frequently trading off object fidelity for size. Without loss of generality, we assume that each object has m different versions. The original version, which can be transcoded to a less detailed one and such a transcoded object is called the transcoded version, is denoted as A_1, whereas the least detailed version, which cannot be transcoded any more, is denoted as A_m . The relationship among different versions of a media object can be expressed by weighted transcoding graph [7], which can be viewed as an extension to the transcoding relation graph [5]. An example of weighted transcoding graph is given in Figure 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 transcode to A_j where A_i is a more detailed version than A_j since it is possible that A_i does not contain enough content information for the transcoding from A_i to A_j. For example, transcoding can not be executed between A_4 and A_5 due to insufficient content information.

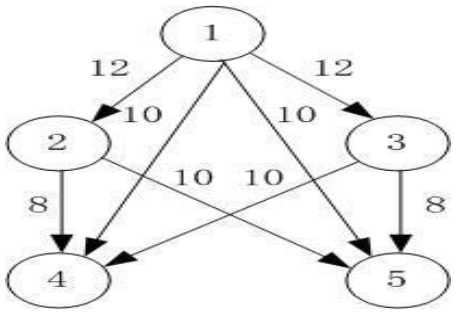

Figure 1. An Example of Weighted Transcoding Graph

Regarding the weighted transcoding graph, we give the following definition.

Definition 1 *The weighted transcoding graph, denoted by $W = (N, L)$, is a directed graph, where N is the set of different versions of a media object and L the transcoding relationship between two different versions. The weight for each edge is the transcoding cost from one version to another.*

We model the network as a graph $G = (V, E)$, where $V = (v_1, v_2, \cdots, v_n)$ is the set of nodes, and E is the set of network links. The media objects are maintained by content servers. Each media object is served by exactly one server. The set of all the media objects is de-

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 IEEE

Table 1. A List of the Symbols

Symbol	Decription
$O = (O_1, O_2, \cdots, O_l)$	the set of all the media objects
$A_h = (A_{h,1}, A_{h,2}, \cdots, A_{h,m})$	the set of all the versions of O_h
$b_{A_{h,i}}$	the size of $A_{h,i}$
$V = (v_1, v_2, \cdots, v_n)$	the set of nodes in a network
$v_j^+(A_{h,i})$	the nearest higher level node of v_j where $A_{h,i}$ is cached
B_{h,v_j}	the version of O_h placed at node v_j
$D_{A_{h,i}}$	the less detailed versions of O_h than $A_{h,i}$ including $A_{h,i}$
$f_{A_{h,i},v_j}$	the mean access frequency of $A_{h,i}$ from v_j
c_{v_i,v_j}	the cost of transmitting a media object between v_i and v_j
$w(A_{h,i}, A_{h,j})$	the transcoding cost from $A_{h,i}$ to $A_{h,j}$ for a media object

noted by $O = (O_1, O_2, \cdots, O_l)$. In our analysis, we assume that each media O_h object has m_h versions, denoted by $A_h = (A_{h,1}, A_{h,2}, \cdots, A_{h,m_h})$. $b_{A_{h,i}}$ is the size of $A_{h,i}$. We assume that the access frequencies for $A_{h,i}$ from v_j, denoted by $f_{A_{h,i},v_j}$, are independent. The cost of transmitting a media object between v_i and v_j is denoted by c_{v_i,v_j}. 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 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. The corresponding transcoding relation graph among different versions of a media object is denoted by W and the transcoding cost of a media object from $A_{h,i}$ to $A_{h,j}$ is given by the weight on the edge $(A_{h,i}, A_{h,j})$, which is denoted by $w(A_{h,i}, A_{h,j})$. $D(A_{h,i})$ is the set of all versions that can be transcoded from $A_{h,i}$. If a version can not be directly transcoded from the version cached, then the transcoding cost is the least reachable transcoding cost from the version cached. A list of symbols is given in Table 1. Our mathematical model is formulated based on these symbols.

As we mentioned above, it is necessary and important to find a method to optimally determine the locations for placing K transcoding proxies among the en-route nodes such that the specified objective is obtained, since we cannot place at each node a transcoding proxy. Placing a transcoding proxy at a node enables the requests previously pass through it to be satisfied here; therefore, the cost, including transmission cost and transcoding cost, will be saved and we define it as *single cost gain* in this paper. We start with computing the single cost gain of placing a transcoding proxy at a node, which is defined as follows.

Definition 2 *$g(v_j)$ is a function for calculating the single cost gain of placing a transcoding proxy at node v_j.*

$$g(v_j) = \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_j})} f_{A_{h,k},v_j} \cdot (c_{v_j^+(A_{h,k}),v_j} + w(B_{h,v_j^+(A_{h,k})}, A_{h,k}) - w(B_{h,v_j}, A_{h,k})) \quad (1)$$

where $v_j^+(A_{h,k})$ is the nearest higher level node of v_j at which $A_{h,k}$ is cached, and B_{h,v_j} is the version of O_h cached at node v_j.

Now we start to formulate the problem of finding the optimal locations for placing K transcoding proxies among the en-route nodes. Consider the snapshot when requests from clients are being served (see Figure 2).

Let 0 be the content server, n be the client issuing the requests, and $1, 2, \cdots, n-1$ are the en-route nodes on the path from 0 to n. Based on the single cost gain of placing a transcoding proxy at a node, we define the *aggregate cost gain* of placing K transcoding proxies on the path from node 0 to n as follows.

Definition 3 *Given $K, f_{A_{h,i},v_j}$, c_{v_i,v_j} and $w(A_{h,i}, A_{h,j})$ where $(i = 1, 2, \cdots, m; j = 1, 2, \cdots, n; h = 1, 2, \cdots, l)$. Let $v_1, v_2, \cdots, v_K$ be a set of K nodes such that $1 \leq v_1 \leq v_2 \leq \cdots \leq v_K \leq n$. Based on Equation (1), the aggregate cost gain of placing K transcoding proxies at*

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 **IEEE**

Figure 2. Coordinated En-Route Web Caching in Transcoding Proxies

$v_1, v_2, \cdots, v_K$ *which is denoted by* $S(n : v_1, v_2, \cdots, v_K)$ *is defined as*

$$G(n : v_1, v_2, \cdots, v_K) = \sum_{j=1}^{K} \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_j})} \left(f_{A_{h,k},v_j} - f_{A_{h,k},v_{j+1}}\right) \cdot (c_{v_j^+(A_{h,k}),v_j} + w(B_{h,v_j^+(A_{h,k})}, A_{h,k}) - w(B_{h,v_j}, A_{h,k})) \quad (2)$$

where $v_j^+(A_{h,k})$ *is the nearest higher level node of* v_j *at which* $A_{h,k}$ *is cached,* v_j^- *is the nearest lower level node of* v_j *at which a transcoding proxy is placed, and* B_{h,v_j} *is the version of* O_h *cached at node* v_j*. If* $K = 0$*, then we define* $G(n : \phi) = 0$*. Finding* $v_1, v_2, \cdots, v_K$ *that maximizes* $G(n : v_1, v_2, \cdots, v_K)$ *is referred to as the* $n \cdot K$*-optimization problem.*

Obviously, our objective is to compute the locations for placing K transcoding proxies in a subset of nodes $\{v_1, v_2, \cdots, v_k\}$ that maximizes the aggregate cost gain as defined in (2).

3 Dynamic Programming Based Solution

Before solving the problem described in (2), we give the following definition.

Definition 4 *Given* $f_{A_{h,i},v_j}$*,* c_{v_i,v_j} *and* $w(A_{h,i}, A_{h,j})$*,* $(i = 1, 2, \cdots, m; j = 1, 2, \cdots, n; h = 1, 2, \cdots, l)$*. Let* $v_1, v_2, \cdots, v_k$ *be a set of* k *nodes such that* $1 \leq v_1 \leq v_2 \leq \cdots \leq v_k \leq n$*. Based on Equation (2), the* constrained aggregate cost gain *of placing* k *transcoding proxies at* $v_1, v_2, \cdots, v_k$*, which is denoted by* $H(n, \alpha : v_1, v_2, \cdots, v_k)$*, is defined as*

$$H(n, \alpha : v_1, v_2, \cdots, v_k) = \sum_{j=1}^{k} \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_j})} [(f_{A_{h,k},v_j} - f_{A_{h,k},v_{j+1}}) \cdot (c_{v_j^+(A_{h,k}),v_j} + w(B_{h,v_j^+(A_{h,k})}, A_{h,k}) - w(B_{h,v_j}, A_{h,k})) - \alpha] \quad (3)$$

where $v_j^+(A_{h,k})$ *is the nearest higher level node of* v_j *at which* $A_{h,k}$ *is cached,* v_j^- *is the nearest lower level node of* v_j *at which a transcoding proxy is placed, and* B_{h,v_j} *is the version of* O_h *cached at node* v_j*. If* $k = 0$*, then we define* $H(n, \alpha : \phi) = 0$*. Finding* $v_1, v_2, \cdots, v_k$ *that maximizes* $H(n, \alpha : v_1, v_2, \cdots, v_k)$ *is referred to as the* n*-optimization problem.*

Before presenting an algorithm to solve (3), we discuss the relationship between the solutions to (2) and (3).

From (3), we can easily get that the number of transcoding proxies to be placed in the network is relevant to the parameter α greatly. Hence, the proper selection of α determines the optimal number of transcoding proxies to be located among the en-route nodes in a network. The crucial observation is that the optimal number of transcoding proxies to be located is a monotonically decreasing function of α, that is, as α increases, the optimal number decreases monotonically. Therefore, we can determine the optimal locations for placing K transcoding proxies among the en-route nodes by tuning the parameter α. The relationship between the optimal number of transcoding proxies k^* to be placed and the parameter α can be visualized in Figure 3. Therefore, we can solving (2) by tuning the parameter α in (3) until we find the exact number K.

Now we start to present a solution to (3). The following theorem shows that an optimal solution to (3) must contain optimal solutions to some subproblems.

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 **IEEE**

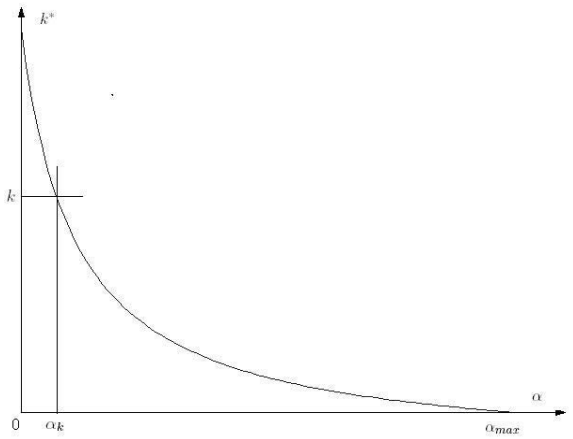

Figure 3. Relationship between k^* and α

Theorem 1 *Suppose that $\{v_1, v_2, \cdots, v_I\}$ is an optimal solution to the n-optimization problem in (3) and $\{u_1, u_2, \cdots, u_p\}$ is an optimal solution to the $(v_I - 1)$-optimization problem, then $\{u_1, u_2, \cdots, u_l, v_I\}$ is also an optimal solution to the n-optimization problem for given α.*

Proof By definition, it is obvious that the following inequality is correct.

$$\begin{aligned} & H(v_I - 1, \alpha : u_1, u_2, \cdots, u_l) \\ \geq\ & H(v_I - 1, \alpha : v_1, v_2, \cdots, v_{I-1}) \end{aligned}$$

Therefore, we have

$$\begin{aligned} & H(n, \alpha : u_1, u_2, \cdots, u_l, v_I) \\ =\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,u_1})} \\ & \left[(f_{A_{h,k},u_1} - f_{A_{h,k},u_2}) m(A_{h,k}, u_1) - \alpha\right] \\ +\ & \cdots \\ +\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,u_p})} \\ & \left[(f_{A_{h,k},u_p} - f_{A_{h,k},u_{p+1}}) \cdot m(A_{h,k}, u_p) - \alpha\right] \\ +\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_I})} \\ & \left[(f_{A_{h,k},v_I} - f_{A_{h,k},v_{I+1}}) \cdot m(A_{h,k}, v_I) - \alpha\right] \\ =\ & H(v_I - 1, \alpha : u_1, u_2, \cdots, u_l) \\ +\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_I})} \\ & \left[(f_{A_{h,k},v_I} - f_{A_{h,k},v_{I+1}}) \cdot m(A_{h,k}, v_I) - \alpha\right] \\ \geq\ & H(v_I - 1, \alpha : v_1, v_2, \cdots, v_{I-1}) \\ +\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_I})} \\ & \left[(f_{A_{h,k},v_I} - f_{A_{h,k},v_{I+1}}) \cdot m(A_{h,k}, v_I) - \alpha\right] \\ =\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_1})} \\ & \left[(f_{A_{h,k},v_1} - f_{A_{h,k},v_2}) \cdot m(A_{h,k}, v_1) - \alpha\right] \\ +\ & \cdots \\ =\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_{I-1}})} \\ & \left[(f_{A_{h,k},v_{I-1}} - f_{A_{h,k},v_I}) \cdot m(A_{h,k}, v_{I-1}) - \alpha\right] \\ +\ & \sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_I})} \\ & \left[(f_{A_{h,k},v_I} - f_{A_{h,k},v_{I+1}}) \cdot m(A_{h,k}, v_I) - \alpha\right] \\ =\ & H(n, \alpha : v_1, v_2, \cdots, v_{I-1}, v_I). \end{aligned}$$

On the other hand, since $\{v_1, v_2, \cdots, v_I\}$ is an optimal solution to the n-optimization problem, we have

$$\begin{aligned} & H(n, \alpha : u_1, u_2, \cdots, u_l, v_I) \\ \leq\ & H(n, \alpha : v_1, v_2, \cdots, v_{I-1}, v_I). \end{aligned}$$

So we have

$$\begin{aligned} & H(n, \alpha : u_1, u_2, \cdots, u_l, v_I) \\ =\ & H(n, \alpha : v_1, v_2, \cdots, v_{I-1}, v_I). \end{aligned}$$

Hence, the theorem is proven. #

Before presenting a dynamic programming-based algorithm for solving (3), we give the following definition.

Definition 5 *Define H_n^* to be the maximum aggregate cost gain of $H(n, \alpha : v_1, v_2, \cdots, v_k)$ obtained by solving the n-optimization problem and I_n the maximum index in the optimal solution. If the optimal solution is an empty set, define $I_n = -1$.*

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 **IEEE**

Obviously, we have $I_0 = -1$ and $H_0^* = 0$. From Theorem 1, we know that if $I_r > 0$,

$$\begin{aligned} &H_r = H_{I_r-1} \\ &+\sum_{h=1}^{l} \sum_{A_{h,k} \in D(B_{h,v_{I_r}})} \\ &((f_{A_{h,k},v_{I_r}} - f_{A_{h,k},v_{I_r+1}}) \cdot (c_{v^+_{I_r}(A_{h,k}),v_{I_r}} \\ &+w(B_{h,v^+_{I_r}(A_{h,k})}, A_{h,k}) - w(B_{h,v_{I_r}}, A_{h,k})) - \alpha) \end{aligned} \tag{4}$$

Therefore, we can check all possible locations of I_r $(0 \le r \le n)$ and select the one that maximizes $H(r : v_1, v_2, \cdots, v_k)$. So we have

$$\begin{cases} H_0^* = 0 \\ H_r^* = \max\limits_{1 \le v_i \le r} \{0, H^*_{v_i-1} + \sum\limits_{h=1}^{l} \sum\limits_{A_{h,k} \in D(B_{h,v_i})} \\ ((f_{A_{h,k},v_i} - f_{A_{h,k},v_{i+1}})(c_{v^+_i(A_{h,k}),v_i} \\ +w(B_{v^+_i(A_{h,k})}, A_{h,k}) - w(B_{v_i}, A_{h,k})) - \alpha)\} \end{cases} \tag{5}$$

and

$$\begin{cases} I_0 = -1 \\ I_r = \begin{cases} -1 \;\; if \;\; H_r^* = 0 \\ v \quad if \;\; H_r^* = H^*_{v-1} + \\ \quad \sum\limits_{A_{h,k} \in D(B_{h,v})} ((f_{A_{h,k},v} - f_{A_{h,k},v+1}) \\ (c_{v^+(A_{h,k}),v} + w(B_{v^+}(A_{h,k}), A_x) \\ -w(B_v, A_x)) - \alpha) \end{cases} \end{cases} \tag{6}$$

Based on Theorem 1 and the recurrences above, the problem as described in (3) can be solved using dynamic programming. After computing H_n^* and I_n, we can start from $v_r = I_n$ and obtain all the locations iteratively.

It is easy to see that the time complexity of the dynamic programming algorithm is $O(n^2lm)$, where n is the number of nodes, l is the number of the media objects, and m the maximum number of versions for all the media objects.

4 Simulation Model

In this section, we describe the simulation model used for performance analysis. We have performed extensive simulation experiments for comparing the results of our model with those of random transcoding proxy placement algorithm. The network in our simulation consists of numerous nodes and content servers. To the best of knowledge, it is difficult to find true trace data in the open literature to simulate our model. Therefore, we generated the simulation model from empirical results presented in [1, 2, 4].

Table 2 lists the parameters and their values used in our simulation.

Table 2. Parameters Used in Our Simulation

Parameter	Value
Number of WAN Nodes	200
Number of MAN Nodes	200
Delay of WAN Links	Exponential Distribution $p(x) = \theta^{-1}e^{-x/\theta}$ $(\theta = 0.45\ Sec)$
Delay of MAN Links	Exponential Distribution $p(x) = \theta^{-1}e^{-x/\theta}$ $(\theta = 0.06\ Sec)$
Number of Servers	100
Number of Web Objects	1000 objects per srever
Average Web Object Size	$26KB$
Object Access Frequency	Zipf-Like Distribution $\frac{1}{i^\alpha}$ $(i = 0.7)$
Request Rate Per Node	$U(1, 9)$ requests per second
Transcoding Rate	$20KB/Sec$

The network topology is randomly generated by the Tier program [4]. We have conducted experiments for a lot of topologies with different parameters and found that the performance of our model was insensitive to topology changes. Here, we list only the experimental results for one topology due to space limitations. Table 2 shows the characteristics of this topology and the workload model, which are chosen from the open literature and are considered to be reasonable.

The WAN (Wide Area Network) is viewed as the backbone network to which no servers or clients are attached. Each MAN (Metropolitan Area Network) node is assumed to connect to a content server. The number of objects generated is N and these N objects are divided into two type: *text* and *media*. We assume that for each media object it has five versions and the transcoding graph is as shown in Figure 4. The size 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

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 IEEE

quotient of the object size to the transcoding rate. In our experiments, 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 [2, 16]. Specifically, the probability of a request for an 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. Both the delay of MAN links and WAN links follow exponential distribution, where the average delay for WAN links is 0.46 seconds and the average delay for WAN links is 0.07 seconds.

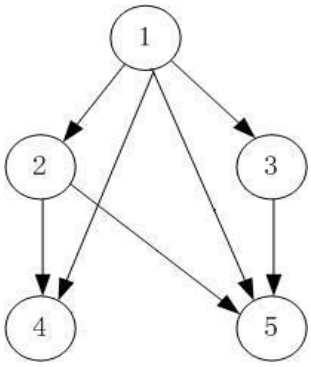

Figure 4. Transcoding Graph for Simulation

The total cost, including the cost for transferring request and the cost for transcoding, 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 propagation delay, the transmission delay, transcoding delay, and the searching 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.

5 Performance Results

In this section, we compare the performance of our model with that of random placement algorithm in terms of several performance metrics. The performance metrics employed in our simulation include delay-saving ratio (DSR) [1], average access latency (AST), request response ratio (RRR) [2], version hit ratio (VHR)[3], and content hit ratio (CHR)[4]. In the following figures, DPA denotes the results for our dynamic programming algorithm, RPA the results for the random placement algorithm.

In our experiments, we compare the performance of different models across a wide range of number of transcoding proxies placed.

The first experiment is to investigate DSR as a function of the number of transcoding proxies and Figure 5 shows the simulation results. As presented in Figure 5, we can see that our model outperforms the random placement algorithm since our model considers transcoding caching in a coordinated optimal way, whereas the random placement algorithm acts in a random way. Specifically, the mean improvement of DSR over RPA is about 22.9 percent.

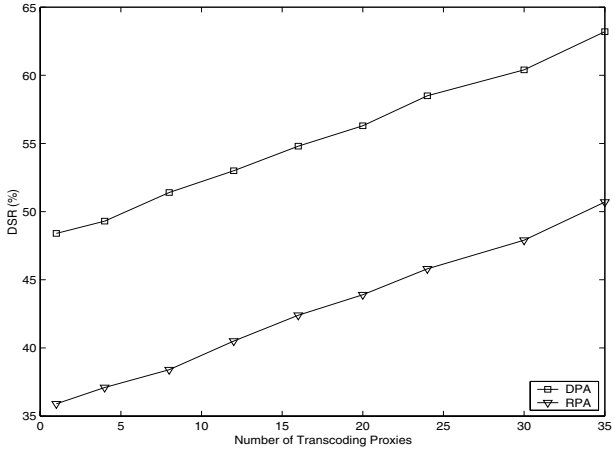

Figure 5. Experiment on DSR

Figure 6 shows the simulation results of ASL as a function of the number of transcoding proxies. We also describes the results of RRR as a function of the number of transcoding proxies in Figure 7. As we have known,

[1] DSR is define as the fraction of communication and server delays which is saved by satisfying the references from the proxy instead of the server

[2] RRR is defined as the ratio of its access latency to the size of the target object.

[3] VHR is defined as the ratio of the number of requests satisfied by the exact versions in the nearest higher level transcoding proxies as a whole to the total number of requests.

[4] CHR is defined as the ratio of the number of requests satisfied by the exact versions or the more detailed versions in the nearest higher level transcoding proxies as a whole to the total number of requests.

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 **IEEE**

the lower the ASL or the RRR, the better the performance. We can see that all models provide steady performance improvement as the number of transcoding proxies increases. We can also see that our dynamic programming algorithm significantly improves both ASL and RRR compared to the random placement algorithm. This is because our model determines the optimal locations in a coordinated way, while the random placement algorithm places transcoding proxies randomly. The average improvements of ASL and RRR over RPA are about 82.1 percent and 127 percent, respectively.

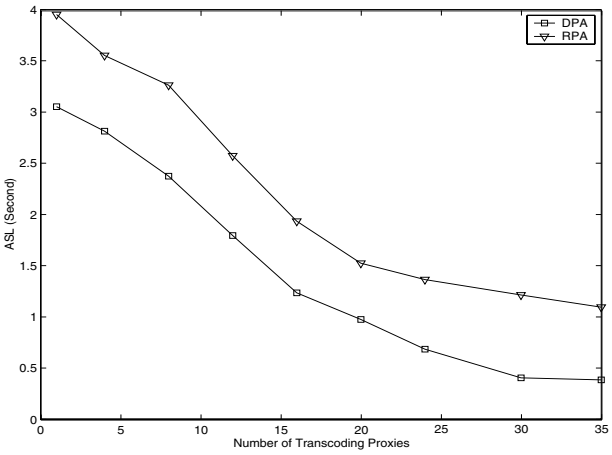

Figure 6. Experiment on ASL

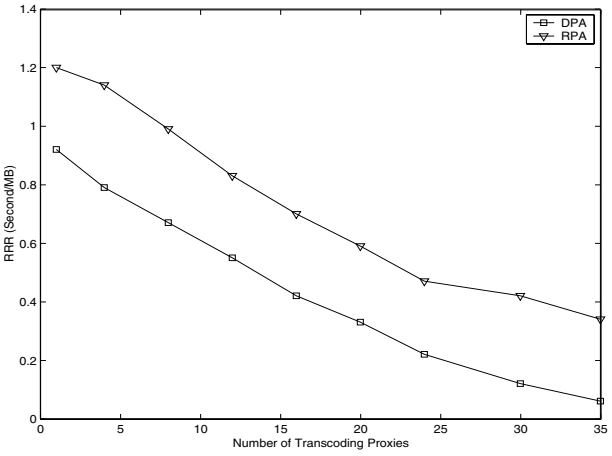

Figure 7. Experiment on RRR

Figure 8 and 9 show the simulation results of VHR and CHR as functions of number of transcoding proxies respectively. As we have known, the higher the VHR or the CHR, the better the performance. As we can see, all models provide steady performance improvement as the number of transcoding proxies increases. We can also see that our dynamic programming algorithm significantly improves both VHR and CHR compared to the random placement algorithm, since our model determines the optimal locations in a coordinated way, while the random placement algorithm places transcoding proxies randomly. Specifically, the mean improvements of VHR and CHR over RPA are about 22.1 percent and 20.1 percent, respectively.

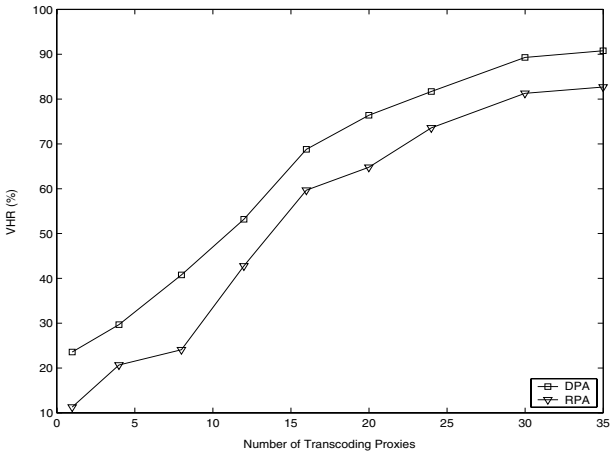

Figure 8. Experiment on VHR

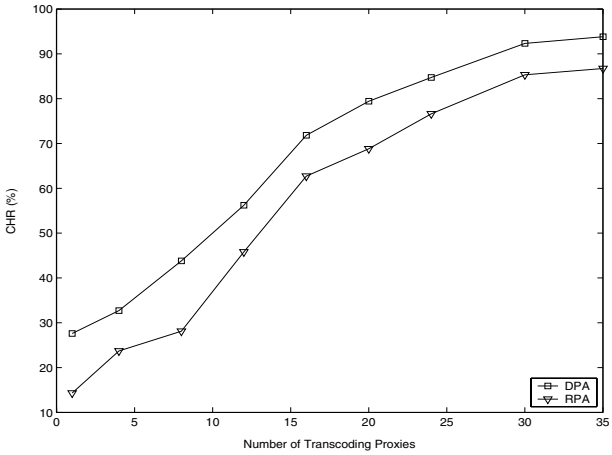

Figure 9. Experiment on CHR

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 **IEEE**

6 Conclusion

Transcoding proxy placement is one of the most important issues in transcoding caching. In this paper, we studied the transcoding proxy placement problem and presented a novel mathematical model for this problem. We also proposed low-cost dynamic programming-based algorithms by which the optimal locations for placing fixed number of transcoding proxies among the en-route nodes in a network can be obtained. We have performed extensive simulation experiments to compare the proposed model with those of the random placement model. The results show that our model significantly outperforms random placement model.

References

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

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

[3] E. A. Brewer, R. H. Katz, E. Amir, H. Balakrishnan, Y. Chawathe, A. Fox, S. D. Gribble, T. Hodes, G. Nguyen, V. N. Padmanabhan, M. Stemm, S. Seshan, and T. Henderson. *A Network Architecture for Hetergeneous Mobile Computing*. IEEE Personal Comm., Vol. 5, No. 5, pp. 8-24, Oct., 1998.

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

[5] V. Cardellini, P. Yu, and Y. Huang. *Collaborative Proxy System for Distributed Web Content Transcoding*. Proc. ACM Int'l Conf. Information and Knowledge Management, pp. 520-527, 2000.

[6] C. Chandra and C. S. Ellis. *JPEG Compression Metric as a Quality-Aware Image Transcoding*. Proc. USENIX Second Symposium Intenet Technology and Systems, pp. 81-92, 1999.

[7] C. Chang and M. Chen. *On Exploring Aggregate Effect for Efficient Cache Replacement in Trancoding Proxies*. IEEE Transactions on Parallel and Distributed Systems, Vol. 14, No. 6, pp. 611-624, June, 2003.

[8] M. D. Dahlin, R. Y. Wang, T. E. Anderson, and D. A. Patterson. *Cooperative Caching: Using Remote Client Memory to Improve File System Performance*. Proc. First Symp. Operating Systems Design and Implementations, pp. 267-280, 1994.

[9] R. Floyd and B. Housel. *Mobile Web Access Using Network Web Express*. IEEE Personal Comm., Vol. 5, No. 5, pp. 47-52, Dec., 1998.

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

[11] M. R. Korupolu and M. Dahlin. *Coordinated Placement and Replacement for Large-Scale Distributed Caches*. IEEE Transaction on Knowledge and Data Engineering, Vol. 14, No. 6, pp. 1317-1329, 2002.

[12] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. *Placement Algorithms for Hierarchical Coorperative Caching*. Proc. 10th Ann. ACM-SIAM Symp. Discrete Algorithms, pp. 586-595, 1999.

[13] P. Krishnan, D. Raz, and Y. Shavitt. *The Cache Location Problem*. IEEE/ACM Transaction on Networking, Vol. 8, No. 5, pp. 568-582, 2000.

[14] B. Li, X. Deng, M. J. Golin, and K. Sohraby. *On the Optimal Placement of Web Proxies in the Internet: The Linear Topology*. Proc. Eighth IFIP TC-6 Int'l Conf. High Performance Netwroking (HPN) pp. 485-495, 1998.

[15] R. Mohan, J. R. Smith and C. Li. *Adapting Multimedia Internet Content for Univeral Access*. IEEE Transaction on Multimedia, Vol. 1, No. 1, pp. 104-114, March, 1999.

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

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 **IEEE**

[17] P. Rodriguez, C. Spanner, and E. W. Biersack. *Analysis of Web Caching Architectures: Hierarchical and Distributed Caching*. IEEE/ACM Transaction on Networking, Vol. 9, No. 4, pp. 404-418, 2001.

[18] J. Shim, P. Scheuermann, and R. Vingralek. *Proxy Cache Algorithms: Design, Implementation, and Performance*. IEEE Transaction on Knowledge and Data Engineering, Vol 11, No. 4, pp, 549-562, 1999.

[19] X. Tang and S. T. Chanson. *Coordinated En-Route Web Caching*. IEEE Transactions on Computers, Vol. 51, No. 6, pp. 595-607, June, 2002.

[20] X. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay. *Design Considerations for Distributed Caching on the Internet*. Proc. 19th Int'l Confernece Distributed Computing Systems (ICDCS), pp. 273-284, 1999.

Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)
0-7695-2096-0/04 $20.00 © 2004 **IEEE**