Coordinated En-Route Transcoding Caching for Tree Networks

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

Abstract

As transcoding caching is attracting an increasing amount of attention, it is important and necessary to find methods to distribute multiple versions of the same media object in the Internet. In this paper, we first present a mathematical model for the problem of optimally determining the locations in which to place multiple versions of the same media object in tree networks such that the specified objective is achieved. This problem is formulated as an optimization problem. Second, we propose a low-cost dynamic programming-based solution for solving this problem, by which the optimal locations are obtained. Finally, we evaluate our model on different performance metrics through extensive simulation experiments and compare the results of our model with those of existing models that consider transcoding caching either on a path or at individual nodes only.

Key words: Transcoding caching, Internet, tree network, dynamic programming, optimization problem.

1. Introduction

Transcoding is a transformation that is used to convert a media object from one form to another, frequently trading off object fidelity for size. As audio and video applications have proliferated on the internet, it is definitely important to find methods to distribute multiple versions of the same media object in the Internet, which is defined as transcoding caching in this paper. Transcoding caching is attracting more and more attention since it plays an important role in the functionality of web caching [4, 12, 18]. En-route caching is a recently developed caching architecture [14, 22] in which caches are placed on the access path from the user to the server. Each en-route cache intercepts any request that passes through its associated node, and either satisfies the request by sending the media object to the client or forwards the request upstream along the path to the server until it can be satisfied. Transcoding can be executed by various components in the network such as server, proxy, and client. In the case of the client, it can preserve the original semantics of system architecture and transport protocols. However, this solution is extremely expensive when the clients are mobile users, due to connection bandwidth and power limitations. In the case of the server, it is not necessary to perform transcoding during the time between the client issuing a request and the server's response 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 on the server. Further, this method is not flexible in dealing with changing clients' needs. For these reasons, it will be better to transcode the media objects in intermediate proxies. Much research has been focused on exploring the advantages of this approach [8, 11, 12], in which an intermediate proxy is capable of transcoding the requested media object to a proper version according to the client's specification before sending the media object to the client. In this paper we refer to the cache attached to an intermediate proxy as a transcoding cache. Cooperative caching, in which caches cooperate to fulfill each other's requests and make storage decisions, is a powerful paradigm to improve cache effectiveness [10, 15, 16].

Existing caching schemes for transcoding caching consider distributing multiple versions of the same media object either on a path or at each node. In this paper, we address the problem of optimally deciding the locations in which to store multiple versions of a media object among the en-route caches for coordinated en-route transcoding caching in tree networks such that the specified objective is achieved. To solve this problem, we present an original model, which makes caching decisions on all the en-route caches along the routing path in a coordinated way. In our model, cache status information along the routing path of a request is used to optimally determine the locations for caching multiple versions of the same media object. We formulate this as an optimization problem and a low-cost dynamic programming-based solution is developed to obtain the optimal locations. We also extend this solution to solve the problem of optimally determining the locations in

Proceedings of the Tenth International Conference on Parallel and Distributed Systems (ICPADS'04)
1521-9097/04 $ 20.00 IEEE

which to place a fixed number of multiple versions of the same media object in tree networks. We implement our algorithms and evaluate our model on various performance metrics through extensive simulation experiments. The implementation results show that our model significantly outperforms existing models that consider transcoding caching either on a path or only at individual nodes.

This paper is organized as follows. We generalize the mathematical model in Section 2. Section 3 presents and discusses the dynamic programming-based solution. The simulation model and the performance evaluation are described in Section 4. Section 5 concludes the paper.

2. Mathematical Model

The network we use in this paper is modelled as a tree $T = (V, E)$, where $V = (v_1, v_2, \cdots, v_n)$ is the set of nodes, and E is the set of network links. We denote the set of all nodes that are the children of node v as $C(v)$, the set of all nodes that are the descendants of node v as $D(v)$. $B(v)$ expresses the set of all branches of a tree rooted from node v. Without loss of generality, we assume that there is only one content server at the root by which all the media objects are maintained. In our analysis, we assume that each node is associated with an en-route cache. Our analysis can be easily extended to the case in which caches are associated with certain subset of nodes by only including the nodes with caches in the graph. A client's request for a media object goes along the path from the client to the server until it is satisfied by the first node on the path whose cache stores a more detailed version of that object. After transcoding if necessary, the requested version will be sent back to the client along the same path. We also assume in this paper that the routing path is symmetric. For the asymmetric case, we can consider a subset of V by excluding those noses which are not on both upstream and downstream paths. Such a simplification is validated in [22]. All the routing paths from the clients form a tree topology [14, 22]. Figure 1 shows an example of such a tree topology.

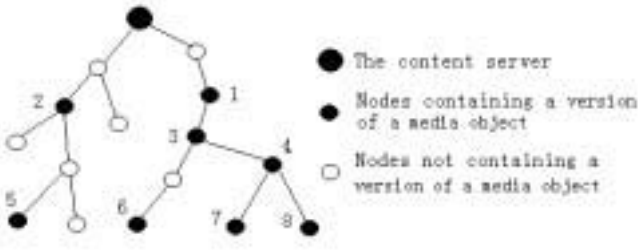

Figure 1. Coordinated En-Route Transcoding Caching

Let $A = (A_1, A_2, \cdots, A_m)$ be the set of all the versions of a media object. b_{A_j} is the size of A_j. We assume that the access frequencies for A_j from v_i, denoted by f_{A_j,v_i}, 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 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. The relationship among different versions of a media object can be expressed by a weighted transcoding graph [9], which can be viewed as an extension to the transcoding relation graph [7]. An example of a weighted transcoding graph is shown in Figure 2, 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 when 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. In our example, transcoding can not be executed between A_4 and A_5 due to insufficient content information. The number beside each edge is the transcoding cost from one version to another. The transcoding cost of a media object from A_i to A_j is given by the weight on the edge (A_i, A_j), which is denoted by $w(A_i, A_j)$. $\phi(A_i)$ is the set of all versions that can be transcoded from A_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.

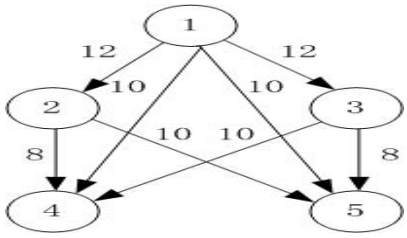

Figure 2. A Weighted Transcoding Graph

A list of symbols used in this paper is given in Table 1. Our mathematical model is formulated based on these symbols.

The media object placement is trivial if the cache sizes are infinite, in which case, all the versions of a media object can be stored in every cache; thus, the total access cost is minimized. Due to this reason, when a new version of a media object is inserted into a cache, one or more objects may need to be removed from the cache to make room for it. Storing a media object at a node enables all the requests previously passing it now to be satisfied at it (Transcoding

Symbol	Decription
$A = (A_1, A_2, \cdots, A_m)$	the set of versions of a media object
s_{A_i}	the size of A_i
$V = (v_1, v_2, \cdots, v_n)$	the set of nodes in a network
$v^+(A_i)$	the nearest higher level node of v that stores a more detailed version than A_i
$z_b^-(A_i)$	the nearest lower level node of v that stores a less detailed version than A_i in branch b
H_v	the version cached or to be cached at node v
$f_{A_j,v}$	the mean access frequency of A_j from v
c_{v_i,v_j}	the cost of transmitting a media object between v_i and v_j
$w(A_i, A_j)$	the transcoding cost from A_i to A_j for a media object
$\phi(A_i)$	the set of all versions that can be transcoded from A_i
$B(v)$	the set of all branches of a tree whose root is node v
$C(v)$	the set of all nodes that are the children of node v
$D(v)$	the set of all nodes that are the descendants of node v

Table 1. A List of the Symbols

may be necessary); hence, the total cost, including transmission and transcoding costs, which is defined in this paper as *cost saving*, is decreased. Similarly, removing the copy of an object from a node increases its access cost, which is defined as *cost loss*. The media object placement problem for coordinated en-route transcoding caching is further complicated due to caching dependencies, e.g. a placement decision at one node in the network affects the performance gain of caching the same media object at other nodes. The optimal locations for caching multiple versions of a media object depend on the cost savings and the cost losses at all the nodes along the routing path. Our objective is to minimize the total access cost of all the media objects in the network. We begin with computing the cost saving and the cost loss of caching a media object at a single node. Let $m(A_i, v_j)$ be the miss penalty of version A_i with respect to node v_j, which is defined as the additional cost of accessing A_i if the version cached at node v_j is removed. In our model, $m(A_i, v_j)$ is given by the following definition.

Definition 1 *$m(A_i, v_j)$ is a function for calculating the miss penalty of A_i if the version cached at node v_j is removed, which is defined as*

$$m(A_i, v_j) = c_{v_j, v_j^+(A_i)} + w(B_{v_j^+(A_i)}, A_i) - w(B_{v_j}, A_i) \quad (1)$$

where $v_j^+(A_i)$ is the nearest higher level node of v_j that stores a more detailed version than A_i (including A_i), $c_{v_j, v_j^+(A_i)}$ is the additional access cost, $w(B_{v_j^+(A_i)}, A_i)$ is the new transcoding cost , and $w(B_{v_j}, A_i)$ is the original transcoding cost.

Obviously, the cost saving of caching A_i at v_j is defined as follows.

Definition 2 *$s(A_i, v_j)$ is a function for calculating the cost saving of caching A_i at v_j.*

$$s(A_i, v_j) = \sum_{A_k \in D(A_i) \cup \{A_i\}} f_{A_k, v_j} \cdot m(A_k, v_j) \quad (2)$$

Second, we consider the cost loss of caching a media object at a node. Let $l(A_i, v_j)$ denote the cost loss of caching A_i at v_j. Computing $l(A_i, v_j)$ is a bit more complicated. Obviously, the purged objects should introduce the least total cost loss while creating enough space to accommodate the object to be cached. We apply the following greedy heuristic to decide replacement candidates. Note that the normalized cost loss (NCL, i.e., the cost loss introduced by creating one unit of free space) of ejecting A_i is $\frac{s(A_i, v_j)}{b_{A_i}}$. The objects in the cache are ordered by their NCLs and are selected sequentially, starting from the object with the smallest NCL, until enough space is created. The cost loss of caching a media object at a node is calculated by summing the cost losses caused by all the selected candidates. Therefore, the cost gain of caching A_i at node v_j, denoted by $g(A_i, v_j)$, is calculated by the following equation.

$$g(A_i, v_j) = \sum_{A_k \in D(A_i) \cup \{A_i\}} f_{A_k, v_j} \cdot m(A_k, v_j) - l(A_i, v_j) \quad (3)$$

Based on the cost gain of caching a version of a media object at a single node, the problem of coordinated en-route web caching in transcoding proxies is defined as follows:

Definition 3 *$G(T, P)$, the total cost gain of caching multiple versions of a media object at nodes in $P \subseteq V$ is defined*

as

$$G(T_v,P)=\sum_{v\in P}\sum_{A_x\in\Phi(H_v)}(f_{A_x,v}-\sum_{b\in B(v)}f_{A_x,z_b^-(A_x)})m(A_x,v_j)-l(A_x,v) \quad (4)$$

where $v^+(A_x)$ is the nearest higher level node of v that stores a more detailed version than A_x (including A_x), $z_b^-(A_x)$ is the nearest lower level node of v that stores a less detailed version than A_x (including A_x) in branch b, $H_{v_j^+(A_x)}$ is the version cached at $v_j^+(A_x)$, and H_v is the version to be cached at node v.

3. Dynamic Programming-Based Solution

In this paper we use $T_r=(V_r,E_r)$ to denote a tree whose root is r, where V_r and E_r are the sets of nodes and network links of tree T_r, respectively. Based Definition 3, the problem of coordinated en-route transcoding caching for tree T_r is formally defined as an optimization problem as follows:

$$\max_{P_r}G(T_r,P_r)=\max_{P_r}\{\sum_{v\in P_r}\sum_{A_x\in\Phi(H_v)}(f_{A_x,v}-\sum_{b\in B(v)}f_{A_x,z_b^-(A_x)})m(A_x,v)-l(A_x,v)\} \quad (5)$$

where $P_r\subseteq D(r)\cup\{r\}$, $v^+(A_x)$ is the nearest higher level node of v that stores a more detailed version than A_x (including A_x), $z_b^-(A_x)$ is the nearest lower level node of v that stores a less detailed version than A_x (including A_x) in branch b, $H_{v^+(A_x)}$ is the version cached at $v^+(A_x)$, and H_v is the version to be cached at node v.

Before presenting the dynamic programming-based solution, we give the following definition. Let $T_{r,w}$ be a subtree of T_r, whose node set is V_w, where $w\in D(r)$. Similarly, we define the problem of coordinated en-route transcoding caching for tree $T_{r,w}$ as an optimization problem as follows:

$$\max_{P_{r,w}}G(T_{r,w},P_{r,w})=\max_{P_{r,w}}\{\sum_{v\in P_{r,w}}\sum_{A_x\in\Phi(H_v)}(f_{A_x,v}-\sum_{b\in B(v)}f_{A_x,v_b^-(A_x)})m(A_x,v)-l(A_x,v)\} \quad (6)$$

where $P_{r,w}\subseteq\{w\}\cup D(w)$, $v^+(A_x)$ is the nearest higher level node of v that stores a more detailed version than A_x (including A_x), $z_b^-(A_x)$ is the nearest lower level node of v that stores a less detailed version than A_x (including A_x) in branch b, $H_{v^+(A_k)}$ is the version cached at $v^+(A_k)$, and H_v is the version to be cached at node v.

Now we start to present a dynamic programming-based solution to the optimization problem formulated in Equation (5). First, we give a theorem that indicates an important property between the optimal solutions to tree T_r and T_{r,r_i}, where $r_i\in C(r)$. Due to space limitation, we do not give the detailed proof of the theorems in this paper.

Theorem 1 *For tree T_r, if $C(r)=\{r_1,r_2,\cdots,r_s\}$, then we have*

$$A_r^*=\cup_{i=1}^s A_{r,r_i}^*$$

where $A_r^*\subseteq D(r)$ is an optimal solution to Equation (5) with respect to tree T_r, and $A_{r,r_i}^*\subseteq D(r_i)\cup\{r_i\}$ is an optimal solution to Equation (6) with respect to tree T_{r,r_i}, $i=1,2,\cdots,s$.

Theorem 1 shows that computing the optimal solution to Equation (5) for tree T_r can be decomposed into computing the optimal solutions to trees T_{r,r_i}, where $C(r)=\{r_1,r_2,\cdots,r_s\}$. Therefore, what we should do is to to find an optimal solution to Equation (6).

Next, we present another theorem that describes an important property of the solution to Equation (6), by which an optimal solution to Equation (6) can be obtained.

Theorem 2 *For tree $T_{r,w}$, if $C(w)=\{w_1,w_2,\cdots,w_t\}$, then we have*

$$A_{r,w}^*=\begin{cases}\cup_{i=1}^t A_{r,w_i}^* & G(T_{r,w},\cup_{i=1}^t A_{r,w_i}^*)\geq G(T_{r,w},A_w^*\cup\{w\})\\ A_r^*\cup\{w\} & G(T_{r,w},\cup_{i=1}^t A_{r,w_i}^*)<G(T_{r,w},A_w^*\cup\{w\})\end{cases}$$

where $A_{r,w}^*\subseteq D(w)\cup\{w\}$ is an optimal solution to Equation (6) with respect to tree $T_{r,w}$, $A_w^*\subseteq D(w)$ is an optimal solution to Equation (5) with respect to tree T_w, and $A_{r,w_i}^*\subseteq D(w_i)\cup\{w_i\}$ is an optimal solution to Equation (6) with respect to tree T_{r,w_i}, $i=1,2,\cdots,t$.

Based on Theorems 1 and 2, the original problem can be solved using dynamic programming with the following recurrences.

1. Suppose a node v in tree $T_r=(V_r,E_r)$ has t children. If $t=0$, then $A_v^*=\phi$; otherwise, $A_v^*=\cup_{i=1}^t A_{v,v_i}^*$, where $v_1,v_2,\cdots,v_t$ are the children of v.

2. Suppose that node u is the descendent of node v in tree $T_r=(V_r,E_r)$ and u has l children $u_1,u_2,\cdots,u_l$. If $l=0$, then

$$A_{v,u}^*=\begin{cases}\{u\} & G(P_{v,u},v)\geq 0\\ \{\phi\} & G(P_{v,u},v)<0\end{cases}$$

where $G(P_{v,u})$ is the maximal cost gain on the path from u to v and a version of a media object is stored at v; otherwise,

$$A_{v,u}^*=\begin{cases}\cup_{i=1}^l A_{v,u_i}^* & G(T_{v,u},\cup_{i=1}^l A_{v,u_i}^*)\geq G(T_{v,u},A_u^*\cup\{u\})\\ A_u^*\cup\{u\} & G(T_{v,u},\cup_{i=1}^l A_{v,u_i}^*)<G(T_{v,u},A_u^*\cup\{u\})\end{cases}$$

Now we start to discuss the solution above. The following theorem describes an important property of the algorithm proposed above.

Theorem 3 *If A_r^* is an optimal solution to Equation (5) with respect to tree T_r, then we have*

$$\sum_{i=1}^{m} f_{A_i,v} \cdot c_{v,r} - w(H_v, A_i) - l(A_i, v) \geq 0 \quad \forall \ v \in A_r^*$$

Theorem 3 shows that we should only consider the nodes where the local cost gain is beneficial, i.e. the cost saving outweighs the cost loss with respect to that single node. It can be shown that the complexity of this dynamic programming-based algorithm is $O(n^2 m)$, where n is the total number of nodes that are locally beneficial in the network and m is the number of the versions that a media object owns.

4. Simulation Model and Performance Evaluation

We have performed extensive simulation experiments to compare the results of our model with those of existing caching models. The network in our simulation consists of numerous nodes and content servers. 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 [1–3, 5, 9]. The system configuration is outlined in section 4.1, and four existing caching models used for the purpose of comparison are introduced in Section 4.2.

4.1. System Configuration

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

The network topology was randomly generated by the Tier program [5]. Experiments for many topologies with different parameters have been conducted and it was found that the performance of our model was 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 is shown in Table 2, 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. Each MAN and WAN node is associated with an en-route cache. The number of objects generated is N and these N objects are divided into two types: *text* and *media*. Similar to the studies in [3, 6, 13, 21, 22], 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 $26KB$. We also assume that each media object has five versions and that the transcoding graph is as shown in Figure 3. 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 MAN node randomly generates the requests, and the average request rate of each node follows the distribution of $U(1, 9)$, where $U(x, y)$ represents a uniform distribution between x and y. The access frequencies of both the content servers and the objects maintained by a given server follow a Zipf-like distribution [3, 19]. 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 0.46 seconds and the average delay for WAN links is 0.07 seconds.

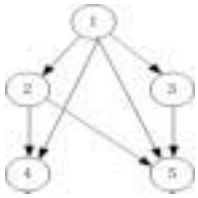

Figure 3. Transcoding Graph for Simulation

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.

4.2. Existing Caching Models

In addition to the model presented in Section 2, we also consider the following caching models for comparison purposes.

- LRU: Least Recently Used (LRU) evicts the web object which is requested the least recently. The requested object is stored at each node through which the object passes. The cache purges one or more least recently requested objects to accommodate the new object if there is not enough room for it.
- $LNC - R$ *[20]*: Least Normalized Cost Replacement ($LNC - R$) is an model that approximates the opti-

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
Web Object Size Distribution	Pareto Distribution $p(x) = \frac{ab^a}{a-1}$ $(a = 1.1, b = 8596)$
Web Object Access Frequency	Zipf-Like Distribution $\frac{1}{i^\alpha}$ $(i = 0.7)$
Relative Cache Size Per Node	4%
Average Request Rate Per Node	$U(1, 9)$ requests per second
Transcoding Rate	$20KB/Sec$

Table 2. Parameters Used in Our Simulation

mal cache replacement model. It selects for replacement the least profitable documents. The profit function is defined as $profit(O_i) = (c_i \cdot f_i)/s_i$, where c_i is the average delay to fetch document O_i to the cache, f_i is the total number of references to O_i, and s_i is the size of document O_i. Similar to LRU, the requested object is cached by all nodes along the routing path.

- *AE [9]*: Aggregate Effect (AE) is a model that explores the aggregate effect of caching multiple versions of an object in the cache. It formulates a generalized profit function to evaluate the aggregate profit from caching multiple versions of the same media object. When the requested object passes through each node, the cache will determine which version of that object should be stored at that node according to the generalized profit.
- *$TCLT$ [17]*: Transcoding Caching for Linear Topology ($TCLT$) is an algorithm that optimizes caching multiple versions of an object on the path from the client to the server.

4.3. Performance Evaluation

We compare the performance results of our model with that of those models introduced above in terms of several performance metrics. The performance metrics employed 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 (AST), request response ratio (RRR),which is defined as the ratio of the access latency of the target object to its size, object hit ratio (OHR), which is defined as the ratio of the number of requests satisfied by 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, $LNC-R$, AE, and $TCLT$ denote the results for the four models introduced in Section 4.2, and $TCTN$ shows the results for the model of coordinated en-route transcoding caching for tree networks proposed in Section 2. Table 3 lists the notations used in this section.

In our experiments, we compare the performance results of different models 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 Figure 4 shows the simulation results. As presented in Figure 4, we can see that our model outperforms the other models since our model consider en-route transcoding caching by optimally determining the locations in which to place multiple versions of a media object in a coordinated way, whereas existing models, including LRU, $LNC-R$, and AE, consider en-route web caching either on a path or only at a single node. Specifically, the mean improvements of DSR over $TCLT$, AE, $LNC-R$, LRU are 4.3 percent, 21.3 percent, 29.7 percent, and 31.7 percent, respectively.

Figure 5 shows the simulation results of ASL as a function of the relative cache size at each node; we describe the

Item	Notation	Decription
Performance Metrics	DSR	Delay-Saving Ratio (%)
	ASL	Average Access Latency (Second)
	RRR	Request Response Ratio (Second/MB)
	OHR	Object Hit Ratio (%)
	HSL	Highest Server Load (MB/Second)
Performance Results	$TCTN$	Transcoding Caching for Tree Networks
	$TCLT$	Transcoding Caching for Linear Topology
	AE	Standing for Aggregate Effect
	$LNC-R$	Least Normalized Cost Replacement
	LRU	Least Recently Used

Table 3. Notations Used in Performance Analysis

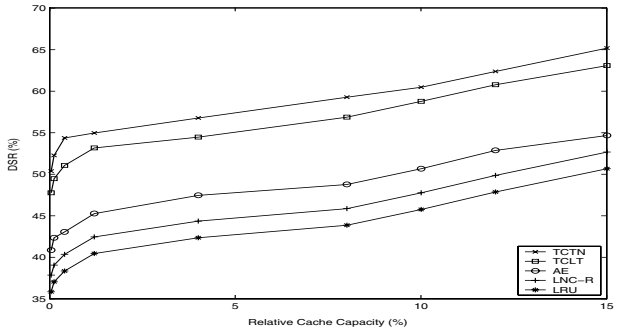

Figure 4. Experiment on DSR

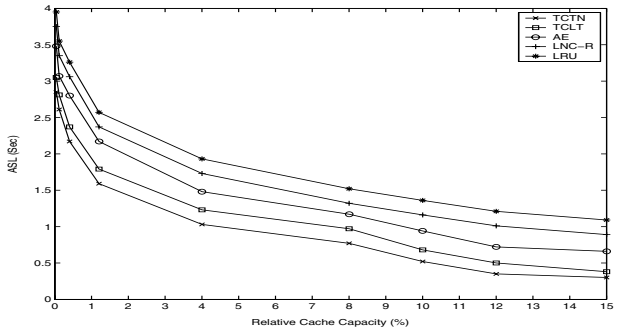

Figure 5. Experiment on ASL

results of RRR as a function of the relative cache size at each node in Figure 6. Clearly, the lower the ASL or the RRR, the better the performance. As we can see, all models provide steady performance improvement as the cache size increases. We can also see that $TCTN$ significantly improves both ASL and RRR compared to $TCLT$, AE, $LNC-R$ and LRU, since our model determines the optimal locations for tree networks in a coordinated way, while the others place multiple versions of a media object for linear topology or at each en-route cache. For ASL to achieve the same performance as $TCTN$, the other models need 2 to 12 times as much cache size.

Figure 7 shows the results of OHR as a function of the relative cache size for different models. By computing the optimal locations, we can see that the results for our model can greatly outperform those of the other models, 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.

Figure 8 shows the results of HSL as a function of the relative cache size. It can be seen that HSL for our model is lower than that of the other models. We can also see that HSL decreases as the relative cache size increases.

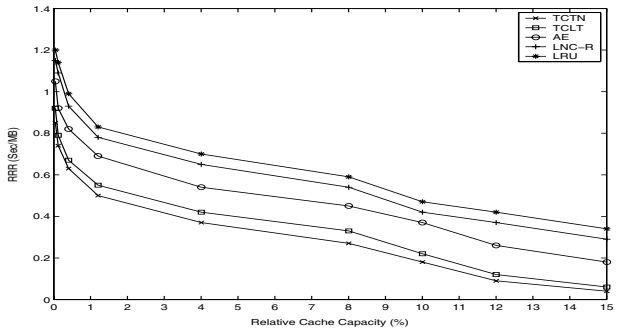

Figure 6. Experiment on RRR

5. Conclusion

In this paper, we studied the problem of optimally determining the locations in which to place multiple versions of the same media object in tree networks. We presented a novel model for this problem and formulated this problem as an optimization problem. The implementation results show that our model can significantly outperform existing models that consider transcoding caching on a path or at individual nodes. Our methods make significant contribu-

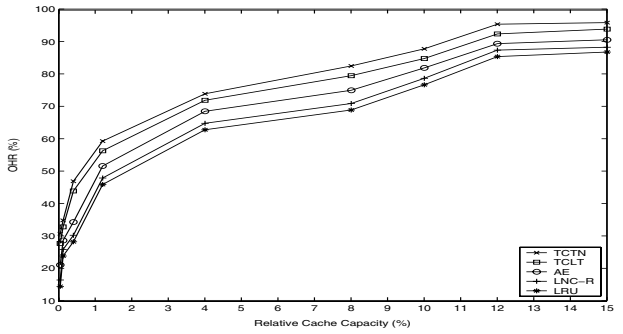

Figure 7. Experiment for OHR

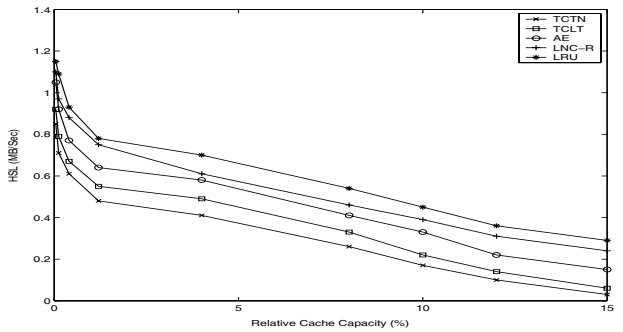

Figure 8. Experiment for ASL

tions to transcoding caching, since the locations for placing copies of an object among the en-route caches in tree networks can be optimally obtained.

References

[1] C. Aggarwal, J. L. Wolf, and P. S. Yu. *Caching on the World Wide Web*. IEEE Transaction on Knowledge and Data Engineering, Vol 11, No. 1, pp, 94-107, 1999.

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

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

[4] 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 Heterogeneous Mobile Computing*. IEEE Personal Comm., Vol. 5, No. 5, pp. 8-24, Oct., 1998.

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

[6] P. Cao and S. Irani. *Cost-Aware WWW Proxy Caching Algorithms*. Proc. First USENIX Symp. Internet Technologies and Systems (USITS), pp. 193-206, 1997.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] S. Jin and A. Bestavros. *Greeddual* Web Caching Algorithm Exploiting the Two Sources of Temporal Locality in Web Request Streams*. Computer Comm., Vol. 4, No. 2, pp. 174-183, 2001.

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

[15] 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.

[16] 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.

[17] K. Li and H. Shen. *Coordinated En-Route Web Caching in Transcoding Proxies*. Submitted for Publication.

[18] 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.

[19] 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.

[20] P. Scheuermann, J. Shim, and R. Vingralek. *A Case for Delay-Conscious Caching of Web Documents*. Computer Network and ISDN Systems, Vol 29, No. 8-13, pp, 997-1005, 1997.

[21] 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.

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