PAPER Special Section on New Challenge for Internet Technology and its Architecture

Selective Update Approach to Maintain Strong Web Consistency in Dynamic Content Delivery

Zhou SU[†a)], ***Member***, **Masato OGURO**[††], ***Student Member***, **Jiro KATTO**[†], ***Member***, ***and*** **Yasuhiko YASUDA**[†], ***Fellow, Honorary Member***

SUMMARY Content delivery network improves end-user performance by replicating Web contents on a group of geographically distributed sites interconnected over the Internet. However, with the development whereby content distribution systems can manage dynamically changing files, an important issue to be resolved is consistency management, which means the cached replicas on different sites must be updated if the originals change. In this paper, based on the analytical formulation of object freshness, web access distribution and network topology, we derive a novel algorithm as follows: (1) For a given content which has been changed on its original server, only a limited number of its replicas instead of all replicas are updated. (2) After a replica has been selected for update, the latest version will be sent from an algorithm-decided site instead of from its original server. Simulation results verify that the proposed algorithm provides better consistency management than conventional methods with the reduced the old hit ratio and network traffic.

key words: *content delivery networks, consistency algorithm, web cache performance, network traffic*

1. Introduction

With the growth in popularity of the Internet and the wide availability of streaming applications, how to efficiently distribute the stored content has become a major concern in the Internet community.

Some content delivery networks (CDN) [3], [4] have emerged, and they work directly with content providers to cache and replicate the providers' content close to the end users by using geographically distributed edge servers. More recently, some other researchers have also advocated using a CDN structure composed of dedicated transit nodes to distribute the large contents [19].

Although CDNs facilitate static file sharing, newly-developed applications, such as online auction and remote collaboration, demand that they should be able to manage dynamically-changing files. There has been some research [6], [9], [12], [16], [18] on this problem, which is called consistency management. However, most of these studies treat different replicas of the same content to be managed for Web consistency in the same manner. Furthermore, how to optimally select a surrogate instead of an original server to update the content has not been discussed.

Manuscript received January 25, 2007.
Manuscript revised April 23, 2007.

[†]The authors are with the School of Science and Engineering, Waseda University, Tokyo, 169-8555 Japan.
[††]The author is with Nomura Research Institute, Ltd., Tokyo, 100-0005 Japan.
a) E-mail: zhousu@asagi.waseda.jp

DOI: 10.1093/ietcom/e90–b.10.2729

In this paper, we therefore propose an optimal algorithm for controlling Web consistency in content delivery. Firstly, we carry out a theoretical analysis of the Web access and the freshness time of objects. Based on the analytical result, we then propose a *consistency priority* and assign different priorities to different replicas of the same content. When a given content is changed on its original server, instead of all its replicas over the whole network, only its replicas with high *consistency priorities* will be updated.

Secondly, if one replica of a given content is selected to be updated, the latest version of this content will be sent from a surrogate with the lowest *update priority*, which is proposed based on the network topology and bandwidth. Therefore, the latest version will be sent from an algorithm-decided site instead of from its original server to reduce the network traffic.

Finally, through simulations we check the performance of our proposal when the related parameters are changed, and find that our proposal can efficiently improve the hit ratio and network traffic against the previous algorithms. We also show that the necessary parameters in our proposed algorithm can be obtained from the information readily available in the local system.

This paper is organized as follows: in Sect. 2, an overview of the CDN system is provided. In Sect. 3, related work with regard to consistency management algorithms is reviewed. Section 4 presents mathematical analyses of Web access and user delay. And our proposed algorithm is also presented. In Sect. 5, extensive simulation results are given and conclusions are presented in Sect. 6.

2. Content Delivery

2.1 Content Delivery Overview

How to efficiently distribute the Web content has attracted much research. Content delivery networks (CDNs) appeared recently and are deploying quite rapidly [1]–[4], [10], [13], [14], [17], [29]. Their concern is mainly placed on efficient delivery of static content, i.e. HTML files and images. Some CDN companies advocate their support for the dynamically changed content, but their technical details are not yet clarified nor verified.

In Peer to Peer (P2P) networks, users can determine from where different files can be downloaded with the help of a directory service [7], [11], [24]. Peer-to-peer systems

such as Napster [5] depend on little or no dedicated infrastructure. There is, however, the implicit assumption that the individual peers participate for a significant length of time instead.

Recently, ideas of Overlay Network, where each connection in the overlay is mapped onto a path in the underlying physical network, are being discussed to facilitate both CDN and P2P. For example, Kazaa [22] organizes the clients into an overlay P2P network. But the performance of overlay changes with time as nodes dynamically join and leave the system, which is one of the problems to be resolved for P2P network. [19] proposed a CDN architecture where a set of intermediaries act as transit nodes (TNs), which organize themselves into a content delivery network and are used to replicate and forward data.

2.2 Contribution

In this paper, we propose an optimal algorithm for Web consistency to selectively decide both the replica to be updated and the surrogate to get the latest version from. The goal is to improve the user delay caused by the old-versioned data and to reduce the network traffic caused by delivering the data in Fig. 1.

In order to do that, two priorities are proposed: One is *consistency priority*, which is used to decide whether the replica should be updated or not. The other is *update priority*, which means the priority of node to get the updated data from.

The *consistency priority* is proposed by predicting the access within its update period, based on a theoretical analysis of the Web access and the freshness time of objects. The *update priority* is calculated by the reduced delay during the delivery of the updated contents among surrogates.

How to use them are as follows: Firstly, each replica of the same content on different surrogates is assigned a *consistency priority*. When the original content is changed on its original server, instead of all its replicas over the whole network, only its replicas with higher *consistency priorities* than the threshold will be updated. Secondly, if one replica of a given content is selected to be updated, a surrogate with the lowest *update priority* will be selected to provide the latest version of this content. As a result, the latest version can be sent from an algorithm-decided site with the reduced network traffic and user delay.

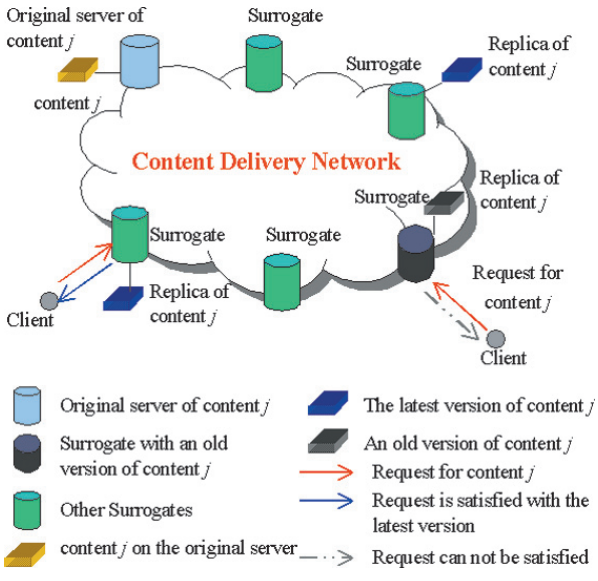


Fig. 1 Consistency control in dynamic content delivery.

3. Related Work

In *Propagation* method, the updated version of a document is delivered to all copies whenever a change is made to the document on the origin server. Although the copies always keep the latest version of the originals by the *Propagation*, this method may generate significant levels of unnecessary traffic if documents are updated more frequently than accessed.

In *Invalidation* [6], an invalidation message is sent to all copies when a document is changed on the origin server. This method does not make full use of the delivery network for content delivery and each replica needs to fetch an updated version individually at a later time. Therefore, the user-delay may get worse if a frequently accessed document can not be updated on time.

[9] addressed a set of models that capture the characteristic of dynamic content at the sub-document level in terms of independent parameters such as the distribution of objects size, their refresh times and reusability across time.

Cluster Lease [16] was designed to maintain data consistency by propagating server notifications to cluster of proxies in the content delivery networks. However, how to reduce the network traffic caused by the propagation between server and proxies is not mentioned.

[18] proposed a hybrid approach that can generate less traffic than the propagation approach and the invalidation approach. The origin server makes the decision of using either propagation or invalidation method for each document, based on the statistics about the update frequency at the origin server and the request rates collected by all replicas. However, the algorithm calculates the request times within a fixed period which is unrelated to the documents' lifetime. Moreover, the replicas on different surrogate are carried out by the same decision even different surrogate has different access pattern.

Some Web services employ the time to live (*TTL*) mechanism [8] to refresh their replicas. However, how to decide the proper value of *TTL* is still not resolved.

MONARCH [12] divided Web objects into several different groups based on object relationships and object change characteristics. Furthermore, it identifies the relationships among objects composing a page and used relationships to keep all objects consistent.

Although assigning different value of *M-TTL* (*MONARCH*-based *TTL*) to different groups can be a combination of *TTL* and *MONARCH*, how to cooperatively decide

the *M-TTL* and keep the consistency among replicas stored in different sites have not been resolved.

We ourselves proposed an integrated pre-fetching and replacing algorithm for the hierarchical image based on a cooperative proxy-server model, in which the metadata of the hierarchical image was used to keep the data consistency with user-satisfaction [23]. We also presented a scheme for stream caching by using hierarchically distributed proxies with adaptive segments assignment, in which "segment" meant a group of pictures [27]. This method clarified effectiveness of "local-scope" server cooperation (in the network) with per-segment management and discussed how to reduce the overhead in network.

4. Theoretical Analysis

In this section, we give an analysis of Web consistency over the Content delivery network. Firstly, in Sect. 4.1 we introduce the notations used in the network model. Secondly, theoretical analyses of Web access distribution are presented in Sect. 4.2. Then, based on the analytical results, how to reduce user perceived latency is discussed in Sect. 4.3. How to reduce the computation complexity and get the necessary parameters are introduced in Sect. 4.4. Finally, The mathematically optimized consistency algorithm is proposed in Sect. 4.5.

4.1 Notations

We assume that each surrogate is located in a different administrative domain, such as an autonomous system (AS). Let λ_i (bytes/second) denote an aggregate request rate from clients to the surrogate i ($i = 1, 2, \ldots, I$).

As for the contents, we assume that there are J different contents in our CDN. A parameter P_j defines the request probability for content j (i.e., content popularity). Here, B_j denotes the data size of content j. In this paper we look on content j as an update object specified by (j). And its origin server is defined as $o(j)$.

Let $X_{i,j,}$ be a parameter which takes a binary value of

$$\begin{aligned} X_{i,j} &= 1 \quad \text{(if object } j \text{ is stored on server } i\text{)} \\ X_{i,j} &= 0 \quad \text{(otherwise)} \end{aligned} \tag{1}$$

Then, we can get a matrix $\boldsymbol{X}$ of which one element is $X_{i,j}$, which represents a placement pattern of contents. In the matrix X, only if content j is available on surrogate i, $X_{i,j}$ is set to be 1. Otherwise, it keeps zero. We all know that it is hard to manage all contents on all surrogates, however, studies [28] show that the surrogate-popularity shows a Power-Law distribution, where 80% of the requests to the Web contents is served by only top 4% most popular surrogates. Furthermore, Web contents also show extremely different popularity according to Zipf-like distribution. Therefore, in a real system, only a small part of popular Web contents on popular surrogates need to update their states of X (as its value can be exchanged according to the coming request), while other surrogates always keep its $X_{i,j}$ to be 0. This makes possible for us to manage the status of X in practical CDN system.

As for the link between two surrogates, $D_{m,n}(\boldsymbol{X})$ means the shortest distance (hop count) from surrogate m to surrogate n under the placement $\boldsymbol{X}$. And $C_{m,n}$ denotes the average bandwidth (per hop) along the above path from surrogate m to surrogate n. Table 1 presents a list of parameters used in our network model.

4.2 Definition of Web Access Distribution

Let $\Lambda = \sum \lambda_i$ be the total request rate from all the domains. Then, for a given surrogate i, its surrogate popularity can be given by λ_i/Λ.

According to the Zipf distribution which the distribution of Web access follows, the probability that the content j is requested can be obtained as follows:

$$P_j = \frac{\Omega}{r_j^{\alpha}} \tag{2}$$

where Ω, α are parameters of the Zipf distribution, and r_j is the ranking of request times. Therefore, we can get the probability that a request happens for the content j from surrogate i by:

$$P(i, j) = \frac{\Omega}{r_j^{\alpha}} \cdot (\lambda_i/\Lambda) = \frac{\Omega \cdot \lambda_i}{r_j^{\alpha} \cdot \Lambda} \tag{3}$$

Reference [9] shows that the freshness time of objects follows a Weibull distribution with a CDF:

$$F(v) = 1 - e^{1-(av)^b} \tag{4}$$

where v is the freshness time and a and b are constants of the Weibull distribution. In this case, for content j, the mean $E(v)$ (called MTTF or MTBF) of the freshness time is given by:

$$E(v_j) = a_j^{-1/b_j} \cdot \Gamma\left(1 + \frac{1}{b_j}\right) \tag{5}$$

where a_j, b_j are parameters of the Weibull distribution and $\Gamma(\cdot)$ is the gamma function.

Table 1 Client workload and model parameters.

Parameters	Definition
$o(j)$	Server originally stores content j
r_j	Ranking of request times of content j
λ_i	Aggregate request rate to surrogate i
B_j	Data size of content j
P_j	Request probability of content j
$X_{i,j}$	Element of placement matrix X
$t_{0,j}$	Time when content j was updated last time
$D_{m,n}(\boldsymbol{X})$	Shortest distance (hop count) from surrogate m to surrogate n under the placement $\boldsymbol{X}$
$C_{m,n}(\boldsymbol{X})$	Average bandwidth (per hop) during the path from surrogate m to surrogate n under the placement $\boldsymbol{X}$
$R_{i,j}$	Consistency priority of content j on the surrogate i
$\Delta T_{i,j}$	Update priority of content j on the surrogate i

Assume the time when content j was updated last time is $t_{0,j}$, and during the period from $t_{0,j}$ to $(t_{0,j} + E(v_j))$, there are $W_{E(vj)}$ total requests happened in the whole CDN system. The client requests happen according to a Poisson process, for example, and then the number of requests for content j from surrogate i within this period can be obtained by:

$$R_{i,j} = W_{E(v_j)} \cdot \frac{\Omega}{r_j^{\alpha}} \cdot (\lambda_i/\Lambda) = \frac{\Omega \cdot \lambda_i \cdot W_{E(v_j)}}{r_j^{\alpha} \cdot \Lambda} \quad (6)$$

If the $R_{i,j}$ is greater than one, it means at least one request happens for this object since it has been changed last time. Then, to avoid sending the invalidation version of the data, the replica of content j on surrogate i should be updated when the original changes next time. Here, we define $R_{i,j}$ as *consistency priority*, of which usage will be discussed in Sect. 4.5.

Note that CDN service providers often provide stronger guarantee mechanisms for content replication to the client content providers who pay more money. Our proposal can be extended to satisfy the above requirement by introducing a price index (PR_j). Here, PR_j is the price index of the client who requested the content j. Then, the adjusted request times can be $PR_j * P(i, j)$ and the Eq. (6) will be:

$$\begin{aligned} R_{i,j} &= W_{E(v_j)} \cdot \frac{\Omega \cdot PR_j}{r_j^{\alpha}} \cdot (\lambda_i/\Lambda) \\ &= \frac{\Omega \cdot PR_j \cdot \lambda_i \cdot W_{E(v_j)}}{r_j^{\alpha} \cdot \Lambda} \end{aligned} \quad (7)$$

In the above equation, as the price is introduced, for the same content with a different price index, the one which has larger price index PR_j (which means the requester pay more) will have a larger $R_{i,j}$, then it will be firstly considered to be updated. In other words: CDN service providers can often provide stronger guarantee mechanisms for content replication to the client content providers who pay more money.

4.3 Minimization of User Perceived Latency

In this subsection, we make the analysis of network traffic cause by sending the modified version of a document. Our goal is to fetch the latest version of the modified document from an alternative surrogate instead of the original sever to minimize the user perceived latency.

When a request happens for content j from a given surrogate i where the latest version of content j is not available, we assume that there are K surrogates where the latest version of content j is available except the original server $o(j)$. Let $k(i, j)=\{1, \ldots, K\}$ represent one of the surrogates storing latest version of content j, we can get:

$$\begin{aligned} &K \leqq I \;\&\; k(i, j) \neq i \\ &k(i, j) \neq o(j) \;\&\; X_{k(i,j),j} = 1 \end{aligned} \quad (8)$$

where I is the number of servers (i.e. surrogates). These equations mean that surrogate $k(i, j)$ must be neither the original server ($o(j)$) of the request content nor the surrogate (i) where this request happened, and it has the replica of the requested content j ($X_{k(i,j),j} = 1$).

Assume that content j is originally stored in server $o(j)$ and $C_{k(i,j),i}$ is the average bandwidth (per hop) during the path from surrogate $k(i, j)$ to surrogate i. Let us denote Q_j be the ratio that the requested content j is not the latest version. Then, when a request for content j happens at surrogate i and surrogate $k(i, j)$ sends the latest version to satisfy this request, the user delay during the delivery from surrogate $k(i, j)$ to surrogate i is given by

$$T_{k(i,j),i,j}(X) = \frac{1}{\Lambda} \lambda_i \cdot B_j \cdot P_j \cdot Q_j \cdot D_{k(i,j),i}(X)/C_{k(i,j),i}(X) \quad (9)$$

where $\Lambda = \sum \lambda_i$ is the total request rate from all the domains. If we continue to define:

$$G_j = \frac{1}{\Lambda} \cdot B_j \cdot P_j \cdot Q_j \quad (10)$$

$$U_{k(i,j),i}(X) = D_{k(i,j),i}(X)/C_{k(i,j),i}(X) \quad (11)$$

Eq. (9) becomes as follows:

$$T_{k(i,j),i,j}(X) = \lambda_i \cdot G_j \cdot U_{k(i,j),i}(X) \quad (12)$$

Similarly, if a client sends a request for content j to surrogate i and original server $o(j)$ sends the latest version to this client, the user delay during the delivery from server $o(j)$ to surrogate i is given by

$$T_{o(j),i,j}(X) = \lambda_i \cdot G_j \cdot U_{o(j),i}(X) \quad (13)$$

For the whole system, if we always let the original server send the latest version to the surrogate where the request happens, then the average user delay can be obtained:

$$T_o(X) = \sum_i \sum_j T_{o(j),i,j}(X) = \sum_i \sum_j \lambda_i \cdot G_j \cdot U_{o(j),i}(X) \quad (14)$$

For the whole system, if a client sends a request for a given content j at a given surrogate i, and a selected surrogate $k(i, j)$ (instead of the original server) sends the latest version to this client, the user delay is given by:

$$\begin{aligned} T_k(X) &= \sum_i \sum_j T_{k(i,j),i,j}(X) \\ &= \sum_i \sum_j \lambda_i \cdot G_j \cdot U_{k(i,j),i}(X) \end{aligned} \quad (15)$$

For the surrogate $k(i, j)$ selected when a request happens for content j from surrogate i, we can calculate the reduced user delay $\Delta T_{k(i,j)}$ by taking a difference of Eq. (12) and Eq. (13).

$$\Delta T_{k(i,j)} = G_j \cdot \lambda_i \cdot (U_{o(j),i}(X) - U_{k(i,j),i}(X)) \quad (16)$$

Here, we define $\Delta T_{k(i,j)}$ as *update priority* and use this parameter in our proposal as described in Sect. 4.5.

4.4 Computational Complexity and Related Parameters

Since the scale of CDN is being increased recently, to manage all surrogates' all replicas' update will cause a great amount of computational complexity. Therefore, how to reduce the computational complexity should be considered. Fortunately, previous researches [20] showed that the distribution of web requests from a fixed group of users follows a Zipf-like distribution, where most of web requests to surrogates are just for a very small set of objects, for example top 10%. Furthermore, in [28] it had been found that 80% of the requests to the Web contents is served by only top 4% most popular surrogates. Therefore, to reduce computation complexity of our algorithm, it is suggested that we only need to calculate the priorities for popular surrogates and popular contents.

As for how to get the necessary parameters to carry out the proposal: in our algorithm, $W_{E(vj)}$ need to be calculated to obtain *consistency priority* in Eq. (6). The Web log of CDN keeps the records of the request times when time goes on. There are some methods for prediction [25], [26], [32]. If the fresh time $E(v_j)$ can be predicted, the total request times $W_{E(vj)}$ during the period of $E(v_j)$ can be obtained. In order to grasp the current update period of the object, an auto-regressive (AR) model can be used to predict its current update period from its past records which are available as local information in CDN. Besides, the bandwidth $C_{k(i,j),i}$ needs to be measured to calculate *update priority* in Eq. (12). There are lots of methods to measure the available bandwidth including some algorithms such as *Pathchar* [33], *PacketPair* [30] and *SloPS* [31] with a guaranteed accuracy.

4.5 Proposed Algorithm

We present our algorithm as follows:

1. Step1: Scalable Update Selection

When a given content j changes at server $o(j)$, a *consistency priority* $R_{i,j}$ will be calculated according to Eq. (6). For content j's each replica ($X_{i,j} = 1$) over the whole network, only when its consistency priority $R_{i,j}$ is beyond the threshold *Th*, the replica of content j on surrogate i will be updated. Otherwise, this replica will not be updated until a new request for content j happens at the site i next time. Therefore, when a given content j is changed at its original server, not all its replicas ($X_{i,j} = 1$) over all network will be updated according to the analysis of Web access distribution.

Here, *Th* can be decided according the following situation, for example. If there is one request happened during the lifetime (from the time of last update to the update this time) of this content, then we can think this content should be updated in order to avoid sending the invalid (old) version of this content to the user when the request happens. In other words: this content is worth being updated as it would be requested once according to the analysis in our algorithm. So *Th* can be set to be 1 as an example in the practical view.

2. Step2: Scalable Lowest Delay Update

Assume that there are K ($X_{k(i,j),j} = 1$, $k(i, j) = \{1, \ldots, K\}$ & $K \leqq I$) replicas of content j selected to be updated, for each replica at surrogate $k(i, j)$, an *update priority* $\Delta T_{k(i,j)}$ (i.e. reduced user delay) will be calculated according to Eq. (16). The latest version of content j will be sent from surrogate $k(i, j)$ with the lowest $\Delta T_{k(i,j)}$. Therefore, the latest version will be sent from an algorithm-decided site instead of its original server resulting in a reduction of the user delay.

5. Evaluation of Algorithms

In this section numerical results will be presented by simulation experiments to validate the proposed algorithm.

5.1 Simulation Conditions

There are 21 surrogates in our network simulator. As some studies showed that most communication networks have Power-Law link distributions [15], [28], where the i'th most connected node has Ω/r_i^{β} neighbors, as for the network topology, we carry out our proposal under the Power-Law link distribution.

Because the distribution of web request has already proved to follow a Zipf distribution, which states that the relative probability of requests for the i'th most popular page is proportional to Ω/r_i^{α}, the access frequency is decided by this Zipf distribution with a Zipf parameter 0.72 [20], [21].

About the contents, there are 1000 different objects. We assume that the average size of objects is 10 KBytes and the size of an invalidation message for each object is 100 Bytes [18]. The threshold of the consistency priority in our algorithm is 1.

Client requests arrive according to a Poisson process. All clients are always redirected to the closest server without failure of request routing. The update rate of a given object is decided at random. The total request times in the simulations are 800000.

There are five replication algorithms we will study:

- Propagation Policy
- Invalidation Policy [6]
- Hybrid Policy [18]
- M-TTL Policy
- Proposal

To evaluate different algorithms, we use two performance measures. One is traffic generated during the process of sending a latest version of a given content. The other is *Old Hit*, which is the percentage of objects are invalid (not the latest version) when a request arrives at the replica.

5.2 Simulation Results

Figure 2 shows the result of *Old Hit*, which means that the

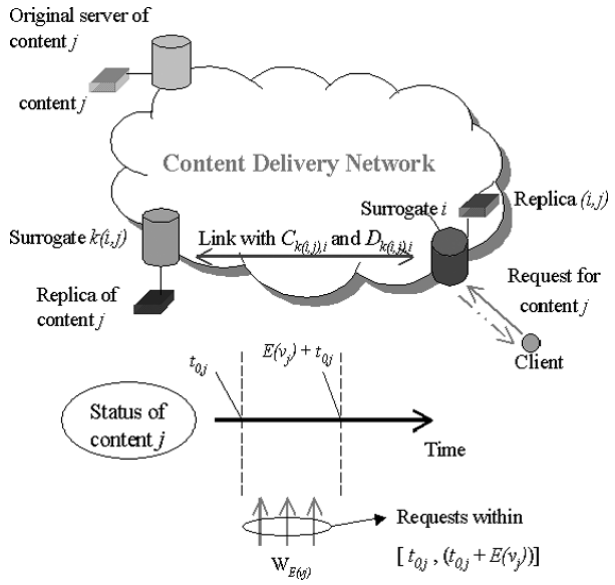


Fig. 2 Status of the content and surrogates.

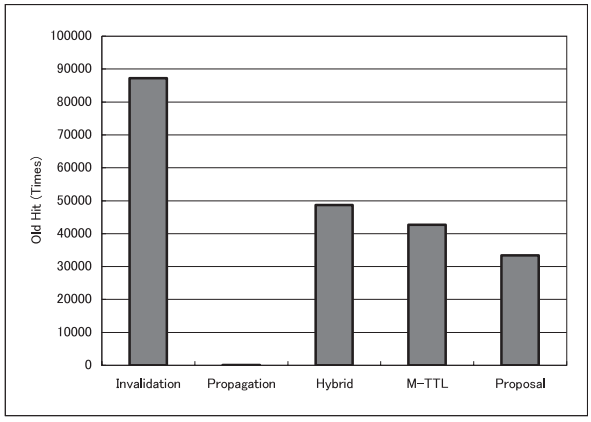


Fig. 3 Comparison of Old Hit with different replication algorithms.

requested data is not of the new version. Because the updated version of the requested document is delivered to all copies when a change is made at its origin server in the *Propagation*, its *Old Hit* is zero. However, the network traffic generated by the *Propagation* is the worst in Fig. 3, where the traffic caused by sending the new version is shown.

As for the *Invalidation* [6], when a change is made at its origin server, this method only sends an invalidation message instead of the content itself. Then, the client need to wait for the new version sent from the original server after the request. As a result, the *Old Hit* is the worst. Furthermore, sending the invalidation messages to all surrogates where the copies are stored also causes the additional network traffic. As a result, the total traffic can not be decreased efficiently.

The *Hybrid Policy* sets a threshold based on the request frequency, if the request is beyond the threshold, the *Propagation Policy* is carried out, otherwise, *Invalidation Policy* will be used. In *M-TTL Policy*, objects are divided into different groups based on the object change frequency. Different groups manage their objects separately according to the predicted *TTL*.

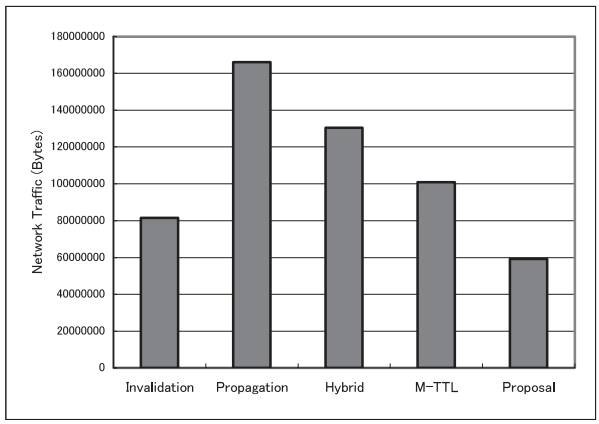


Fig. 4 Comparison of network traffic with different replication algorithms.

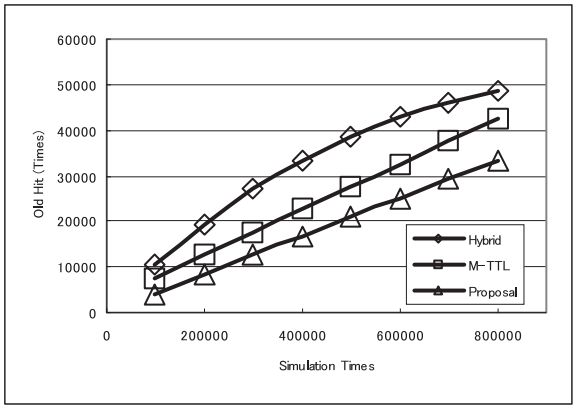


Fig. 5 Old Hit under different simulation times.

From the results in Fig. 3 and Fig. 4, we can obtain the following conclusions: The two conventional methods (*Propagation Policy* and *Invalidation Policy*) have their own drawbacks respectively. The *Propagation Policy* generates the worst network traffic and the *Invalidation Policy* causes the highest *Old Hit* and more network traffic compared with the other 3 methods (*Hybrid Policy, M-TTL Policy and Proposal*). As for the above 3 methods, they can get the balance of two conventional ones: Compared with the *Invalidation*, their traffics are closed to that of *Invalidation* but their *Old Hits* are much better. Compared with the *Propagation*, although the *Old Hit*s are more than the *Propagation*, their traffic can be greatly reduced.

Since these 3 methods (*Hybrid Policy, M-TTL Policy and Proposal*)' performances seem quite close compared with the other two methods (*Propagation Policy* and *Invalidation Policy*) in Fig. 3 and Fig. 4, we continue to test the performances of these 3 algorithms when the related parameters are changed.

We firstly test the *Old Hit* with respect to the simulation times. From Fig. 5, it shows that the proposed algorithm

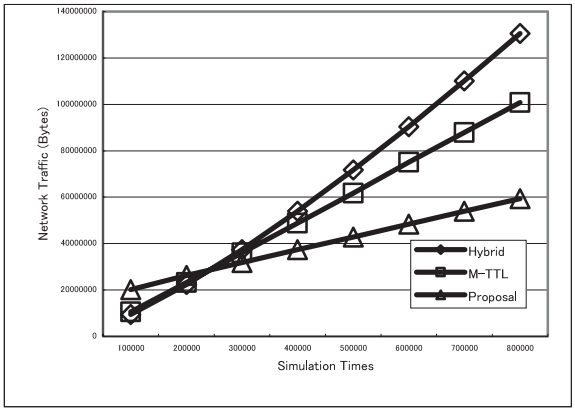


Fig. 6 Network traffic under different simulation times.

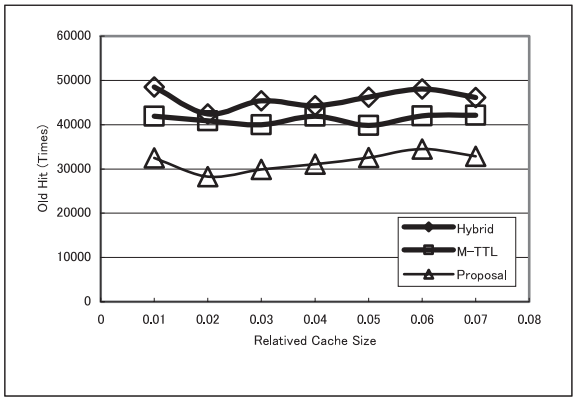


Fig. 7 Old Hit under different relative cache size.

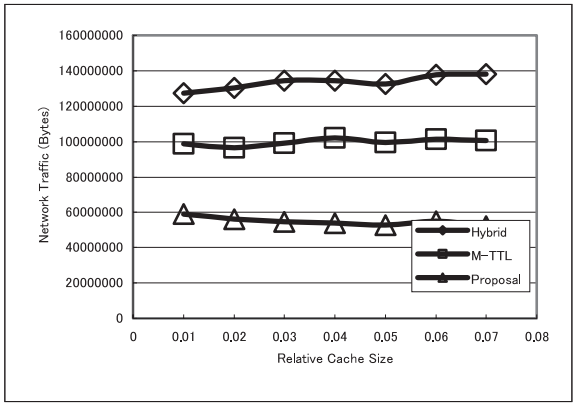


Fig. 8 Network traffic under different relative cache size.

obtains stable performance when the simulation times are increased. It always outperforms the other two algorithms (*Hybrid Policy and M-TTL Policy*). The reason is because the proposal decides whether an object should be updated in a surrogate by considering not only contents request frequency, but also the access pattern of surrogates and fresh distribution.

We then do the simulation about network traffic when the different simulation times are carried out. Similar result is shown in Fig. 6, where the proposed one reduced network traffic most.

Here, the performance of Hybrid Policy doesn't change much when simulation times are increased. In Hybrid Polity, whether Invalidation or Propagation should be carried out is decided by the rate of request times of this content (Z_j) to the total request times for all contents in CDN ($\sum Z_j$). For an given contents, with the increase of simulation times, its request (Z_j) will be increased, however the total request times for all object ($\sum Z_j$) is also increased. As the simulation is carried out by Zipf distribution, for an given content, the ratio ($Z_j / \sum Z_j$) of request times of this content to the total request times for all contents in CDN will not be changed. Then the decision of Hybrid Policy for the update of this content j will not be changed either.

Why our proposal is better than Hybrid Policy can be explained as follows: 1) In Hybrid, it calculates the request times within a fixed period unrelated to an given replica's lifetime (the time of being valid). But our proposal calculates the access time from the time of last update to the update time this time. It can reflect the latest popularity of contents since being updated. 2) In Hybrid Policy, for the different replica of the same content on the surrogate, it is looked as the same and is assigned to the same rate ($Z_j / \sum Z_j$). But, in our proposal, we look at the different replica of the same content j on different surrogate i to be different by different consistency priority ($R_{i,j}$) since different surrogate i has different request pattern and popularity distribution. As a result our proposal outperforms the Hybrid.

Finally, we change the relative cache size to test the performance of each algorithm. The relative cache size means the percentage of all contents that a surrogate can store in. Figure 7 and Fig. 8 show that the proposal outperforms the other two algorithms when the relative cache size is changed. Note even if the amount of total data changes, the result will be the same, because we use a relative cache-size. When the total data size is getting larger, the available cache space will also be increased as it is the relative size of total data. Here we just use the relative cache size to show the performance of every algorithm under the situation of the limited cache-capacity.

6. Conclusion

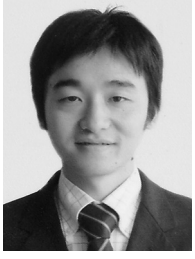
This paper discussed how to optimally manage Web consistency among replicas in content distribution networks and presented an efficient scheme to update them without wasting surrogates' resources. Based on mathematical analysis, we proposed a novel algorithm to minimize average user delay over traversed domains where the scalable content consistency is obtained. Our proposal dealt with not only popularities of contents and surrogates but also surrogate load. We then compared our proposal with other conventional methods using computer simulations.

There are a number of works to be done as further researches. Firstly, theoretical analysis should be expanded to be applicable to general cases. Secondly, implementation is

to be carried out.

References

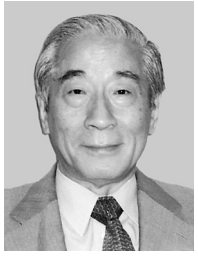
[1] J. Kangasharju and K.W. Ross, "Performance evaluation of redirection schemes in content distribution networks," 5th International Web Caching and Content Delivery Workshop, May 2000.
[2] A. Beck and M. Hofmann, "Enabling the Internet to delivery content-oriented services," Proc. 6th International Web Caching and Content Distribution, Boston, USA, June 2001.
[3] Adero, ⟨URL: http:www.adero.com⟩
[4] Akamai, ⟨URL: http:www.akamai.com⟩
[5] Napster, ⟨URL: http://www.napster.com⟩
[6] P. Cao and C. Liu, "Maintaining strong cache consistency in the World-Wide Web," Proc. 17th Int'l Conf. Distributed ComputingSystems, pp.71–86, May 1997.
[7] H.M. Radha, M.V.D. Schaar, and Y. Chen, "The MPEG-4 fine grained scalable video coding method for multimedia streaming over IP," IEEE Trans. Multimed., vol.3, no.1, pp.53–67, March 2001.
[8] V. Cate, "Alex: A global file system," Proc. 1992 USENIX FileSystem Workshop, pp.1–12, May 1992.
[9] W. Shi, E. Collins, and V. Karamcheti, "Modeling object characteristics of dynamic Web content," IEEE Globecom 2002, Taiwan, Nov. 2002.
[10] M. Chesire, A. Wolman, G.M. Voelker, and H.M. Levy, "Measurement and analysis of a stream media workload," USITIS'01, pp.259–309, San Francisco, CA, March 2001.
[11] S. Acharya, B. Smith, and P. Parnes, "Characterizing user access to videos on the videos on the World Wide Web," SPIE/ACM MMCN 2000, pp.130–141, San Jose, CA, Jan. 2000.
[12] M. Mikhailov and C.E. Wills, "Evaluating a new approach to strong Web cache consistency with snapshots of collected content," Twelfth International World Wide Web Conference, pp.20–24, May 2003.
[13] I. Cidon, S. Kutten, and R. Soffer, "Optimal allocation of electronic content," IEEE INFOCOM, pp.205–218, Anchorage, AK, April 2001.
[14] B. Li, M.J. Golin, G.F. Italiano, and X. Deng, "On the optimal placement of web proxies in the Internet," IEEE INFOCOM, pp.21–25, New York, NY, March 1999.
[15] L. Qiu, V.N. Padmanabhan, and G.M. Voelker, "On the placement of web server replicas," IEEE INFOCOM, pp.22–26, Anchorage, AK, April 2001.
[16] A. Ninan, P. Kulkarni, P. Shenoy, K. Ramamritham, and R. Tewari, "Scalable consistency maintenance in content distribution networks using cooperative leases," IEEE Trans. Knowl. Data Eng., vol.15, no.4, pp.813–828, July/Aug. 2003.
[17] S.-J. Lee, W.-Y. Ma, and B. Shen, "An interactive video delivery and caching system using video summarization," WCW2001, pp.1859–1869, Boston, MA, June 2001.
[18] Z. Fei, "A novel approach to managing consistency in content distribution networks," Proc. 6th International Web Caching and Content Distribtution Workshop (WCW'01), pp.71–86, Boston, MA, June 2001.
[19] S. Ganguly, A. Saxena, S. Bhatnagar, S. Banerjee, and R. Izmailov, "Fast replication in content distribution overlays," IEEE INFOCOM, pp.2246–2256, Miami, FL, March 2005.
[20] L. Breslao, 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, New York, April 1999.
[21] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "On the implications of zipf's law for Web caching," 3rd International WWW Caching Workshop, pp.2223–2245, June 1998.
[22] Kazaa, ⟨URL: http://www.kazaa.com⟩
[23] Z. Su, T. Washizawa, J. Katto, and Y. Yasuda, "Integrated prefetching and caching algorithm for graceful image caching," IEICE Trans. Commun., vol.E86-B, no.9, pp.2753–2763, Sept. 2003.
[24] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," ICS, pp.84–95, 2002.
[25] G. Antoniol, G. Casazza, G. Di Lucca, M. Di Penta, and E. Merlo, "Predicting Web site access: An application of time series," Proc. IEEE the third International Workshop on Web Site Evolution, Florence, Nov. 2001.
[26] J.E. Pitkow and M.R. Recker, "A simple yet robust caching algorithm-based on dynamic access patterns," Proc. Second World-Wide Web Conference, Amsterdam, 1994.
[27] Z. Su, J. Katto, T. Nishikawa, M. Murakami, and Y. Yasuda, "Stream caching using hierarchically distributed proxies with adaptive segments assignment," IEICE Trans. Commun., vol.E86-B, no.6, pp.1859–1869, June 2003.
[28] L.A. Adamic, B. Humberman, R. Lukose, and A. Puniyani, "Search in power law networks," Phys. Rev. E, vol.64, pp.046135-1–046135-8, 2001.
[29] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Proxy caching mechanisms with video quality adjustment," SPIE ITCom, Feb. 2001.
[30] S. Keshav, "A control-theoretic approach to flow control," Proc. SIGCOMM'91, pp.3–15, Zurich, Sept. 1991.
[31] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," Proc. SIGCOMM'02, pp.295–308, Pittsburgh, PA, Aug. 2002.
[32] G. Antoniol, G. Casazza, G. Di Lucca, M. Di Penta, and E. Merlo, "Predicting Web site access: An application of time series," Proc. IEEE the Third International Workshop on Web Site Evolution, pp.57–61, Florence, Nov. 2001.
[33] V. Jacobson, "Pathchar," ftp://ftp.ee.lbl.gov/pathchar/, 1997.



Zhou Su received the B.E. and M.E. degrees from Xi'an Jiaotong University, Xi'an, China, in 1997, 2000, and Ph.D. degree from Waseda University, Tokyo, Japan, in 2003, respectively. He was an exchange student between Waseda and Xi'an Jiaotong University from 1999 to 2000. From 2001 he had been a research associate at Waseda University and he is currently an assistant professor at the same university. His research interests include multimedia communication, web performance and network traffic. He received the SIEMENS Prize in 1998, and ROCKWELL Automation Master of Science Award in 1999. He is a member of the IEEE and IEE.

Masato Oguro received the B.S. and M.E. degrees in Science and Engineering from Waseda Univesity in 2005 and 2007, respectively. His research interest includes application layer multicast. He has been working with Nomura Research Institute, Ltd since 2007.

Jiro Katto born in Tokyo, Japan, in 1964. He received the B.S., M.E. and Ph.D. degrees in electrical engineering from University of Tokyo in 1987, 1989 and 1992, respectively. He worked for NEC Corporation from 1992 to 1999. He was also a visiting scholar at Princeton University, NJ, USA, from 1996 to 1997. He then joined Waseda University in 1999, where he is now a professor at the Department of Computer Science, School of Science and Engineering. His research interest is in the field of multimedia signal processing and multimedia communication systems such as the Internet and mobile networks. He received the Best Student Paper Award at SPIE's conference of Visual Communication and Image Processing in 1991, and received the Young Investigator Award of IEICE in 1995. He is a member of the IEEE and the IPSJ.



Yasuhiko Yasuda born in Tokyo on July 7, 1935. He received B.E. and M.E. degrees in Electrical Engineering, and D.E. degree in Electronic Engineering from the University of Tokyo, respectively in 1958, 1960 and 1963. In 1963 he joined the Institute of Industrial Science, the University of Tokyo as associate professor and was promoted to full professor in 1977. Since retiring from the University of Tokyo, Dr. Yasuda had been professor from September, 1992 at the Department of Electronics, Information and Communication Engineering (now the Department of Computer Science), School of Science and Engineering, Waseda University. At the end of March, 2006 he retired from Waseda University. He has been given titles of Professor Emeritus by both the University of Tokyo and Waseda University. His fields of interest have been digital communications, image coding and processing, Internet applications, and mobile and satellite communications. During his long career he has made some significant contributions including the invention of delta sigma modulation, the proposal of hierarchical image coding, etc. He is a past president of IEICE and a past president of IIEEJ. He served Chairman of Radio Regulatory Council (Soumu-shou), Chief of the IT Research Organization at Waseda University and so on. Dr. Yasuda has been awarded numerous prizes from various organizations including IEICE and NHK.