# Mobile Agent based Web Cache Proxy System

Katsumi KISHIMOTO†, Tomokazu NAGATA†,
Yuji TANIGUCHI‡, Shinji KONO† and Shiro TAMAKI†
†University of the Ryukyus, Graduate School of Engineering and Science
‡Center for Integrated information processings, University of the Ryukyus
†1 Senbaru, Nishihara, Okinawa, 903-0213 JAPAN
†Tel:+81-98-895-8715
†E-mail: katsumi@ads.ie.u-ryukyu.ac.jp

## Abstract

A distributed Web cache proxy is introduced to reduce both network traffic and the Web contents retrieval latency. A simple Web cache proxy has problems: there is a limit to the number of clients that can use the same cache, and thereby the effectiveness of the cache is limited. Also, if many clients connect to the proxy simultaneously, the connectivity, response and availability of the proxy is not guaranteed.

This paper describes a prototype implementation of Mobile Agent based Web cache proxy system, that uses Aglets, a kind of Java mobile agent. We investigate the effectiveness of our method by analyzing the log information of the conventional Web proxy.
**Key Word:** Mobile Agent, Guiding Prefetch, Proxy server, Web cache

## 1 INTRODUCTION

We introduce a distributed Web cache proxy system to reduce both the network traffic and the Web contents retrieval latency. A conventional simple Web cache proxy system contacts the host specified in the URL, requests an object, passes it on to the client, and caches it for future use. Since the Web proxy which required the client is only stored in the cache, the cache hit ratio does not increase. On the other hand, Web cache consistency mechanisms ensure that cached Web contents are eventually updated to reflect changes to the original data and reduce the network traffic. There are several mechanisms currently in use on the Internet. For example, cooperating caches, time-to-live-fields and so on.

Cooperating caches is one of the methods that shares the Web cache between Web cache proxy servers. For example, when a primary Web cache server does not have the Web cache which is required by a client, instead of requesting the Web server, the primary Web cache server asks the request from another cooperating Web cache server.

Time-to-live fields uses priority of an object's life time to determine how long cached data remains valid. Each object is assigned a time to live (TTL). When the TTL elapses, the data is considered invalid. TTLs are very simple to implement in HTTP using the optional "expires" header field specified by the protocol standard [1]. In practicability, the TTL is set to a relatively short interval, such that data is reloaded unnecessarily, but stale data are rarely returned. TTL fields are most useful for information with a known lifetime, such as online newspapers that change daily.

But these various methods are difficult in realizing the increasing hit ratio and Web cache consistency. Some of the problems are in the WWW structure itself. The WWW is a document distribution system based on a Client-Server model, and the simple WWW cache passes only the cache contents which was requested by a client. To solve these problems, an intelligent method is necessary for Web cache proxy systems.

### 1.1 Demands from various situations

We consider demands from various situations. Here, we picked three situations and their demands are as follows.

- Client demands

  1. When accessing some information which is needed on the Internet, searching by an Internet search engine is one of the effective methods. However there is no engine which covers all the Web servers. Since Web servers are increasing rapidly, improving the information discovery and access time is important.

  2. To reduce the network load, introducing an effective Web cache proxy needs a faster response time in various environments.

  3. Filtering the Web contents is useful. For example, Smart-Filter can filter the Web contents which was selected by an administrator.

- Server demands

  1. Web contents consist of images, links, documents and so on. However on proxy, getting Web contents is online. However, a Web server connects various clients. It is difficult to reduce the repeated requests and responses to the clients.

  2. Since almost all of the data is Web documents, images, movies and so on, and these data is sent to each client, it is difficult to reduce the server load.

- Network demands

  1. If many clients connect to a Web server simultaneously, the connectivity, response time and availability to a Web server is not reduced.

## 2 EXPERIMENTS BASED ON THE CONVENTIONAL SYSTEM

To analyze the conventional method, we used two Web cache proxy servers. "seagull" which is used at University of the Ryukyus campus, and "proxy-sc" which is used at the elementary and junior high schools attached to the University of the Ryukyus [2, 3]. We use Squid2.2STABLE5 [4] as the Web cache proxy server, and there is a "sibling relationship" between "seagull" and "proxy-sc". For example, if "seagull" does not have the cache which was requested by a client, "seagull" will ask the request from "proxy-sc".

These servers recorded all URL requests using HTTP and FTP protocols. In this experiment, we verify the log data from the point of hit ratio.

### 2.1 Experimental environment

Our campus has about 3,000 computer terminal units and "seagull" is open to the public. The elementary and junior high schools attached to the University of the Ryukyus have 100 computer terminal units, and more than 1,200 users used a Web browser that makes access to "proxy-sc". The network bandwidth is 10Mbps (local) and 6Mbps (outside). The specifications of "seagull" and "proxy-sc" are as follows.

Table I: Specification of the Web Proxy Servers (Squid)

| Machine | seagull | proxy-sc |
|---|---|---|
| OS | Slackware4.0 (Linux) | Slackware4.0 (Linux) |
| CPU | PentiumPro (300Mhz)x2 | Pentium II 400Mhz |
| Memory | 256M | 320M |
| HDD | 7.168GB | 6,144GB |

Figure 1 shows the request distribution of the two servers. Each of the points represent the amount of collected data. The access pattern is similar to each other, and the average of hit ratio between "seagull" and "proxy-sc" is 1.5 %.

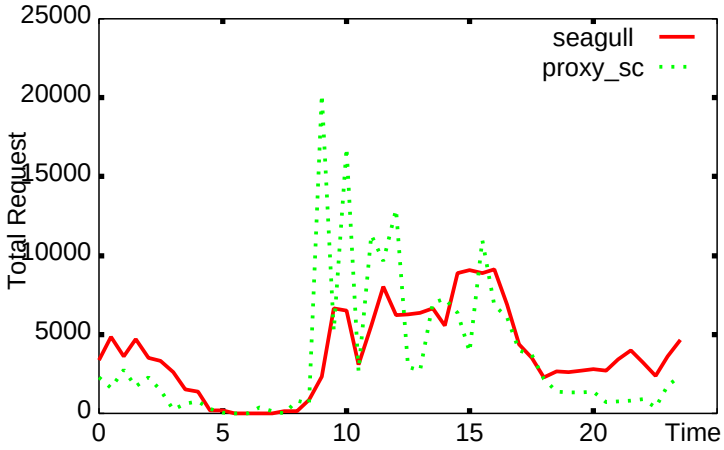


Figure 1: Request distribution of the two servers

Figure 2, 3 shows the amount of requested cache and hit ratio. If "seagull" does not have the cache which was requested by a client, "seagull" will ask the request to "proxy-sc", and vice versa. If clients have their own cache, a Web cache proxy which has sibling relationships is called a third level cache. In this case, the proxy cache is only used when the (first level) client caches misses. If the client cache is persistent between invocation of clients (i.e., the client cache uses disk), then the Web cache proxy hit ratio should decline over time as the per-client cache fills (or, in some sense, adapts to the user behavior). So in the cache of simple Web cache proxy, sibling relationships between neighbors have no effect to increase cache hit ratio. Thus there are many useless Web caches. The average of response time, $T_{avg}$ is defined as;

$$T_{avg} = T_{hit} \times p + T_{miss} \times (1 - p) \tag{1}$$

Where $T_{hit}$ is the response time if cache was hit while $p$ is a value between 0 to 1, representing the hit ratio. The value $T_{miss}$ is the case when cache was not hit. Supposing $T_{hit} = 5$, $T_{miss} = 60$ and $T_{avg} < 10$, then $p$ is found to be 90% by substituting these values in the formula 1. However, the actual hit ratio is about 30% on average. Thus a conventional method is not sufficient to increase useful Web cache and hit ratio.

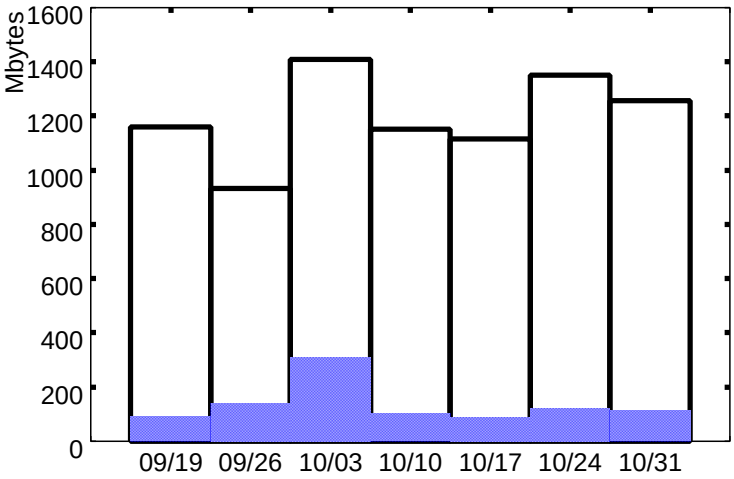


Figure 2: Cache hit ratio in "seagull"

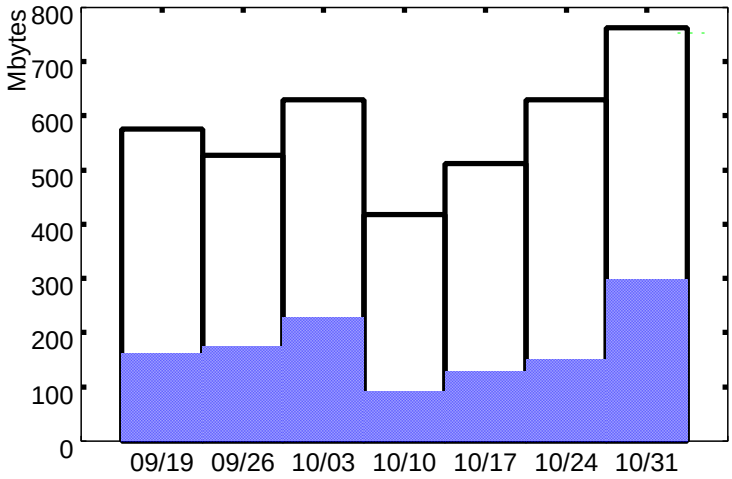


Figure 3: Cache hit ratio in "proxy-sc"

## 2.2 Simulations

One way to increase the Web cache proxy performance is simply to configure a caching proxy, and to collect the desired statistics. But this has some problems. For example, the WWW at present has no mechanism to determine if a cached content is not the latest version from the original server. So studies show that real user workloads can never cache all documents to measure the maximum possible hit ratio, because the users cannot accept old Web contents.

To solve these problems and to find a new factor which increases the Web cache hit ratio, we carried simulations under some conditions with the logs which was recorded at "seagull" and "proxy-sc".

## 2.3 Design

We want to observe the followings:

1. Does the Web proxy's cache have the same effect on "seagull" and "proxy-sc"?
2. How much effect does a simple prefetching of the Web contents have on the hit ratio of "seagull" and "proxy-sc"?

To investigate the above, we used two logs which were recorded at "seagull" and "proxy-sc" by carrying out the following experiments:

- Experiment 1: starting with no stored cache.
- Experiment 2: starting with no stored cache and a simple prefetching the Web contents.

## 2.4 Experimental results

The experimental results are shown in Tables II and III. The experimentation period was about one hour and we chose the best one from the results . Table II shows that the conventional Web cache have much effect at "seagull". On the other hand, Table III shows a different pattern for "proxy-sc". Table III depicts that for Experiments 1 and 2, for each TCP BYTES, the hit ratio in "proxy-sc" is about 40 %.

The TCP COUNTS hit ratio have more effect by the conventional Web cache, however, TCP BYTES hit ratio does not increase. These results mean that the conventional Web cache and simple prefetching of the Web contents which are required by the clients are difficult in increasing the hit ratio and reducing the network traffic.

To develop a new method that enables the Web cache proxy system to be optimized on the several network is necessary, we focus on the TCP COUNTS hit ratio of each table. Because the hit ratio on Table III is higher than that on Table II as shown by Experiments 1 and 2.

Table II: Simulation Results (seagull)

| | TCP COUNTS | | TCP BYTES | |
|---|---|---|---|---|
| | counts | hit(%) | Mbytes | hit(%) |
| Original Log | 1780 | 640.8(36%) | 19.03 | 1.5440(8%) |
| Experiment 1 | 1780 | 17.8(1%) | 90.91 | 0.9091(1%) |
| Experiment 2 | 1780 | 17.8(1%) | 97.99 | 0(0%) |

Table III: Simulation Results (proxy-sc)

| | TCP COUNTS | | TCP BYTES | |
|---|---|---|---|---|
| | counts | hit(%) | Mbytes | hit(%) |
| Original Log | 8987 | 1617.66(18%) | 50.58 | 22.2552(44%) |
| Experiment 1 | 8987 | 718.96(8%) | 57.37 | 22.3743(39%) |
| Experiment 2 | 8987 | 718.96(8%) | 57.36 | 22.9440(40%) |

# 3 OUR PROPOSED APPROACH

On the Internet, there are a lot of Web documents, images and so on. Thus it is difficult to cache all of them. To increase the cache hit ratio, one of the simple methods is to prefetch the Web contents which will be referred to clients. However, if simple cache prefetching is not optimized by each network form and conditions, the network traffic will increase. So we discuss a Web cache proxy system to solve these problems by using a Mobile Agent and Guiding Signal [5].

Mobile network agents are programs that can be dispatched from one computer and transported to a remote computer for execution. Once they arrive at the remote computer, they present their credentials and obtain access to local services and data. The remote computer may also serve as a broker by bringing together agents with similar interests and compatible goals, thus providing a meeting place at which agents can interact. In this paper, We used Aglet [6, 7] which is a Java mobile agent.

Guiding Signal is information which is able to make decisions, or to predict the behavior of a specific group. For instance, in a school class which uses the Internet, when a teacher appoints an URL as a demonstration, then all the students try to access the same URL.

For our experiments, the system was introduced to the Elementary and junior high schools which are attached to our University. We suppose that a request from a teacher is sufficient as a Guiding Signal.

## 3.1 Implementation

### 3.1.1 Client modifications

In the elementary and junior high schools, to build on an existing work, we modified the client's browser to connect "proxy-sc" by using JavaScript. If a client's browser cannot connect to the proxy, the client's browser will connect to the Web server directly.

### 3.1.2 Server modifications

We made two types of Aglets in order to realize our proposed system. The Observer Aglet is like a watchman of Squid's log, and creates the Prefetch Aglet. The Prefetch Aglet prefetches the links of Web contents which was given by the Observer Aglet. These Aglets will behave as a manager of Squid's Web cache. This system behaves as follows.

1. A client starts the Web browser. If the client's browser cannot connect to the proxy, then the client's browser will connect to the Web server directly.
2. If Squid receives requests from the teacher's browser, then Squid records on access.log.
3. If the Observer Aglet finds the request from the teacher's browser, the Observer Aglet creates a Prefetch Aglet.
4. The Prefetch Aglet is given the request by Observer Aglet, and will be dispatched to a remote Aglets server.
5. When the Prefetch Aglet arrives at the remote Aglets server, the Prefetch Aglet will request the Web content to the Web server, and will search links in the Web content.
6. The Prefetch Aglet requests the links to Squid. If it finished, the Prefetch Aglet will be disposed (See Figure 4).

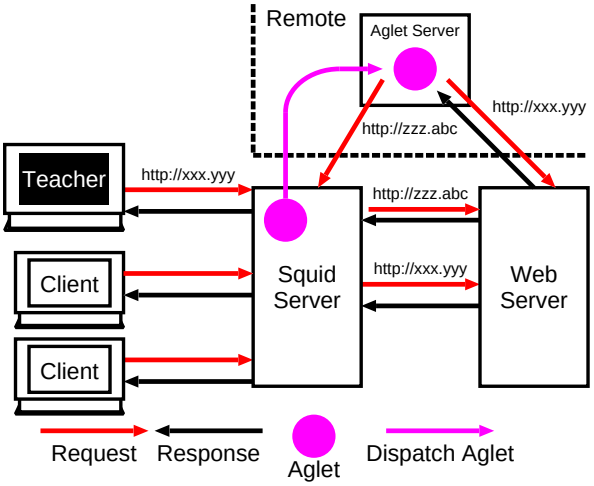


Figure 4: Construction of the Web cache system

## 3.2 Experimental analysis

We chose two school classes randomly which uses the Internet. In this experiment, a class with 37 students was used for the conventional method. For our proposed method we used a class of 50 students. Even though the size of classes is different, experimental evaluation showed that the size of a class does not influence the outcome. The experimental period was about one hour.

Figure 5 shows the distribution of Web cache hit ratio for each client and the corresponding amount of data. For example in Figure 5, in the case of $x = 5.0Mbytes$ and $y = 35\%$, the amount of data which was hit is found to be $1.8Mbytes$. Also the maximum hit ratio is 88% for the proposed Guiding Signal Prefetching and its average hit ratio was 53%, while the average for conventional method was 37%.

Figure 6 shows the request hit ratio for each client. This is almost similar to Figure 5. Thus the requests from a teacher are suitable Guiding signals. Table IV shows the response time from the proxy to a client. Although the Total requests is more than in the conventional proxy, the average response time is smaller than for the conventional proxy.

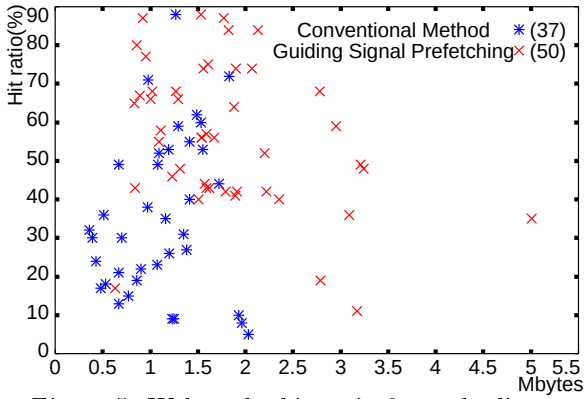


Figure 5: Web cache hit ratio for each clients

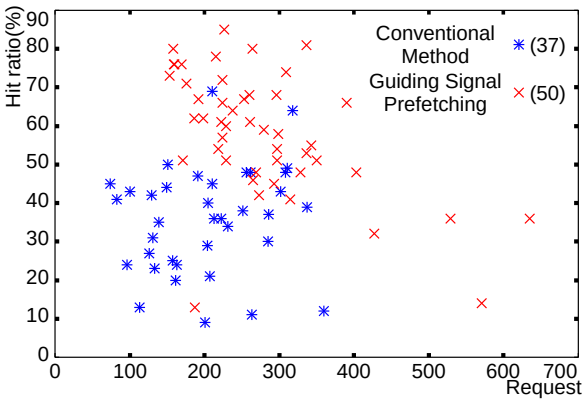


Figure 6: Request hit ratio for each clients

Table IV: Average of response time

| | Conventional Proxy | Our System |
|---|---|---|
| Total Requests | 7612 | 14852 |
| Total Response(s) | 29938983 | 21213488 |
| Average(ms) | 3933 | 1428 |

# 4 CONCLUSIONS AND FUTURE WORK

Our study of Web cache proxy system based on Mobile Agent and Guiding signal reveals the following insights for caching proxy design:

1. For our experiments, a caching proxy's hit ratio is usually low. We suppose that the reason is that when Web browsers use their own caches, the proxy acts as a second level cache.
2. Simple prefetching of the Web contents increase useless Web cache [8]. However, our Prefetch Aglet prefetches the links of Web contents based on the request from a teacher at the remote Aglets server, thus prefetching load and network load can be distributed.
3. Our Prefetch Aglet can select a Web cache proxy server optionally. This means that a Web cache can be distributed. Also our system can effectively increase the Web cache hit ratio and distribute load in each server.
4. If a Prefetch Aglet cannot reach the desired destination, the Prefetch Aglet will wait until the network route is recovered, or change the destination. Thus our system is able to avoid network obstacles.
5. Repeated requests can be reduced by Guiding signal.

In this experiment, we verified that our system can realize a high hit ratio of Web cache. But we considered only one Web cache proxy. As future work, our system will consider the multi-level caching proxy to distribute a useful Web cache [9].

Prefetching Caches requires some mechanism such that cached contents are up-to-date. For example issuing a "Conditional Aglet" which can make the Web cache up-to-date. However, these simple methods increases network load. As future work, we will also consider the cooperation between Mobile Agent and Web access engine [10, 11].

## References


[1] "Hypertext Transfer Protocol HTTP/1.0", HTTP Working Group Internet Draft, October 14, 1995

[2] "Center for Integrated information processings, University of the Ryukyus", http://www.cc.u-ryukyu.ac.jp/.

[3] "Open System Network", http://www.osn.u-ryukyu.ac.jp/.

[4] "Squid Internet Object Cache", http://squid.nlanr.net/Squid/.

[5] Katsumi KISHIMOTO, Tomokazu NAGATA, Yuji TANIGUCHI and Shiro TAMAKI"Mobile Agent based Class-support Web cache system", IEICE Technical Report, Vol.99, No.500, pp.55-60 (in Japanese) (1999.12.14).

[6] "IBM Aglets Home page", http://www.trl.ibm.co.jp/aglets/.

[7] "Aglets Software Development Kit", IBM Tokyo Reseach Laboratory, 1998.

[8] Ken-ichi CHINEN, "Studies on Effective Methods of Providing and Retrieving Information in The Internet", 1998.

[9] Radhika Malpani, Jacob Lorch and David Berger. Making World Wide Web Caching Servers Cooperate. In Fourth International World Wide Web Conference, pages 107-110 , 1995.

[10] Hideki YAMANAKA, "Web Objects Organizer - Autonomous Distributed Web Load Management System -", IPSJ, 99-DPS-93, Vol.99, No.30, pp.19-24(1999).

[11] Yamana H., Tamura K., Kawano H., Kamei S., Harada M., Nishimura H., Asai I., Kusumoto H., Shinoda Y. and Muraoka. Y.: : "WWW Information Collection with Distributed WWW Robots", DEWS98,No.24 (in Japanese) (1998.3.5-7).