Analele Universităţii de Vest, Timişoara
Seria Matematică – Informatică
Vol. , fasc. ,

A New Mechanism for Fast Delivery of Proxy Cache Objects

Florian Mircea Boian and Darius Vasile Bufnea

Abstract. This paper advocates the introduction of a new web based mechanism to solve the so called Duplicate Transfer Detection problem. This mechanism brings some obvious advantages regarding the download time and the traffic amount of data in the browser - proxy server - web server architecture. It is intended to be deployed in large enterprise networks having a significant number of web clients that access the Internet through a corporate proxy server. Although a solution to the Duplicate Transfer Detection problem had been previously proposed, it seems that its deployment in Internet has been dropped. By our own performed measurements, we urge the implementation and the deployment of a solution for this problem. The paper suggests a web services based implementation in this sense. Our proposed model is perfectly interoperable with the existing web architecture and can coexist with classic entities of the browser - proxy server - web server architecture not aware of our proposed model. This facilitates a transparent in-time migration to the presented model, whose benefits are also revealed.

AMS Subject Classification (2000). 68M10; 68M14; 68U35
Keywords. proxy server, proxy cache, web object, MD5 checksum

1 Introduction

Currently, a web client that accesses the Internet through a corporate proxy server can benefit from already cached web objects that other clients have previously requested. This will result in fast object delivery to the client (i.e. higher navigation speed), less traffic from the Internet towards the corporate network (i.e. less bandwidth usage, higher bandwidth available to other client / applications) and lower web server load (having less objects to serve to clients). This existing proxy-based web architecture fit everybody: from the corporate user to the corporate staff and web servers' maintainers. However, there are certain situations, very often encountered, when the proxy server will retrieve from the Internet an object that has been already cached. This is because the object is identified by its URL (Uniform Resource Locator) rather by its content. This problem is known as the Duplicate Transfer problem.

This paper introduces a new web-based mechanism for caching and serving web objects by their content. The serving part regards both the web server and the proxy server. The introduced mechanism is transparent to the end-user - the only user side aspect of the proposed mechanism is a faster delivery of certain web objects. No modifications or additions are necessary client side, fact that might ensure an easier and faster deployment process of the presented mechanism.

This paper is structured as follows: first, we will present previous work related to traffic improvements at application level, especially improvements dealing with the web traffic. We will present next the Duplicate Transfer Problem and present the previously proposed methods for treating this problem. A scenario often encountered in today's enterprise networks, induced by our own network administration experience will be presented next, scenario that reveals the disadvantages of the current state of facts in the browser - proxy server - web server architecture. These disadvantages together with some statistical results from an experimentally proxy cache analysis will reveal the imperative needs for a new mechanism in the web client - proxy server - web server architecture in order to deal with the Du-

plicate Transfer problem. We will continue with a web services approach for dealing this problem, approach that perfectly integrates in the existing web framework. The paper ends with conclusions and future work approaches.

2 Previous Work

Considering the dominant position of the Web traffic in the current Internet it is a natural need for the research community to focus its efforts in developing new mechanisms for improving the Web traffic. Currently, some developing mechanisms in these directions regard:

- Improving the download time from the client point of view, by dynamically select a web mirror closer to the client;
- Proxy cache adaptation of multimedia content in order to reduce client download time and / or to improve the playback quality for embedded clients [1];
- Dynamic computation of the Web session life time, function of the user behavior over time [2], [3].

Regarding the Duplicate Transfer Detection, the Squid proxy server development projects list includes the Duplicate Transfer Detection project [4]. Unfortunately, this project is listed as stale project, not being actively developed. The Duplicate Transfer Detection project was developed for Squid version 2.4STABLE7 a version which is five year old.
All these methods, located at the application level of the TCP/IP stack, do not exclude each other, rather they are complementary.

3 Duplicate Transfer Detection

In this section we take a deep analysis on what the Duplicate Transfer Problem means. Let $C = \{C_1, C_2, \ldots, C_n\}$ be a set of web clients located in the same LAN or in the same enterprise Intranet. These clients access web resources located in the Internet through a corporate HTTP proxy server.

We will denote this proxy server by S. An object O located at URL_O requested by a client C_i, object that has not been previously requested by another client, will be stored after retrieval in the web proxy's cache. Because in the proxy's cache, objects are indexed and identified by their URL rather by their content, in the situation when a second different client C_j will request an identical object P ($O = P$) but located at a different URL URL_P ($URL_O \neq URL_P$) object P will also be retrieved from the Internet, even if an object with a similar content already exists in the proxy's cache. This second retrieval process is a waste from the client C_j's spent time point of view or from the consumed bandwidth point of view.

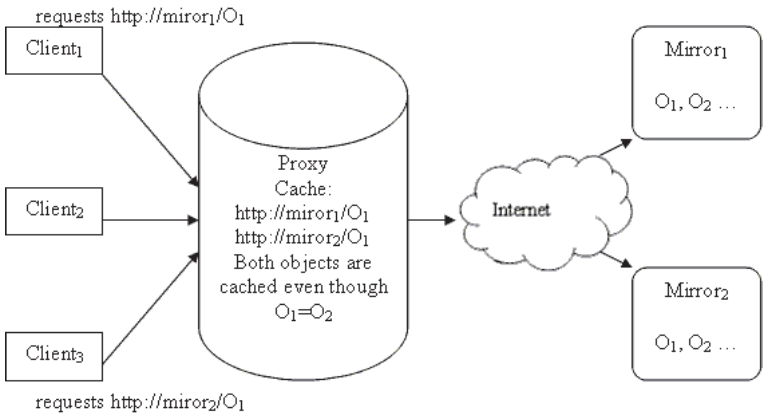

Figure 1: Department network architecture

A real scenario that covers the formal description above follows. Our Computer Science Department hosts a series of Linux workstations labs. Periodically, all these Linux workstations follow an automatically update process which implies retrieval of any available updates from the Internet. An update is identified by a web object located at a specific web repository. For load balancing reasons, the updates repository is replicated over multiple mirrors, each mirror being accessed with a different URL. This fact implies

that the same updates, located in repositories having different URL, are treated by the proxy server as different objects even they have the same content. If one of the Linux workstations is retrieving first the updates from a certain mirror M_1, these updates are cached by our department proxy server. Another Linux workstation that will setup de update process at a future moment of time, might be assigned to download these updates from a different mirror M_2, having a different URL from the first one. Even if the updates are the same, the proxy server will treat these updates as different objects and will retrieve them again from the Internet and also will cache them - i.e. will cache an object twice even if their content is the same.

The disadvantages of this behavior are obvious. First of all, any web client, starting with the second one, accessing a certain update might wait for the update being retrieved from the Internet even if the update is already in the proxy server's cache, located in a location that is faster to access. Secondly, multiple retrievals of the same objects leads to wasted network bandwidth. And finally, there is a waste of storage space at the proxy server level by caching objects having the same content more than once.

An alternative solution to this situation is setting up a repository of updates (a local mirror) located in our department's Intranet. By forcing the Linux workstations to retrieve the updates from this local repository we will save download time (all the clients access a very fast, proximity located update repository) and bandwidth (all the updates are downloaded only once when our mirror is synchronizing with other Internet mirrors). However, this solution has drawbacks. Setting up and maintaining a local repository might be an expensive process (from the human resources point of view). Also, this addresses the problem of our specific scenario, but does not resolve the general problem of having the same object being retrieved and cached by an HTTP proxy server more than once.

4 Duplicate Transfer Problem's Solution

The proposed solution by the research community [4] is identifying and indexing web objects at the level of a proxy server by their MD5 checksum. This new mode of identifying and indexing web objects is not to replace the classical one where web objects are identified and indexed by their source URL. It may be use as an alternative method for maintaining web objects at the level of a proxy server, especially web objects having a considerable size.

In order to maintain web objects at a proxy server by their MD5 checksum certain requirements have to be fulfilled by the web server where the web objects reside. The web server must notify a client requesting a certain web object of the object's checksum prior to the object's content delivery to that client - once a client (i.e. a middleware proxy server) receives an object's checksum it might not be interested anymore in that object's content because he locate the object by its checksum in his own cache.

The delivery of the MD5 checksum to the client, if it is available at the level of the web server, might be an implicit or an explicit process. In the first case, the web server might notify a web client, even though this might not be interested, about the checksum of the web object that will follow using an HTTP header. However, the web server might notify the client about the checksum only when the client explicitly requests it using a different HTTP request (beside the HTTP request for the web object itself). For this second approach we will present in the next section a web-services based model.

In order to deliver the checksum to the interested clients, this checksum must be available at the web server's level. There are multiple approaches for storing and computing the checksum for a web objects:

- The checksum is pre computed and is located in a file in the same web space (web folder) as the web objects. This is a very common situation, when for extremely large file offered for download, the client can also download a file containing the checksum of the large file, information that might be helpful in verifying the download. However, even if this checksum is often present it is never used in a mechanism as the one we proposed in this paper;
- The checksum is computed by the web server "on the fly" when a client sends the first request for a web object. While the web object content is read from the file system to be delivered to the client, the web server can also compute the content's checksum. This is a run-once process, because the checksum information can be stored in a web server memory cache and may be delivered to any other client that might request that web object again.

5 A Web Service for MD5 Checksum Management

An experimental analysis we have performed using [6] shows that the duplicate objects in proxy's cache occupy at least 10% of the total cache, while the number of duplicate objects reaches at least 12% from the total number of objects. In this situation, a solution to this problem is more than necessary, considering that the development of the Duplicate Transfer Detection in Squid figures as stale project, not being actively developed.

For resolving the Duplicate Transfer problem, we propose a web services based approach that will be used for managing the MD5 checksum of objects. We denote by `WSMD5` the web service itself. The web service will maintain using specialized calls we described bellow, a simple database `DB`, having a single table containing two fields: `(URL, MD5_URL)`. By `URL` we can identify an object from the Internet, while `MD5_URL` is the MD5 checksum of the object identified by `URL`.

The architecture of the web service we proposed is depicted in the figure 2. For simplicity and in order to reduce the overhead time, the `WSMD5` web

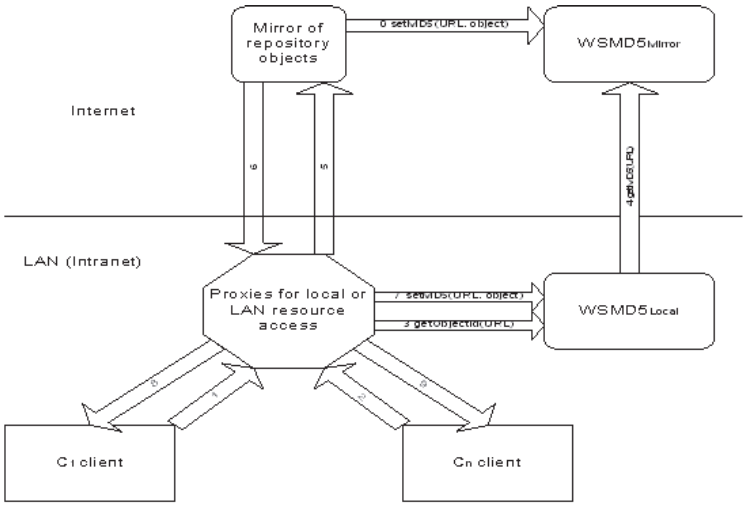

Figure 2: A web service architecture for MD5 dissemination

service can be based on the XMLRPC technology [5]. Because for this particular web service we proposed, there is no need for a discovery service (the proxy server will always know the location of the web object) and the message exchanged format (calls parameters) remain the same the XMLRPC remains the easiest and the most rapid exchange solution. However, in the case where a discovery service is needed or the message format is not the same, a SOAP solution is more suitable. There are only three procedures necessary in order to implement the web service:

`setMD5(object, URL)` - calls no 0 and 7 - computes the MD5 checksum of an object from the local proxy or local mirror, where the object is identified by `URL` and its content by `object`. As an effect, the procedure inserts the pair `(URL, MD5)` in the `DB` database. This is a local only call.

`getMD5(URL)` - call number 4 - is called from the local `WSMD5` to the `WSMD5`

mirror. This is an over web call which returns a very short string: a Base64 ASCII representation of the MD5 checksum requires only 24 bytes. If the mirror can return only MD5 checksum of an object, then the `WSMD5` mirror is not necessary.

`getObjectId(URL)` - call number 3 - returns a reference to an object from the local proxy's cache, the local stored object being equal with the object identified by `URL`.

The global usage scenario of the web service is described by the following algorithm:

```
Mirror Side:
When a new object is added to the mirror, the mirror calls
   setMD5(object, URLofObject);

Proxy Side:
If a client C1 from LAN makes a request to the proxy for a given URL
   Then
      call no 1;
End If;
The local proxy calls
   getObjectId(URL) to the local WSMD5 - call no 3;
For this call, WSMD5 searches by URL in the local data base
   a pair (URL, MD5_URL);
If exists
   Then
      return the reference to the local object in the proxy's cache;
   Else
      The local WSMD5 call getMD5(URL) to the WSMD5 mirror;
      Store the pair (URL, MD5) in local data base;
      Search the DB for an object having the MD5 checksum;
      If found,
         Then in the proxy's cache there is an equal object
```

```
            and return his id.
         Else
            getObjectId returns NULL;
            The proxy makes a normal request / response,
            using calls 5 and 6 to the web mirror.
            The proxy makes two more actions:
               A normal return of the object to the client C1 - call 8;
               setMD5(URL, object) - call no 7;
      End If;
End If.
```

If sometimes in the future, another client, suppose C_n - call no 2 - makes a request again to the same object as C_1 has done, for the second client the proxy will obtain in return a reference to the above object - call 9 - without sending any query out into the Internet.

6 Conclusions and Future Work

We proposed in this paper a new web-services based approach for managing and indexing web objects in a proxy's cache by their content and their MD5 checksum, approach that is perfectly interoperable with the current one that supposed web objects management by their URL. Our approach brings a series of obvious advantages such as: fast objects delivery to a client in certain situations, bandwidth savings from the corporate perspective and lower load from the server point of view.

As a future work, we will propose a basic reference implementation of our method. Also, some statistical analysis must be performed in order to determine the impact of the model and the percent of web objects that will be affected by our proposed mechanism.

References

[1] A. Sterca, C. Cobârzan, D. V. Bufnea, and F. M. Boian, *Evaluating Dynamic Client-Driven Adaptation Decision Support in Multimedia Proxy-Caches*, Proceedings of Knowledge Engineering Principles and Techniques, KEPT 2007 (June, 2007), 298-306.

[2] F. M. Boian, R. Boian, D. V. Bufnea, D. Cojocar, and Al. Vancea, *A Model for Efficient Session Object Management in Web Applications*, Proceedings of the Symposium "Colocviul Academic Clujean de INFORMATICA" (June, 2006), 137-142.

[3] F. M. Boian, D. V. Bufnea, Al. Vancea, A. Sterca, D. Cojocar, and R. Boian, *Some Formal Approaches For Dynamic Life Session Management*, Proceedings of Knowledge Engineering Principles and Techniques, KEPT 2007 (June, 2007), 227-235.

[4] *Squid: Optimising Web Delivery*, http://www.squid-cache.org.

[5] *XML-RPC Home Page*, http://www.xmlrpc.com.

[6] D. Bufnea, *A Tool for MD5 Checksum Retrieval of Squid Cache Objects*, http://www.cs.ubbcluj.ro/~bufny/md5.

Florian Mircea Boian

Department of Computer Science
"Babeş-Bolyai" University
Mihail Kogalniceanu nr. 1
Cluj-Napoca
Romania

E-mail: florin@cs.ubbcluj.ro

Darius Vasile Bufnea

Department of Computer Science
"Babeş-Bolyai" University
Mihail Kogalniceanu nr. 1
Cluj-Napoca
Romania

E-mail: bufny@cs.ubbcluj.ro