Semantic collaborative web caching

Lionel Brunie, Jean-Marc Pierson and David Coquil
LISI, INSA de Lyon
20 avenue A. Einstein, 69621 Villeurbanne cedex, FRANCE

Abstract

Too much information is no information. When the quantity of data becomes too large, users need help and tools to find their path in the information space. Nowadays, any user can browse through Terabytes of data and more. The difficulty is to find the best way among all possible documents. Our feeling is that a user is not generic and has always some particular interests. Thus, the percentage of relevant documents is small, the time to access these documents must be reduced at most.

We propose in this paper a collaborative proxy architecture, close to the user, based on “hot” relevance topics, and the dynamic construction of virtual communities. In this framework, the temperature of a document or a subject reflects its current interest among a community. We propose an architecture that allows proxies not only to manage efficiently cached documents but also to exchange documents with other proxies. Furthermore, these proxies manage meta-data about their partial views of the information space. We show how such an approach fits to enhance the global efficiency of the Web both in terms of access time to relevant documents and in terms of Web content indexing, adding semantic value where raw byte arrays are often considered.

Keywords: Semantic web cache, distributed information systems, proxies.

1 Introduction

Information systems have to face now a big challenge: interconnection challenge. Indeed, new network facilities (high performance LAN, Intranet, Internet) allow proposing new services based on the integration of distributed information systems into a single virtual entity. However information system interconnection poses several issues. The first evident issue is the necessity to use or develop a common representation of the target application. In this framework lie the works concerning ontologies, federations, agents or mobile agents, wrappers, brokers, etc. Another important issue is the performance of the global interconnected system. Indeed interconnecting information systems increases the data servers load, the network load, and may also cause severe query answer delays. This issue is becoming more and more crucial since, on one hand, when more users are connected to the network, the transactional load becomes more important; and on the other hand, queries concern now ”heavy” documents integrating images, scripts, videos. Finally, and this is the main key issue: too much information is no information. When the quantity of data becomes too important, users need help and tools to find their way in the information space.

In this paper we propose to use collaborative proxy architectures to address these issues. Proxies are commonly used to improve the efficiency of data servers and to reduce the latency (i.e., the delay faced by the user before he gets the document) by using local copies of the most requested documents (which allows one to avoid sending requests to the servers repeatedly). However, most of the current technologies base the collaboration between proxies on operational information (like the last access date or the number of accesses over a certain period) and do not consider semantic information (i.e. related to the content of the documents) nor contextual information (related to the users and their queries).

Oppositely, we believe that semantic and contextual proxy collaboration strategies should provide a better efficiency and offer users pertinent tools for searching documents. Thus we discuss below how correlating operational information with semantic and contextual information allows both optimizing proxy management heuristics (e.g. document replacement, document prefetching, document exchanges) and providing users with adaptive and personalized search indexes.

The remaining of this paper is organized into six
sections. Section 2 presents an overview of proxy technologies. Section 3 describes a novel model of collaborating proxies based on the monitoring of semantic and contextual information while section 4 details the modules of our architecture. Section 5 discusses how this proxy architecture can be used to provide users with pertinent information on the information space and the user activity and introduces the notion of virtual community. Section 6 exhibits some quantitative and qualitative results of the use of our platform. Finally section 7 concludes this paper.

2 Proxy technologies: an overview

Caching techniques, on which proxies are based, have been used in many domains of computer science for several decades. Basically, caches copy data delivered by a server in order to answer to data requests later in place of the server. The main expectations are the reduction of the data server load and the improvement of the latency time. These caching techniques are used in computer hardware, operating systems, distributed applications, visualization, etc.

Proxies implement caching techniques in the framework of distributed applications. Located between servers and users, proxies act as data or document caches. Thus, a proxy “covers” a set of users (e.g. University campus, company site, clients of an ISP (Internet Service Provider), etc.). Requests issued by users are actually sent to the proxy instead of being sent to servers. If the proxy owns a copy of the requested data or document, it immediately delivers it to the user. If not, the proxy forwards the request to the target server, copies the response document for future uses, and delivers it to the user. Thus, proxies can both reduce the servers load, reduce the network load, and improve the latency time, on condition that consistency problems between copies and original documents are well managed and that the so-called hit rate (i.e., the number of requests that concern documents yet held by proxies) is reasonable.

One of the key issue in the design and management of a proxy is the cache replacement policy. Indeed, the size of the cache is not unlimited. So proxy caches cannot hold all the available documents. As a consequence, it is mandatory, when the cache is full, to suppress some document(s) to get enough disk space to store the new ones.

Numerous proxy implementations have been proposed [?]. Thus, as one proxy can serve a limited number of users (for evident bottleneck reasons), collaboration protocols between proxies have earlier been proposed [?](e.g. hierarchical caches like Harvest [?] or Squid [?], or distributed caches including prefetching, hash-based caches [?] and directory-based systems [?]).

Proxies can be configured as a hierarchical tree: “bottom” (i.e. navigator cache), institution, national level and even international level. In such an architecture, if a proxy cannot answer a request, it recursively forwards it to the proxy of the higher level. Finally, when the document is found, either at a proxy level or at the document server level, it goes down the hierarchy, leaving a copy in each of the proxies along its path [?]. Proxies can oppositely be “flatly” distributed, with institution proxies only (and of course navigator caches). In this framework, each proxy maintains metadata about the documents stored in the other proxies. Thus, when a document is requested, a proxy does not need to forward the request to all other proxies: instead, it queries the metadata to check if a copy of the document is available in the cache of another proxy. This allows reducing the network traffic and optimizing the latency time. An intermediate solution between hierarchical and flat architectures consists in implementing, apart from classical proxies, metadata collectors, to gather information about the document availability in the proxies [?];

Finally, the location of the proxies over the network is of key importance. Basically, proxies can be either located close to the servers (in which case, they mainly act as server mirrors) or close to end-users in order to reduce the latency time as much as possible ([?, ?]).

In this paper, we propose new directions for improving the service delivered by proxies. Basically, we propose to monitor the semantic content of the caches and to use contextual (i.e. user centered) information to:

- improve the cache management policies,
- develop new collaboration schemes between proxies, including user proxies located on the users computers, and
- provide users with adaptive and enhanced search indexes.

3 Towards a new concept of proxy

In an interconnected information system (the Web can be seen as such a system), proxies do not store rough bytes: they store documents. These documents hold information which can be synthesized by descriptors or metadata. These latter can be either attached...
to the document or extracted from the document content. For instance, keywords, language and authors can be attached to an html page.

Similarly, information systems are not queried by isolated or "autistic" users but by users sharing some special interest or some experience or some common objective or behavior with other users. In that sense and from the information system perspective, "similar" users de facto constitute what we will call below a virtual community. For example, all the clients of an ISP that are interested for football constitute a virtual community.

Finally, users of the Web do not live in a rigid world. The queries they issue are strongly connected to the actual state of their domain of activity or to the fashion or to current events. Conversely, this means for instance that documents can be "hot" (i.e. often requested) one day and never more used later.

All these information (descriptors, user profiles, virtual communities, fashionable data) constitute what we called above semantic and contextual information. We argue in this paper that:

- this semantic and contextual information can be of great value to optimize proxy management policies and proxy communication protocols, and
- monitoring this semantic and contextual information can be of great value for users to optimize their use of the information system by giving them a more adapted and pertinent view of the "document space" (i.e. the available data).

Next sections discuss these two points.

Proxy architecture for semantic and contextual collaboration

Our global architecture is actually based on three levels, two being mandatory (see figure 1):

- **User Proxy (UP):** at the bottom level, one finds proxies located on user computers, where documents are normally cached. Oppositely with classical navigator caches which can be seen as a file warehouse, we implement here a true collaborative proxy mainly to be able to catch the user profile and behavior as well as context information. An added value of this is its non proprietary characteristic. User proxies collaborate on a peer to peer way to exchange documents and attached semantic and context information: each user proxy also collaborates with one Aggregate Proxy to inform it about the content of its cache;

- **Aggregate Proxy (AP):** this kind of proxy aggregates semantic and contextual information from User Proxies connected to it and maintains the directory of all the documents cached at User Proxy level (i.e. it acts as a directory of a partial virtual space). An Aggregate Proxy can collaborate with other Aggregate Proxies (if they exist), in order to enrich its knowledge of the entire cached document space. Optionally, if this Aggregate Proxy is deployed on an architecture offering disk space, it can cache documents in a hierarchical-like approach;

- **Meta Proxy (MP):** this optional level allows a concentrated view of several Aggregate Proxies. No data is actually cached here, but only an aggregate of semantic and contextual information and a directory of the actually cached documents at lower level (i.e. it acts as a directory of the entire virtual space).

4 Modules of the architecture

4.1 Indexing document

The information attached to a document is commonly related to one or more "theme(s) of interest": politics, sport, economy, etc. The task of identifying these themes is called "indexing". By recursively dividing themes into sub-themes, one can build an indexing hierarchy for the whole document collection.

At a given time, some themes are more popular than others (i.e. their documents are more frequently requested than others). These themes are called hot themes. For example, in Europe, in late May 2002 a hot sport theme is clearly the football (soccer) world championship.

We propose to use a data structure for representing semantic links between documents and indexing concepts. This data structure is a single-rooted n-ary tree. Internal vertices of this tree represent concepts, and leaves represent documents (see figure 2).

Internal vertices are connected by edges which represent a generalization / specialization relation between the parent and the child nodes.
We associate a weight to these edges. This weight is the probability of a request for a document located under the child node to be next requested after a document under the parent node in the hierarchy was requested. In other words, this weight represents the "correlation" (in terms of access patterns) between the target node and its "brothers" : the more brother nodes are semantically related, the heavier the weight attached to arrows issued from the parent node to these nodes is. So, children of the root node (representing large themes like politics, sports, education, tourism...) are weakly co-related so weights issued from the root node are small. Oppositely, brother nodes close to the leaves represent concepts which are closely related, like Zidane and the French soccer team, so the probability that after a document related to the French soccer team has been accessed, a document related to Zidane will be soon accessed, i.e. the weight attached to the edge soccer-Zidane, is high.

Edges from a concept to a document represent an indexing relation between the concept and the document. We also associate a weight to these edges. This weight represents the percentage of requests for this document among those indexed by this concept. Note that if a document is related to two concepts, then we duplicate the node representing this document and link the two created nodes to the two related concepts. This is done in order to avoid border effects.

4.2 Temperature

4.2.1 Definition

The temperature is a numerical value specific to each document. It represents the probability for the given document to be requested in the near future. More precisely, it is the synthesis between the number of requests for this document in the last time interval and the semantic links represented by the data structure.

A temperature value is also associated to internal nodes of the data structure. It is mostly used for intermediate computation purposes.

4.2.2 Temperature computation

The algorithm for temperature computation is outlined in Table 1.

Temperature computation occurs at regular requests intervals, every freq requests. The number of accesses

to each document between two consecutive computations is stored in an access table. During this process, documents that have been requested at least once since the last temperature computation are subject to warming, whereas other documents are subject to cooling.

In the first stage (line 02 to 09 of the algorithm), the temperature of 'warm' documents is increased by their corresponding value within the access table. This value is pushed in a stack to be used in a hypothetical future cooling of the document. The temperature of 'cold' documents is indeed decreased by an amount equal to their last increase in temperature.

In the next stage (line 10 to 18), the temperature variation for each document (∆Θ) is diffused along the edges of the data structure. More precisely, for each (document, concept) couple where there exists an edge of weight W between document and concept, the temperature of concept increases or decreases by W*∆Θ. The resulting temperature variation for concept may be further diffused to its parent node as long as the overall weighted variation remains greater than a given threshold.

Finally (line 19 to 27), the temperature variations for concepts is then conversely diffused from the concepts to the documents. Thus, note that a document that has been seldomly accessed can nevertheless see its temperature increase if its related concept has become hot. Indeed, this means that users are now interested by documents related to this concept, so there is a real chance that that document will be soon accessed.

4.3 Temperature, indexing, and its benefits

We summarize here the previous sections and outline the main points of interests. Basically:

- we first propose that proxies use metadata and keywords dynamically extracted by a content analysis of the documents to classify all the documents proxies process. Most documents are not delivered with comprehensive metadata and descriptors. Consequently it can be necessary to analyze and extract descriptors "on the fly" (i.e. at the time they are stored in the cache). Then they can be classified using a classical directory as the ones used in web search engine (we use the Yahoo classification). Thus, documents can be related to one or more 'topic(s)' or subject(s) ;

- A proxy can estimate the "temperature" of a document as the number of 'hits', i.e. the number of user requests that concern this document. The temperature of a subject can, on its side, be evaluated by integrating the temperature of all the
documents that are related to it. Some temperature dispersion heuristics can be used to propagate the temperature from a subject to the correlated subjects. The aim of this procedure is to track the "hot" subjects.

- The temperature function is being used in four ways:
  - to optimize the document replacement policy implemented by a proxy. Clearly, a document that is related to a hot subject should not be removed from the cache, even if its "operational" parameters (e.g. last access date) are not good;
  - to implement prefetching heuristics i.e. to pre-load documents before they are actually queried by users;
  - to optimize collaboration procedures. For example when a subject becomes hot in some proxy, another proxy can decide to prefetch a copy of the related documents from the first proxy. Similarly, proxies can exchange temperature information to optimize their cache management policies;
  - finally, temperature can be used to inform users about the currently hot subjects.

Note that we propose to compute this temperature both at the Aggregate Proxy level and also at the User Proxy level.

4.4 Navigator caches vs User Proxies

We propose to attach proxies to the user computer. Currently, browsers propose to use proprietary local caches. Unfortunately, none of them implements advanced features (except "basic prefetching" i.e. the downloading of all the documents linked to the currently displayed page). Furthermore, browser caches cannot communicate with proxies. So, there is a real need for developing a non-proprietary model of advanced collaborating User Proxy.

Apart from the fact that they will allow the global proxy architecture to deliver a better service (reduced network load, reduced latency time, reduced intermediate proxy load), User Proxies constitute the entrance door to customize and tailor the proxy system to user profile and demand. That is, user proxies:

- can count the number of times a document is loaded from its cache. This information (lost by traditional browser caches) can be used both for the computation of the temperature and, after transmission to the original server, for document access statistics;
- can filter and customize the semantic and contextual information to the user profile (see section 5)

5 Communities

5.1 Virtual communities

Users with the same static "profile" (e.g. people using the same Intranet, students from the same University, ISP clients that declare to be interested by the same subject, etc.) or the same dynamic profile (e.g. people browsing documents related to the same subject) define a virtual community. One can assume, from our proxy point of view, that they have the same "interests" (both in terms of proxy management policies and in terms of semantic and contextual information demands).

As noted in [?], this notion of virtual community can be used:

- to optimize Aggregate Proxies by associating each proxy to a restricted number of communities. Thus we believe to have more specialized proxies handling mainly related documents allows to optimize the whole community resources management;
- to provide users with pertinent information about the content of the proxy cache. With a simple browser, the user has a look to the content of an Aggregate Proxy where the temperatures of the documents are present, giving him the most possible accurate view of the interests in its community;
- to monitor the evolution of the interest of the community (which can be used both by proxy management heuristics and by users);
- to monitor the document usage and the links between documents. This monitoring can be done both at the Aggregate Proxy level and at the User Proxy level;
- to share browsing experiences in order to improve future requests;
- to allow users exchange their opinions about documents (if a user considers that a document he accessed is interesting, it can notify it to the search
engine connected to his/her proxy) which can lead the community to de facto build their specific (adapted to their needs) map of the part of the information system that interest them.

5.2 Setting communities

The subscription of a user to a community is decided by the user himself. A user may subscribe to several communities. As soon as a user is related to a community, its User Proxy is put in relation to the Aggregate Proxy in charge of the community. By this way, it enters in relation with the User proxy of other members of the community. In a hierarchical structure, administrators should organize the proxy collaboration structure in order to reflect the community structure and to deliver the best service. A user may dynamically choose to withdraw from a community (either definitively, or for the session time). He can also choose, when browsing the community proxy cache indexes, to suppress information coming from one community in order to narrow his/her view of the information space.

6 Experiments and discussion

In this section, we present the implementation options, we give results corroborating the adequacy of the temperature notion, and we discuss some qualitative benefits of the use of our platform hardly measurable in terms of speedup or hit rate.

6.1 Prototype implementation

The whole software is implemented using the Java language, for portability reasons. In a first stage and for experiment issues, the indexing tree implements the first three to four levels of the Yahoo! hierarchy, which gives us a limited number of concepts and a limited hierarchy level. This latter point is a real drawback, as discussed previously in 4. Thus, we have coupled our software with the ThoughtTreasure platform for commonsense reasoning [?], which is used when no relevance is found in our Yahoo-like hierarchy. ThoughtTreasure consists of a hierarchy of 27,093 atomic concepts with 35,020 English words and phrases and 21,529 French words (this bilingual feature is important for us as we aim to deploy the software in the French community as well). The weight of the edges between concepts are statically defined for the moment; we plan to automatically compute these from an analysis of the behavior of the users in a community. To index the web documents, concepts are extracted with a simple heuristic mixing the meta-tag keyword of the document (if present), the number of times a word is referenced in the document, the title of the document and the reference links to other web pages.

Today, the core parts of the prototype are developed: indexing, temperature computation and update, cache management policy and collaboration scheme. The integration part is now finished and we plan to make a bundle for evaluation in a few weeks. The evaluation of our platform is difficult to achieve, much more difficult than other traditional proxies [?], i.e analysing log files and deducing the hit rates related to any document cache policy. Indeed, log files are not enough for the evaluation of our strategy, since we have to examine the document in order to place it somewhere in the indexing tree. That means we have to visit all urls found in log files in order to know to which concepts it belongs. This hasn’t been done today (see future works) and we plan to add a little module to the web cache of our campus to get and store this information together with the accessed urls, so that to be able to simulate real users data.

6.2 Simulation platform for temperature relevance

We wanted first to experiment the temperature notion in itself. We conducted several experiments on simulated data to compare hit-rate with and without the temperature use. In our simulation platform, temperature was used: firstly, to determine the coldest elements in the cache to delete to make room for new incoming documents, and secondly to prefetch hot documents not already in the cache. The settings for the experiments were:

- number of documents in the document space: 10000
- number of low level categories for the indexing: 100
- number of documents in each category: 100
- all documents in a category are related to others by a weight of 50 %
- number of super-categories: 10, thus containing 10 categories each
- all documents in a super-category are related to others by a weight of 25 %

I.e. we use an even distribution. From the temperature point of view, an even distribution is not a favorable distribution; uneven distributions, which are more realistic, are usually more favorable since hot categories usually embrace more documents. However we did not want to influence the experiments in any directions.

4
Figure 3. Buffer hit for varying number of documents in the cache

- number of requests to the documents: 5000
- requests were randomly generated assuming that: 
  1) at a given time, some topics are more popular than others (hot themes); 2) if a document on a topic is being requested, documents related to the same one are likely to be requested in the near future;

We have tried to figure out the effect of the cache size on the final hit-rate of our algorithm, as compared with a LRU policy. Results (see figure 3) show that, for a significant number of documents in the cache, our methods outperformed LRU, giving a 26% hit-rate for 600 documents in the cache (1300 hit rate out of 5000 requests). This is due to the fact that the more documents are stored in the cache, higher is the probability that these documents belong to the same concepts, thus taking advantage of our temperature policy. We have also demonstrated that, if the distribution of requests follows a hot topic distribution (half the generated requests fall on a small number of hot topics (20), our approach outperform LRU by up to 100% on small cache (less than 100 documents) and 20% on large cache. Some other experiments corroborate the idea that our algorithm offers in most cases a better hit-rate than LRU when requests have semantic links.

Since one of the originalities of our platform is the extensive use of the temperature notion, the above experiments show the pertinence of the approach for caching data semantically linked. First experiments of a prototype of the whole platform show us that the temperature is well suited for a collaborative cache management policy. However, some benefits are not clearly measurable (difficulty to have a wide experimental platform) nor can they be reduced in a comparison of speedup or hit-rate: next section offers an overview of such benefits.

6.3 Discussion

In addition to the performance issue illustrated earlier in the document with the use of the temperature, we can outline several other interests from our platform, which are more qualitative than quantitative. In [?], the authors describe the impact of the collaborative scheme for the benefit of the hit rate, through real proxy data analysis and an analytic model of web behavior. They conclude that effort does not need to be spent anymore in the design and implementation of highly cooperative and scalable cache systems. Although we agree with a majority of conclusions the author describe, we disagree on the benefits one could expect from these proxies when users sharing interests share their proxies. We argue that the semantic value added in a simple cooperative proxy architecture outbreaks the limitations outlined for performance and hit rate related issues.

We exhibit in the following five particular points of interest of our architecture.

The first interest is the size of the cumulative virtual cache we obtain. Each User Proxy holds documents in its cache that is shared with others proxies. For example, our campus is composed of more than 18000 hosts. If they all implement an User Proxy of 100 MegaByte, we are in front of a 1.8 TeraBytes virtual disk space.

A second point is that we pushed the computation of the indexing algorithm and the temperature update to the end-user’s CPU. Indeed, such a resource is largely unused on most computers while the user browse the web. While he reads a downloaded document, some local computations and either communications with AP (to update temperatures) and other UP (for prefetching for instance) can occur without disturbing him in most cases.

Let’s imagine two Aggregate Proxies located one in New York City, the other in Los Angeles, related to sports: it caches locally meta data and documents about Salt Lake Olympics Games. On a typical day, because of jet lag, the users in New York have browsed the web before users in L.A. Some documents have been downloaded, the NYC Aggregate Proxy has been updated accordingly. While people in L.A. are still sleeping, one can imagine a communication between the two proxies, L.A.’s uploading locally the hot topics (U.S gold medals for example) mentioned by NYC’s, as well as the temperature map. Thus, when the user in L.A. browses the web, documents are already in its Aggregate Proxy: the download will be faster, and even he can use the information in its Aggregate Proxy to have a quick look to hot topics. This is relevant if people from both coasts share the same interests in their community. To offer this service, AP must have a cache space big enough to store retrieved documents, since it is not possible nor reasonable to push documents directly on User Proxies where computers might be sleeping (and even shut down).

Another point of concern is the fault tolerance inherent to the system, since documents may be duplicated. A kind of mirroring is offered at no extra cost. Let’s
imagine a call for participation for a U.S located conference in a scientific community. If one user from U.K. has downloaded the information (final program, registration forms, hotel and tourist information, ...), the probability that the same documents being downloaded also in Italy is high. The original U.S server might be down or highly loaded, thus if the two users subscribed to the same community, the Italian guy shall use the U.K. copies of the documents instead.

The last point of interest is more a semantic value than a performance issue. The content of a proxy cache (actual content + previously stored documents) de facto provides a partial view over the global information system. This view has a real added value since it is related to the actual interest of the users. Indeed, trivially, if a document is or was cached it means that this document was requested by some user(s). This information is not sufficient to consider that this document is highly interesting nor to estimate that the user(s) that got it found it interesting. However, it means that this(these) user(s) at least had the feeling that this document could be interesting. Clearly, if 100 users had this feeling (i.e., requested the document) the probability that the document is interesting becomes higher. If they spent a reasonable time before they requested another document (tending to prove that they actually got a look on it), this probability becomes even higher, etc. Consider the Web. The key issue is not that the web does not offer enough resources but that there are so many documents available that finding out the right document is becoming more and more difficult. So, suppose Louis follows a distance learning course and that he wants to get a tutorial on multimedia information systems: it may be valuable for him to check if some other students in the group found out such teaching materials. In another context, if Jacques is interested by tennis, it may be interesting for him to know what are the web sites that are today the most accessed by tennis fans. If Paul requests a document, he may be interested to know what documents have been accessed by other users after they got the one he is reading (indeed, these documents may be correlated with this latter). So, we believe that document classification (i.e. proxy cache indexing), as it was described in section 4.2, and apart from its operational interest, may really be valuable to users to find their way in the jungle of large decentralized information systems.

Furthermore making this information accessible only requires adapting a simple search engine at the top level proxy and making it operational through a classical web interface or a specific GUI. However, if one can expect from people interested by politics that they are interested by in-depth information about the documents available in proxy caches that actually refer to this subject, there is no evidence that they are also interested by information about sport. In other words, information about document related to sports may be noise for them. In the proposed architecture, bottom proxies, as they have a direct and simple access to the user profile and to his current browsing session, can adapt, tailor the cache indexes to the user needs.

Future works

We have started a project to analyse the behavior of our proxy implementation, mainly to provide a test platform for it. These tests are intended to show the correctness of our indexation schema from real web users, as well as the relevance of the cache management policy together with the communication schema of the architecture (in terms of performances and tuning).

Another point is mainly a performance and implementation issue. In the preceding description, the User Proxy opens a communication with the remote User Proxy to fetch the document: for some reasons (network workload, remote overload, ...) it may be more suitable to ask directly the original server of the document. We plan to add this soon with the help of the Network Weather Service for instance (NWS, [?]) which allows to predict the transfer delay for a file from one point of the network to another. We plan also to add this information onto Aggregate Proxies to maintain information about multiple copies of the same document in the local virtual space with a cost associated with it.

Concerning the subscription of a user to a community, we will add an automatic subscription by the system, after considering the user profile and/or the user browsing session.

7 Conclusion

In this paper we discussed a new architecture for semantic cache of web documents. This paradigm is based on the use of a set of collaborating proxies. The collaboration is based on the classification of the documents stored in the proxy caches and on the computation of the temperature of documents and the subjects of interest. We discussed the pertinence of the temperature. We then presented how this concept of collaborating semantic proxies can be used to provide the user with valuable information about the information system and the current use context. Based on this
concept, we also proposed the notion of virtual community that associates users with a common interest to a dedicated proxy. This notion could allow both enhance the system performance and improve the semantic and contextual services provided to the users.

References

<table>
<thead>
<tr>
<th>Table 1. Temperature computation algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variables and data types:</strong></td>
</tr>
<tr>
<td>Tree: the data structure</td>
</tr>
<tr>
<td>Acces[]: access table document</td>
</tr>
<tr>
<td>Stack: stack for storing temperature variations</td>
</tr>
<tr>
<td>Parent: pointer to parent node</td>
</tr>
<tr>
<td><strong>Internal node:</strong></td>
</tr>
<tr>
<td>Parent: pointer to parent node</td>
</tr>
<tr>
<td>Child[]: table of pointers to child nodes</td>
</tr>
<tr>
<td>Delta: temperature variation in this computation</td>
</tr>
<tr>
<td>Temp_modif[]: table of temperature variation due to each child</td>
</tr>
</tbody>
</table>

01 For each document D do
02 If acces[D] == 0
03 Delta=Pop(D.Stack);
04 D.Temp -= Delta;
05 Else
06 Delta=Acces[D];
07 D.Temp += Delta;
08 Push(D.Stack, Delta);
09 Endif
10 P=Parent(D);
11 While (P!=Root and |Delta*Weight(P,D)|>Threshold)
12 Delta *= Weight(P,D);
13 P.Temp_modif[D]=Delta;
14 P.Delta += Delta;
15 D=P;
16 P=Parent(P);
17 EndWhile
18 EndFor
19 While ((N=BreadthFirstTraversal(Tree)) is not leaf)
20 For each child C of N do
21 if ((N.Delta*Weight(N,C)-N.Temp_modif[C])>Threshold)
22 if (C is a leaf)
23 C.Temp+=N.Delta*Weight(N,C)-N.Temp_modif[C];
24 else
25 C.Delta+=N.Delta*Weight(N,C)-N.Temp_modif[C];
26 N.Delta=0;
27 EndWhile