

# Influence of the Document Validation/Replication Methods on Cooperative Web Proxy Caching Architectures

Víctor J. Sosa\*, Leandro Navarro  
{vjsosa,leandro}@ac.upc.es  
Universidad Politécnic de Cataluña (UPC)  
Jordi Girona 1-3  
Módulo D6-113, 08071  
Barcelona, Spain  
Phone: +34-93.401.68.07  
Fax: +34-93.401.70.55

**Keywords:** cooperative caching, consistency, web

## ABSTRACT

Nowadays cooperative web caching has shown to improve the performance in Web document access. That is why the interest in works related to web caching architectures designs has been increasing. This paper discusses and compares performances of some cooperative web caching designs (hierarchy, mesh, hybrid) using different document validation/replication methods (TTL, invalidation, pushing, etc). It is shown how the performance in a cooperative web proxy caching architecture is affected by the document validation method that is implemented. Comparing typical caching scenarios we found that speedups for some combinations of distributed caching with validation methods can be between 1.3 and 2.5. If the only criteria of decision to construct a cooperative caching system is response time then it is easily to decide that the combination with a speedup of 2.5 is the best one. However, we found that the bandwidth consumption and the number of stale documents in that combination could be prohibitive. That is why we cannot decide to construct a specific cooperative caching system based on a limited number of decision criterions. This paper shows some trade-off and possible alternatives for constructing a cooperative caching system using different combinations of document validation methods with distributed caching architectures.

## INTRODUCTION

The idea behind Web caching consists in getting documents close to clients at a low cost. Caching has been successfully applied in computer memory hierarchies. The advantage in caching is based on the reference locality principle, which specifies that the data that have been recently used are likely to be used in the near future. Cooperating proxy caches are a group of caches that share cached objects and collaborate with each other to do the same work as a single Web cache,

but trying to attend more users in a scalable and efficient way. A topic of particular debate in cooperative web caching is the design of cooperative caching architectures (cooperative mechanisms) aimed at obtaining more efficiency (hit rates and response time). At present there are many protocols of cache communication [29] which show different levels of efficiency in a cooperative web caching system. Web caching reduces network load, server load, and latency of responses. However, web caching has the disadvantage that the pages returned to clients by caches may be stale. This means the pages may not be consistent with the version currently on the server. That is why the second important topic when discussing caches is that of document validation/replication mechanisms. Presently we can find many studies debating the design of cooperative web caching from these two perspectives: on the one hand, works comparing and designing cooperative web caching architectures that can be efficient (better hit rates and response time) and scalable [7][6][15][9][3][2][1], and on the other hand, works comparing and designing document validation/replication mechanisms to keep some grade of consistency in the cached documents [12][4][14]. There are no works talking about the influence or impact on the caching efficiency of combining some cooperative web caching architecture with different document validation/replication mechanisms. We can not say whether a cooperative web caching architecture is better than another if we do not specify which document validation mechanism is to be used. An analysis from this perspective gives us useful information for taking decisions when constructing a cooperative web caching system.

## Cooperative web caching

Three common approaches to implementing a large scale cooperation scheme are hierarchical, distributed (mesh), and hybrid schemes.

In *hierarchical caching* architectures, caches are located at different network levels. In most cases it is assumed that inferior levels in the hierarchy have better quality of service.

\* The author is also supported by COSNET-CENIDET México with the project 2440.01-PR and CIC-IPN México.

Thus, at the most inferior level in the hierarchy we can find the client caches (L1) which are directly connected to the clients (i.e. caches included in Netscape or MS Explorer). At the next level are found the institutional caches (L2), which could be located in some nodes of the campus network. In the next level upwards in the hierarchy we can find the regional or national caches (L3) which could be connected to a national backbone linking several universities or institutions inside or outside the country. At the present time the number of levels commonly used in a hierarchy is on the order of 3 [8][1][11][13], including client, institutional, and regional caches. Some popular hierarchies which follow this structure are the National Laboratory for Applied Network Research (NLNR) in USA [8], Korean National Cache [11], The Spanish Academic Network, Red Iris [10]. In hierarchical schemes, when a request can not be satisfied by client caches (L1), the request is redirected to the institutional caches (L2), which in turn forwards unsatisfied request to the regional or national caches (L3), at this level caches contact directly the origin server. When the document is found, it travels down in the hierarchy, leaving a copy at each intermediate cache. Further requests for the same document travel up the caching hierarchy until the request finds the document. The software that is most used in hierarchies is Squid [6], which is a descendant of the Harvest project [15].

In *distributed caching* architectures (specially *Mesh*) there are no intermediate caches defined by levels, rather, there is a single level of caches where they can cooperate to serve the requests generated by clients. Because there are no intermediate caches which store and centralize all the documents requested by low level caches, the institutional caches need another mechanism to share documents with each other. A popular mechanism to share documents is based on broadcast probe. In this case, caches query their siblings whenever they do not find the requested document in their repositories. This is done commonly by using ICP (Inter Cache Protocol) [23]. This mechanism can significantly increase the bandwidth consumption and client-perceived latencies due to the possible generation of a lot of queries. However this fact would not be a limitation if the cache system is connected by a high speed and reliable network. Other mechanisms that are used to discover shared documents in a distributed caching architecture are those that use directories or summaries [2][5]. This mechanism has the advantage of not requiring that caches send a great number of unnecessary queries to their siblings, which is reflected in bandwidth savings. A drawback of this mechanism is that the directories/summaries only give a probability of an object being found in a sibling, but do not guarantee that the object is really stored in that cache [5]. As further alternatives to distribute and discover documents in a distributed caching system we can find some proposals in

[16][17]. They propose to use hash functions to associate a request with a specific cache. With this approach there are no duplicated documents in more than one cache in the distributed caching system. Because of these hash functions, caches have not to interchange their content. A limitation of this approach is documents are less available, which means we need a very reliable network.

We define *hybrid caching* architectures as a combination of the two previous architectures, which consist of a hierarchy whose ramifications are formed by meshes. Some performance analysis of several caching architectures can be found in [21][1][9][7].

### **Mechanisms of document validation/replication**

The second subject of debate at the time of implementing an architecture of document distribution using cooperative caching architectures is the issue of mechanisms that allow to distribute and to maintain document consistency. We can find two broad strategies to replicate and validate documents in a caching system. These strategies either use pulling or pushing mechanisms. In pulling mechanisms, caches periodically poll the origin server (or an upper level cache, according to the defined architecture). In pushing mechanisms caches do not need to ask (poll) the servers periodically for the freshness of a document, the servers distribute the documents that are updated to interested or subscribed caches. Within these broad strategies we can find several mechanisms to replicate/validate documents. We have grouped them in 3 blocks: Time to Live (TTL), validate verification, and callback or invalidation. In the time to live approach (**TTL**), servers add a time stamp to each document requested by caches. That time stamp defines the period of time that the document could be valid. A typical example of the TTL approach is the Expire tag in HTTP/1.0 (TTLE). The validate verification mechanisms commonly are related to time to live mechanisms. A cache receives a document with a time stamp, this time stamp indicates the last time that the document was modified. When a client asks for that document the cache sends a validate verification message to the server including the time stamp recorded in the document. Server validates the document time stamp with the document last modified time, and it sends a no modification message (the document has been not updated) or the new document with a new time stamp. This approach is implemented in HTTP/1.0 using the If-Modified-Since (IMS) tag. This approach can be combined with expiration or leasing approaches. The IMS messages can occur in several instances of time, depending on the consistency degree that we want to have. For example, if we want strong consistency, caches have to generate an IMS message any time a page is requested by the clients. This would define a threshold time to validate a document equal to zero (**TTLO**). If we bet for a validation

threshold greater than zero (**TTLA**) then any time a cache receives a client request, the threshold has to be verified. If the actual time minus document time stamp is greater than the threshold, an IMS message has to be sent to the server otherwise the cache speculates about document freshness and sends the document to the client.

In callback or invalidation mechanisms, each server keeps track of all caches which have requested a particular page, and whenever that page changes, notifies those caches. This approach does not scale well in the limit of many readers per page; both the state required to store the list of readers, and the OS and network burden of having to contact every reader of a page when it changes, grow linearly in the number of readers. This scaling problem can be overcome by using multicast to transmit the invalidations (**IMC**). By assigning a multicast group to each page, and having clients joining the groups associated with the pages they have accessed. Somewhat related idea (**IMCP**) of pushing content (rather than sending invalidations) via multicast is described in [19][20] (pushing approaches). Multicast solves the scaling problems at the server, however it creates another one at the routers. Routers are required to keep the state of hundreds or thousands of addresses. Moreover, the rate at which clients would be joining and leaving multicast groups, as they read and discard documents will likely create an unscalable overhead on the routing infrastructure. This problem can be solved when the volume of the contract is all the documents in a cache instead of a single document group as it is described in [4].

### SITUATION

As we have seen in the previous sections there are many researches related to cooperative caching systems. A missed point that we have found in those works is that most of them concentrate on a particular caching design topic. The attention is focused on one hand on the construction of efficient cooperative caching architectures (specially inter-cache communication protocols), and on the other hand, on the creation of efficient document validation/replication mechanisms. Table 1 shows several caching researches, some of them present advantages and drawbacks in creating cooperative caching architectures and some others show advantages and drawbacks in using certain document validation/replication mechanisms. This work is focused on verifying whether there is a considerable influence (in response time, bandwidth consumption, document consistency) when we combined some cooperative caching architectures with some document validation/replication mechanisms.

### DESCRIPTION OF THE COMPARISON PROCESS

Creating a system which compares different cooperative caching architectures combined with several document

validation/replication methods is not a trivial work. Generating a mathematic model which take into account all possible variables in a process like this becomes intractable. For those reasons we have created a set of simulations which include different typical cooperative caching architectures combined with some well known document validation/replication methods, creating some common scenarios.

**Table 1.** Several researches in cooperative caching

Archit.	Validation/replication mechanisms	Main research reference	Some results obtained
Hier.	Pulling-TTLE	[15]	- Distribution of workload is better using hierarchies - Hierarchies with 3 levels have better performances than hierarchies with more or less levels.
Hier.	Pulling-TTLA	[14]	- with Adaptive TTL we can get optimal bandwidth consumption and response time. They suggest TTLA protocol (a protocol derived from TTL-Alex) - It is better to use TTLE when we know the document expiration time.
Hier.	IMC	[12][18]	- a good mechanism to guarantee consistency in document frequently modified is using invalidations
Hier	IMC, IMCP, pulling-TTLN, pulling-TTLO	[4]	- IMC and IMCP algorithms show the best performance in documents with high frequency of reads and writes.
Hier. & Dist.	It is not specified	[1][7][9]	- A distributed architecture has better performance than a hierarchy.
Hier., Dist. & Hybrid	It is not specified	[26]	- The best performance is obtained using a hybrid architecture. Distributed caches have less connection time than hierarchies. - Distributed architectures use more bandwidth than hierarchical architectures.
Dist.	It is not specified	[25][17]	- better performance when clients access directly to caches without using hierarchies.
Dist.	Asynchronous and synchronous Pushing	[20][3]	- As additional proposal in client-initiated caching. Better distribution of popular documents. - If we create interest channels we will get better document distribution.
Hier., Dist. & Hybrid	Pulling-TTLO, Pulling-TTLN,IMC, IMCP	This paper	-see results and conclusions sections.

The objective of the simulations will be to evaluate the performance of each cooperative caching configuration when it is working with a particular document validation method. We observe details as client-perceived response time, bandwidth consumption, document staleness (period of time that a document has been inconsistent, and how

many documents), and a speedup which shows the performance improvement compared to a configuration with no caches.

### Simulation scenarios

The simulations set have been created on top of NS (Network Simulator) [22]. We have made appropriate changes to NS which allow us to get our objectives. The simulation scenarios will be constructed based on a combination of two parts: the description of the cooperative caching architectures that will be used, and the validation mechanisms that will be used in each architecture. Table 2 and 3 show these combinations.

**Table 2.** Cooperative Caching Architectures.

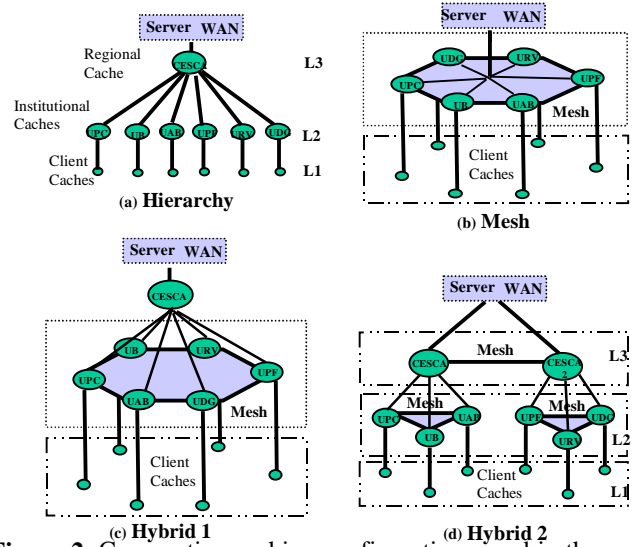
Cooperative Architecture	Caching	Description
(J) Hierarchy		A hierarchy of caches
(D) Distributed (Mesh)		Caches connected in a distributed mesh
(H1) Hybrid 1		Clients are connected to a mesh and at the same time that mesh is connected to a hierarchy
(H2) Hybrid 2		The same as H1 but with less caches in each hierarchy branch.

**Table 3.** Mechanisms of document validation/replication.

Document Validation/Replication Mechanism and descriptions
<b>TTLA0:</b> Mechanism of validation which sends a IMS message (unicast) any time a document in the cache is requested (threshold=0). TTL0.
<b>TTLA:</b> Similar to TTL0 but an IMS message will be sent only if the threshold is passed. The threshold for this experiments is 10% of the life of the document (threshold=0.10). Parameter used in [14].
<b>M:</b> Uses multicast invalidation mechanism, similar to mechanism described in [4]. In this configuration an invalidation message is sent by the server each 60 seconds. If a server document changes during the invalidation period (60 segs.), when caches receive the next invalidation message, they will invalidate that document. If the server document has not changed then when caches receive the next invalidation message they will do nothing. That message is used as a server vital signal. If caches do not receive any vital signal after 5 times the invalidation period (60*5=300 segs for this experiment), caches will invalidate all the documents they have received from that server.
<b>PSM:</b> Uses multicast invalidation combined with pushing mechanisms. A heuristic is used to decide if a document is sufficiently popular to be pushed. When this decision is taken, all the next invalidation messages will include the document if it has changed.
<b>APM:</b> Similar to PSM but pushing is activated when a client sends a pushing request for a document to the server (for this experiment clients always send a document pushing request to server, - like mirrors -)

It is possible to include more combinations of cooperative caching architectures with validation methods, however in this work we have chosen some well known and widely used caching architectures and validation methods. They cover many configurations installed today. As a reference parameter we use the cooperative caching system installed in the Spanish Academic Network located in Catalonia, Spain. It will be a realistic scenario where the results obtained in this work will be applied. In this Academic Network are the following universities, UPC, UB,UAB,

URV, UPF and UDG linked by the Super Computing Center of Catalonia (CESCA) backbone. We have defined 4 possible cooperative caching architectures which represent common cooperation scenarios (figure 2). In figure 2 client caches are represented by small circles connected to institutional caches (caches in the universities), however those small circles are grouping all the clients accessing the Web through institutional caches.



**Figure 2.** Cooperative caching configurations used in these simulations

The number of clients could be hundreds. We have run several simulations using different types of links in order to observe the impact of each configuration using particular link characteristics. Results presented in this paper are based on more general link characteristics because in this way, results could be used for more people using similar configurations. Links between institutional caches (L2) and client caches (L1) have 10Mb bandwidth and 2ms delay.

**Table 4.** Simulation Scenarios

Identifiers	Description
JTTLA0, JTTLA, JM, JPSPM, JAPM	Hierarchical cooperation (J) using the following document validation/replication mechanism: TTLA0, TTTLA, M, PSM, APM
DTTLA0, DTTLA, DM, DPSPM, DAPM	Distributed cooperation (D) using the following document validation/replication mechanism: TTTLA0, TTTLA, M, PSM, APM
H1TTTLA0, H1TTTLA, H1M, H1PSM, H1APM	Hybrid cooperation (H1-figure 2c-) using the following document validation/replication mechanism: TTTLA0, TTTLA, M, PSM, APM
H2TTTLA0, H2TTTLA, H2M, H2PSM, H2APM	Another hybrid cooperation (H2-figura 2d-) using the following document validation/replication mechanism: TTTLA0, TTTLA, M, PSM, APM

Links between institutional caches (L2) and regional/national caches (L3) have 1.5Mb bandwidth and 50ms delay, the same is for links between L2 or L3 and Servers. Packets HTTP (like GET, IMS, etc) are 43 bytes, and invalidation messages are 32 bytes. Every document validation/replication mechanism can be tuned in order to improve performance, however values for setting parameters in these configurations are based on typical values mentioned in [4][14]. All the scenarios are summarized in table 4. Hybrid configuration 2 (H2) in figure 2d is defined because we try to analyze the performance impact between having a few more or less caches in the hierarchy nodes.

### Workloads

Simulations in this work were made using two types of workloads, the first of them was generated requests of synthetic way, following Poisson distribution. Second, it was using the registered log files (traces) in each cache controlled by CESCO. In this paper we presented the results based on the trace files that the CESCO has provided us, since these show real service loads, which causes that the results are more reliable. However the results that we have obtained using synthetic loads do not vary significantly compared with CESCO trace file characteristics, which does that the conclusions obtained with the real traces apply in the synthetic ones. The trace description appears in table 5.

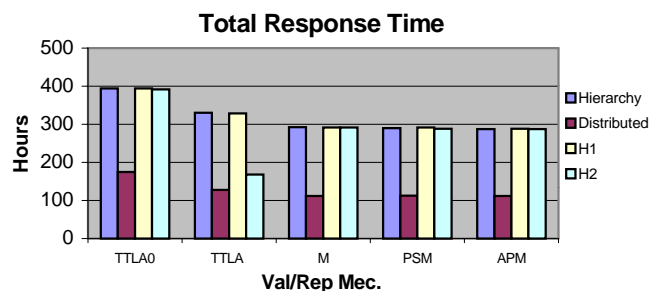
**Table 5.** Workload used in the simulations

Number of requests	3,089,592
Number of objects	212,352
Number of clients	11,765
Average of requests by second	21
Bytes transferred	30GB
Duration	2 days

In each simulation, caches started their work with a warming process which prepared them for not to start empty. The cache warming process was done using requests occurring in day 1, and the beginning of the calculation of the data began with requests made in day 2. It has been verified that a cache warming process gives simulations with more trustworthy results [1]. We did not have information about the frequency of changes or updates of documents. We have used a frequency of changes that is based on a bimodal function [4] that indicates that a part of the documents in the trace are more active documents (they change with more frequency) and the other part is more static (they change with little frequency). On the basis of heuristics we have defined that 10% of documents in the trace are active, and the rest is static. The age of active documents on average was defined as 1 hour, following the results found in [24], and for the static documents as one week.

## RESULTS

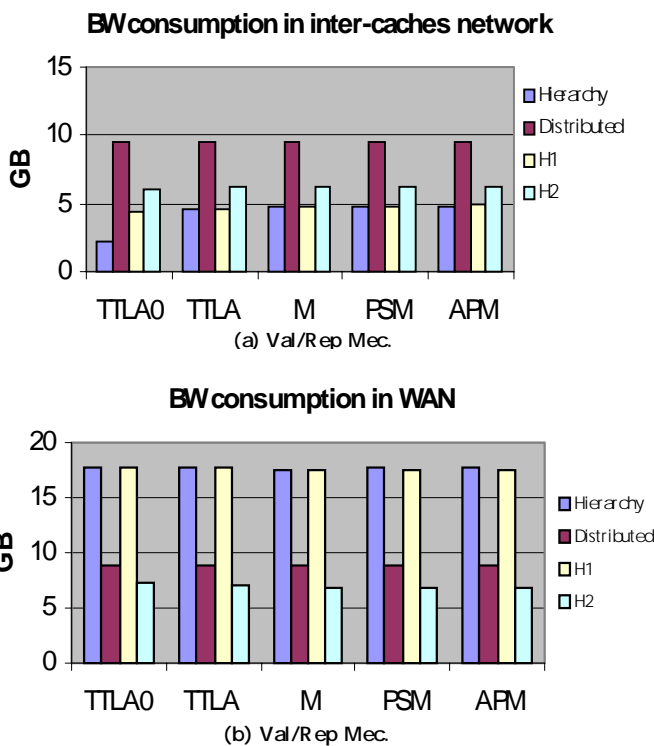
The first results obtained in this work show the effect that each document validation/replication (val/rep) mechanism had with the different cooperative caching architectures. The points to observe are the response times perceived by the client, bandwidth consumption both at level of the network that interconnects to the cooperative system of caching (inter-cache) and in the network that allows the connection with the Internet (WAN). Also we observed the behavior of the cooperative configurations using the val/rep mechanisms with their typical parameter values, obtaining the number of stale documents that each mechanism generates. Figure 3 shows how the val/rep mechanism affects each cooperative caching architecture as to the sum of client-perceived response times during the reproduction of the trace of table 5. We can see that using the settings we have defined in each val/rep mechanism, the distributed architecture is the one that offers the best performance as to the total response time for dispatch of all requests in the trace file. In a perspective of which are the best cooperative caching architecture we can see that distributed architectures are the best option, and the influence caused by the val/rep mechanism is not enough to degrade its performance, it means that using whatever val/rep mechanism, distributed architectures seem to be the best option. If you are looking



**Figure 3.** Total of response time perceived by clients

comparing the val/rep mechanisms we can see TTLA0 has the worst performance in all types of cooperative caching architectures. It is understood that if we have more interest in performance TTLA0 is not a good option. The val/rep mechanism with better alternatives seems to be APM. It does not matter what cooperative caching infrastructure we use. APM has the best response time in all cooperative caching architectures (in some of them only slightly so). However the rest of mechanisms have a similar performance, so it could be interesting to evaluate bandwidth consumption in order to have a broad point of view to decide to construct a caching system. In figure 4, we can see the bandwidth (BW) consumption for each rep/val mechanism. Bandwidth consumption is shown under two points of view. Figure 4a shows bandwidth consumption inside of the inter-cache network. That is,

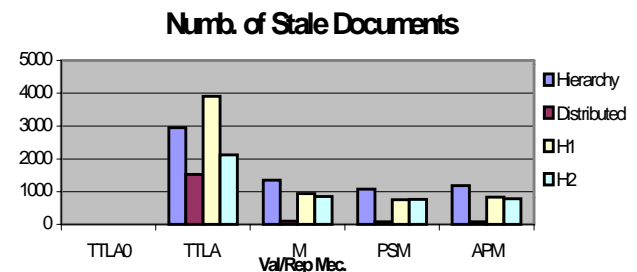
traffic generated only in links connecting the cooperative caching system. Figure 4b shows traffic generated to the wide area network (WAN). That is, traffic which goes to servers outside the inter-cache network. It is important to see both traffics in a separate way because we can see where the bottlenecks are (if any), and to be aware if our HTTP traffic is yielding some problems to another type of traffic inside and outside the inter-cache network. Distributed architectures have the highest inter-cache bandwidth consumption. Conversely hierarchies have the lowest inter-cache bandwidth consumption independently of which val/rep mechanism is being used. Figure 4a shows that TTLA0 is the val/rep mechanism which consume less inter-cache traffic. TTLA0 has a strong dependence of web servers outside inter-cache network, because it validates every request that caches receive for a stored document.



**Figure 4.** Bandwidth consumption (BW) inside and outside of the inter-cache network

Do not forget that TTLA0 had the worst client-perceived response time (fig 3). Figure 4b shows how H2 architecture has the lowest use of WAN bandwidth followed by distributed architecture independently of which val/rep mechanism is used. Cooperative caching architectures which more affect the WAN bandwidth are hierarchies and H1. WAN traffic is important because it could be the reason for variability in client-perceived response time. If we generate less impact in WAN traffic (it is considered that

connections in the inter-cache network have better quality of service than connections in the WAN) is possible to prevent response time peaks, preventing clients from perceiving pathological response time. The main function of the val/rep mechanisms is to maintain document consistency. That is, to try that the documents version that will be given to the clients by caches are the most “fresh” possible. Depending on settings in each val/rep mechanism, they could obtain greater or smaller degree of consistency in documents. In figure 5 we show the number of stale documents that arrived to the clients. As we can see, if our interest is to obtain strong consistency, the mechanism that offers that guarantee to us in any type of architecture is the TTLA0 mechanism. The following mechanism that offers better results is the mechanism of selective multicast invalidation with pushing (PSM).



**Figure 5.** Number of stale documents which were received by clients.

In any case we notice that the mechanism that offers worse results in consistency is the TTLA. We have seen that if we decrease the age average of documents (we have used 10% of the documents age in this experiment) improves the TTLA mechanism remarkably, but as well generates a greater number of verification messages and that means a higher bandwidth consumption (that graph was not included here). Many researches into cache have focused the work on evaluating and comparing two main aspects, on one hand, several cooperating caching architectures (hit ratios, response time), and on the other hand the val/rep mechanisms (number of stale documents, document freshness). Normally the comparison parameter is to see how good (cooperative architecture: response time, val/rep mechanisms: staleness) is one with respect to others. Nevertheless, it is difficult to know the degree of the benefit (if any) that is produced by the cooperative caching architecture and its val/rep mechanism compared with a system that does not use caches, taking as comparison metric the sum of the response times obtained reproducing the trace file in each one of the systems. Figure 6 shows the degree of gain (speedup) obtained by each cooperative architecture and its respective val/rep mechanism. We have called the architecture that does not use caches “Proxies No Caches” (PNC). A speedup equal to 1 indicates that the

cooperative caching architecture and the validation mechanism that it use is not generating gains. Speedups greater than 1 mean the cooperative caching architecture obtains some gains.

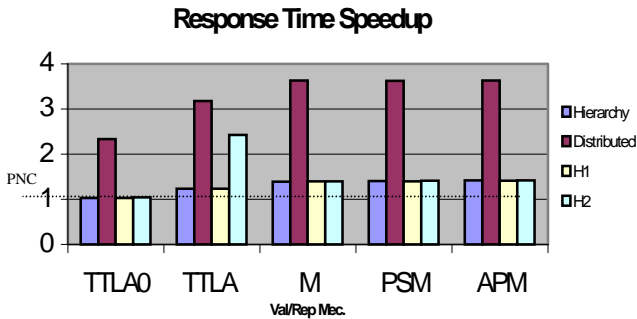


Figure 6. Response time speedup of different configurations

As we see in figure 6 the architecture that gives the most benefits (a speedup up to 2.5 more than PNC) is the distributed one, without mattering what rep/val mechanisms have been used. We can see that the hierarchical, H1 and H2 architectures using TTLA0 validation offer almost a null benefit. This means a drawback because of the cooperative caching administration costs that they are generating. With the purpose of offering a clearer overall state of the cost/benefit of a cooperative caching architecture and its corresponding mechanisms of inv/rep, we have analyzed speedup versus bandwidth consumption (total Inter-cache plus WAN). In figure 7 all the architectures reviewed in this work appear contrasting speedups with bandwidth consumption that they generate. The proximity to the ideal mark represents greater benefit with smaller cost. DAPM, DTTLA and H2M configurations are the best alternatives when the interest is focused on the network infrastructure.

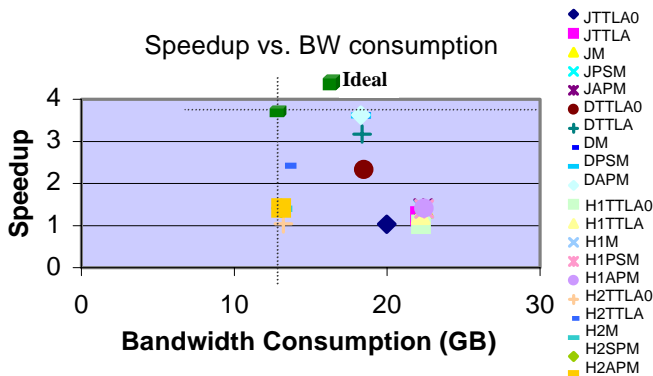


Figure 7. Speedup vs. Total Bandwidth Consumption

If our interest is guarantee a fresh document, the same process using speedup with the number of stale documents received by clients (figure not included) was compared. The values near the ideal mark represent greater efficiency

with less inconsistency in documents. Although the TTL mechanism of verification with threshold equal to 0 (TTL0) does not experience stale documents, the mechanisms of invalidation DAPM and DSPM offer a better performance. The final decision depends on what is more important for us. Invalidation mechanisms can have a pathological effect which depends on the period of time that is chosen to emit the invalidation message. The effect can occur when document updates happen at the same time as the period of invalidation. This causes that the invalidation message always arrive late and therefore the number of stale document increases.

## CONCLUSIONS AND FUTURE WORK

Web caching has been a relevant issue during the last years because it has shown to improve Web access infrastructure. Much of the work done to discover a better cooperative caching architecture is individually centered on one of the following aspects: the mechanisms of caching cooperation or the validation/replication mechanisms. In this work we wanted to show that both aspects have influences on each other, and that we cannot assert that one cooperative caching architecture such as hierarchic, distributed or hybrid, is better or worse than other, without considering the validation/replication mechanism that will be used. The values assigned to each parameter in each val/rep mechanism can generate different results and to get another architecture like the best one. However our intention in this work was not to say if a val/rep mechanism (TTL, Invalidation, etc.), or a distribution architecture (Hierarchy, distributed, etc.) was better than others, rather to stand out that when thinking about the construction of a cooperative caching infrastructure we must consider the influence that the val/rep mechanism could have on different cooperative caching architectures and see what infrastructure alternatives are worth taking into account to construct the cache system. Our simulator is an useful and reliable tool for a priori evaluation of possible combinations and is a good contribution in this paper. We will continue working on our simulator trying to include more val/rep mechanisms to give a wide spectrum for evaluation. We will include mirrors in our simulator. In that way, we can combine mirrors and caches. Mirrors need a certain grade of consistency, that means a different treatment and communication between mirrors and caches, which is an interesting issue to include in our simulator.

## REFERENCE

- [1] S.G. Dykes, C.L., K.A. Robbins, and C.L. Jeffery, "A Viability Analysis of Cooperative Proxy Caching", in Proc. of the IEEE Infocom 2001
- [2] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area Web cache sharing

- protocol”, in ACM SIGCOMM '98, Vancouver, Canada, 1998, pp. 254-265
- [3] J. Gwertzman, and M. Seltzer. "The case for geographical push caching". Fifth Annual Workshop on Hot Operating Systems. 1995
- [4] Haobo Yu, Lee Breslau, and Scott Shenker, "A Scalable Web Cache Consistency Architecture". SIGCOMM99. volume 29, number 4, October 1999. <http://www.acm.org/sigs/sigcomm/sigcomm99/papers/session5-1.html>
- [5] Rousskov A. and D. Wessels, "Cache digests", Computer Networks and ISDN Systems, Vol. 30, no. 22-23, pp. 2155-2168, Nov. 1998
- [6] Squid Internet Object Cache, "Available on line at <http://squid.nlanr.net/>
- [7] R. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay, "Design considerations for distributed caching on the internet", in Proc. of the Int'l. Conf. On Distributed Computing Systems (ICDS'99).
- [8] National Laboratory for Applied Network Research (NLANR), "Ircache project", available on line at <http://ircache.nlanr.net/>
- [9] Wolman A., G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the scale and performance of cooperative Web Proxy caching", in Proc. of the 17<sup>th</sup> ACM Symp. On Operating Systems Principles, Dec. 1999
- [10] Red Académica Española, Red Iris. <http://www.rediris.es>
- [11] Korea National Cache, <http://cache.kaist.ac.kr>
- [12] Cao, P. And Liu C., "Maintaining strong cache consistency in the World Wide Web", In Proc. of the Int'l. Conference on Distributing Computing Systems (May 1997), pp. 12-21
- [13] N. G. Smith, "The UK national Web Cache – The state of the art", Computer Networks and ISDN System, 28:1407-1414, 1996
- [14] Gwertzman, J. and Seltzer, M. "World Wide Web Cache Consistency". In USENIX Conference Proceedings (Copper Mountain Resort, CO, USA, Dec. 1996) pp. 141-151
- [15] Chankhunthod A., P.B. Danzing, C. Neerdaels, M.F. Schwartz, and K. J. Worrell. "A hierarchical internet object cache". In Proc. of the 1996 USENIX Technical Conf. Pags. 153-163. January 1996.
- [16] D. Karger, A. Sherman, A. Berkhemier, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web Caching with Consistent Hashing", Eighth International World Wide Web Conference, May 1999.
- [17] V. Vallopillil and K. W. Ross, "Cache Array Routing Protocol v1.1" Internet Draft, Feb. 1998, available on line at: <http://ds1.internic.net/internet-drafts/draft-vinod-carp-v1-03.txt>.
- [18] L. Zhang, S. Floyd, and V. Jacobson, "Adaptive Web Caching", in Proc. of 2<sup>nd</sup> Web Cache Workshop, Boulder, CO, June 1997
- [19] Rodriguez, P. Biersack, E. W., "Continuous multicast distribution of Web documents over the internet", IEEE Network Magazine (March-April 1998)
- [20] Touch, J. "The LSAM proxy cache – a multicast distributed virtual cache". In Proc. of the Third Int'l. WWW Caching Workshop (June 1998).
- [21] Rodriguez, P. Biersack, E. W., and Ross, K.W. "Improving the WWW: Caching or multicast", In Proc. of the third Int'l. WWW Caching Workshop (June 1998)
- [22] Virtual InterNetworks Testbed, VINT Project, University of Southern California, NS Web Site: <http://mash.cs.berkeley.edu/ns>
- [23] D. Wessels and K. Claffy, "Application of Internet Cache Protocol (ICP), version 2", Internet Draft: draft-wessels-icp-v2-appl-00. Work in progress, IETF, May 1997.
- [24] Douglas F., Feldmann A., Krishnamurthy B. and Mogul J. "Rate of change and other metrics: a live study of the world wide web", Tech. Rep. 97.24.2, AT&T Labs, Dec. 1997. A shorter version appeared in Proc. Of the 1<sup>st</sup> USENIX Symposium on Internet Technologies and Systems.
- [25] D. Povey and J. Harrison, "A Distributed Internet Cache", In Proceedings of the 20<sup>th</sup> Australian Computer Science Conference, Sydney, Australia, February 1997.
- [26] Rodriguez, P., Spanner C., and Biersack, E. W., "Web Caching Architectures: Hierarchical and Distributed Caching". 4<sup>th</sup> International Web Caching Workshop, San Diego, USA. 31<sup>st</sup> March – 2<sup>nd</sup> April, 1999.
- [27] TERENA. WWW cache coordination for Europe. Available online at <http://www.terena.nl/task-forces/tf-cache/>
- [28] A. Dingle and T. Partl. "Web Cache Coherence". Fifth International World Wide Web Conference, 1996
- [29] I. Melve. "Inter-cache communication protocols". IETF WREC Working Group Draft, 1999.