

Two-Tier Cooperation: A Scalable Protocol for Web Cache Sharing

Andrea Santoro, Bruno Ciciani*
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
email {santoro, ciciani}@dis.uniroma1.it

Michele Colajanni
Dipartimento di Scienze dell'Ingegneria
Università di Modena
Via Campi, 41100 Modena, Italy
email colajanni@unimo.it

Francesco Quaglia
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
email quaglia@dis.uniroma1.it

Abstract

The benefits of Web caching can be improved by systems of cooperative cache servers that share their cached documents. The increasing number of Web cache servers over the Internet makes the scalability of the cooperation protocol a major issue to be addressed. In this paper we propose a Two-Tier Cooperation protocol (2TC), which is specifically designed for systems of dozens or hundreds of cache servers with no centralized control. 2TC embeds two classical cooperation approaches for distributed Web caching systems, namely Informed Cooperation (IC) and Query Cooperation (QC), that are applied within different subsets of cache servers in the system. IC is applied within subsets of close servers and lets them cooperate through mutual exchange of state information related to their cache content. QC lets more distant cache servers cooperate through query/reply messages to locate documents within the global cache. Thanks to the use of IC among close cache servers, QC can explore the cache content of several cache servers through a single query message. High scalability arises as few queries explore the cache content of many cache servers and state information is exchanged within small groups of close cache servers. We report experimental results based on real traces that compare a prototype implementation of 2TC with classical protocols of the informed and query classes. The results point out a strong reduction (up to 50%) of the amount of transferred information to manage cooperation. This overhead reduction is achieved with no performance degradation in terms of latency and cache hit rate.

* Bruno Ciciani is currently Visiting Professor to the IBM Thomas J. Watson Research Center, NY.

1 Introduction

The growing popularity of the World Wide Web causes Internet traffic increase and congestion. A way to deal with this issue consists of maintaining copies of Web documents at cache servers located overall the network. Whenever a client needs a Web document, it sends an HTTP request to a close cache server. The request is forwarded to the Web server only if the cache server does not cache a valid copy of the document. Otherwise, the cached copy is sent immediately to the client. This solution limits both client-server traffic and load on Web servers, thus reducing the latency of document retrieval perceived by the clients. Many results show that the document hit rate in single cache server systems could be low. Therefore, many authors have proposed systems with multiple cooperative cache servers where some server can serve cache misses of others. Any form of cooperation requires some exchange of information among cache servers to be settled. To achieve this, there are two classical, opposite solutions:

Query Protocol (QP). Upon a local cache miss, a cache server broadcasts a query message to locate the document within the global cache. This solution was proposed by Harvest project [12], and then implemented in NetCache [11] and Squid [14]. Both of them use the Internet Cache Protocol (ICP) as the query mechanism [17].

Informed Protocol (INF). Each cache server frequently informs the others about changes in its set of cached documents. In this way, documents can be located within the global cache without the need for query

mechanisms.

Relaxations of pure INF require cache information to be exchanged infrequently. These solutions reduce the overhead of continuous message exchange at the expense of weakening information consistency. Some of them, e.g. Summary Cache [5], re-introduce a query mechanism to confirm document location within the global cache.

Cooperation can be established along a vertical direction, namely *hierarchical Web caching* descending from the Harvest project [12], or along a horizontal direction, namely *distributed Web caching* [8]. Various studies have shown that pure hierarchical caches have scalability and coverage problems, especially for large sets of cooperative cache servers [5, 7, 15]. This paper focuses on distributed Web caching and aims at improving the scalability of both INF and QP. We recall INF exhibits poor scalability as strong information consistency is impracticable for large global caches. Indeed, the overhead for consistency maintenance could easily overcome the profit of cooperation. Analogously, the query mechanism underlying QP could reveal ineffective for large global caches because of two reasons. The overhead of query broadcasts becomes too expensive. It is likely that the sender cache server does not receive many replies within the time-out expiration because of the large distance among cache servers and possible network congestion. One of the causes of limited scalability of QP and INF is that they enforce the same cooperation method between any pair of cache servers. To address the scalability issue, we propose a Two-Tier Cooperation protocol (2TC) where cache servers are grouped into clusters (not necessarily partitioned), and where QP and INF are combined in the following way:

- INF is used as the intra-cluster protocol, thus avoiding the query mechanism among close cache servers. This contributes to very fast document delivery within the cluster at the price of a limited overhead when the cluster has reasonably small size. Basically, 2TC guarantees that each cache server is informed about the cache content of few close cache servers.
- QP is used as the inter-cluster protocol. In this way, thanks to the presence of the intra-cluster protocol INF, a single query message allows the sender to explore the cache content of a whole remote cluster. To reduce the risk of false misses due to replies that do not arrive within the time-out expiration, each cache server typically sends query messages only to “easily reachable” remote peers¹.

¹Although this set can change dynamically due to network conditions, in this paper we consider the instance of statically chosen query destinations.

High scalability of 2TC arises since INF operates within clusters that are small with respect to the size of the global cache. Hence, up to date information is guaranteed with low overhead. Furthermore, any query allows the sender cache server to access a large amount of information related to the location of documents within the global cache. We have implemented a prototype of 2TC that exhibits a strong reduction (up to 50%) of the amount of information transferred among the cache servers to manage cooperation as compared to classical INF and QP. This is achieved with no decrease of global hit ratio and with no relevant increase of the latency perceived by the clients as compared to the best results of INF and QP. This result points out that 2TC can be used to manage document sharing in very large global caches. Although the experiments reported in this paper are limited in the number of cache servers, other preliminary experimental results and simulations lead us to think the benefits of 2TC for larger configurations are even better than the results achievable by INF and QP alone. The remainder of the paper is organized as follows. In Section 2, we present an overview of existing cooperation protocols and discuss scalability issues. In Section 3, we describe the 2TC protocol. In Section 4, we report experimental results.

2 Related Work

We focus on protocols where cooperation is finalized to the discovery and delivery of documents, while we do not consider protocols that use cooperation for document insertion and replacement in the global cache.

2.1 Query Protocols

In the context of the Harvest project [3, 12], discovery and delivery of documents is done through the *Internet Cache Protocol* (ICP), which is based on UDP query/reply messages [17]. When a cache server experiences a local cache miss, it sends a query message to all the peers in order to discover whether one of them caches a valid copy of the requested document. In the positive case, the recipient cache server replies with a hit message, otherwise it replies with a miss message. The ICP protocol is used by the popular proxy server Squid [14] and commercial Net-Cache [11]. The main drawback of ICP is that any discovery phase requires queries to all the peers, therefore the overhead quickly becomes prohibitive as the number of cache servers grows. In [5] it has been shown that this overhead may become unacceptable even with less than 10 cooperative cache servers. Another problem is that UDP does not guarantee reliability of query/reply deliveries, hence a time-out, typically set to 2 seconds, is used by the sender cache server to decide when to redirect the HTTP request to the Web server. When the size of the global cache grows, the

probability of UDP packet loss grows as well, especially during the busiest hours originating traffic peaks on the network. This increases the probability of time-out expiration, especially for the case of global misses. The final effect is an increase of the average latency experienced by the clients.

2.2 Informed Protocols

In the *Summary Cache* proposal [5], each cache server keeps a “summary” of the URLs of documents cached by its peers. The summaries may be inaccurate because of possibly infrequent notifications of the cache content to the peers, and because summaries are not complete but represented as “bloom filters” (this is done to keep low the amount of notification information). Upon a local cache miss, the cache server checks for potential remote hits and sends queries only to promising peers to confirm the document location within the global cache. The number of queries per local miss is typically reduced as compared to ICP, thus allowing Summary Cache to scale better than ICP. Moreover, information on local cache content is piggybacked on query/reply messages. The key issue for the effectiveness of Summary Cache is the accuracy of the summaries. If the summary does not indicate that a requested document is cached at some peer, a potential cache hit is lost, thus reducing the global hit ratio. In addition, as no query is sent to that peer, state information about locally cached documents does not flood. Keeping low the probability of lost hits could require relatively frequent exchange of notification messages, which could increase too much the cooperation overhead for the case of very large numbers of cache servers.

Informed cooperation protocols avoiding query/reply messages have been proposed by Rabinovich, Chase and Gadde. In [13] the authors propose a solution where information about the content of the global cache is kept internally to any cache server. Similarly to Summary Cache, this protocol requires each cache server to inform its peers about the locally cached documents. Upon a local miss, the cache server checks the directory of URLs cached by its peers for finding potential remote hits. In the positive case, unlike Summary Cache, no query message is sent. Instead a TCP connection is established for downloading the document through HTTP. If the remote hit is false, then the penalty paid by this protocol is usually higher than that of Summary Cache because TCP connections induce more overhead and latency as compared to UDP query/replies. On the other hand, if no false remote hit arises, then the Rabinovich et al. protocol allows faster download of the document, since no query mechanism is activated at all. To make this protocol really effective, each cache server has to inform frequently the peers about its cache content. However,

this solution moves the Rabinovich et al. protocol towards the pure INF approach. This may dramatically increase the overhead when the set of cooperative cache servers becomes large. Actually, this protocol has not been designed to manage global caches of large size. Instead, its main task is to allow very fast document download within small groups of close cache servers.

A solution similar to the Rabinovich et al. protocol has been proposed in [4]. The main difference is that when a remote cache miss occurs, the remote peer provides a (possibly empty) list of potential owners of the requested document.

Other cooperation protocols rely on the use of an external global directory manager. As an example, in the *Directory Server* protocol there is a directory server that keeps track of documents cached within the global cache and serves queries for cache hits [6]. The main advantage is that each cache server must notify its cache content only to the directory server. On the other hand, the directory server may become a bottleneck of the system. Another major issue is where to locate the server in order to guarantee easy access to all the cache servers. These problems could be solved by replicating the directory server. However, this choice is likely to originate large overhead for maintaining consistency of the replicas of the global directory. The authors do not exclude the possibility of hybrid solutions in which small portions of the global directory are partially replicated at some cache servers in order to reduce the number of queries to the external server. CRISP caches [8] are examples of these architectures.

The *Cache Array Routing* protocol is a rather different proposal [16] that aims at partitioning the URL space among cooperative cache servers. However, this approach may result ineffective for wide-area caching where the network bandwidth is limited and the distances between cache servers are not uniform.

2.3 General Comments

In previous solutions, with the exception of the one in [4], no cache server acts as a directory server for the peers. Hence, the state information each cache server knows about the content of other caches is never used for serving cache misses of other cache servers. As a consequence, a query message towards a cache server explores only the cache content of the recipient server. This points out that state information known by a cache server is not fully exploited to improve the discovery phase. On the other hand, in [4] no exploration queries are used, therefore high global hit ratio can be provided only by frequent notification of the cache content among the peers. This choice is likely to originate high overhead in large configurations of cache servers. The 2TC protocol we propose uses an informed approach among

close cache servers and also a query mechanism among more distant cache servers. Moreover, it allows any cache server to act as a directory server for the peers. High scalability should be guaranteed by the fact that 2TC aims at exploiting properties of informed and query based solutions by applying them to the most adapt subsets of cache servers in the global cache.

3 Two-Tier Cooperation Protocol

Let us consider a set S of n cache servers $\{P_1, \dots, P_n\}$. We associate with each pair of cache servers (P_i, P_j) a distance measure $\Delta(P_i, P_j)$ which represents the average data transfer cost between P_i and P_j . This measure can be based on various metrics, such as number of hops, latency, bandwidth, leasing cost or a combination of these. Moreover, it can be evaluated statically or updated depending on current network conditions. In this paper, we focus on the static case. Let C_i be the subset of S containing any cache server $P_j \in S$ such that $\Delta(P_i, P_j) \leq \gamma_i$. Hence, each cache server P_i is associated with the cluster C_i containing all the cache servers that have a distance from P_i less than or equal to γ_i . The value of γ_i represents the threshold distance for the determination of C_i . If $\gamma_i = 0$, then C_i contains only P_i . If $\gamma_i = \infty$, then C_i coincides with the entire set of cache servers S .

3.1 Informed Cooperation

The informed cooperation mode (IC) of the 2TC protocol is used within each cluster C_i . Specifically, each cache server $P_j \in C_i$ periodically notifies to P_i its cache content. P_i , in its turn, maintains a local directory of documents cached in the cache servers belonging to C_i . The directory is updated every time a notification message is received. Therefore, P_i is semi-informed about the cache content of the cache servers belonging to C_i . We use the term “semi” due to possibly infrequent notification, which may originate inaccuracy in the local directory of P_i . Another factor that could impact the accuracy of the local directory is the degree of reliability of the transmission protocol for notification messages. On the basis of information that IC provides to P_i , we can distinguish three types of client requests:

Local-hit. The document requested by the client is in the cache of P_i .

Potential-cluster-hit. The document requested by the client is not in the cache of P_i , but the local directory indicates it is cached by some cache servers belonging to C_i . In this instance, the directory content indicates a remote hit in C_i . This hit is potential due to possible inaccuracy of the local directory content. Specifically,

there exists the risk that the remote cache server does not currently cache the document even if P_i has not been informed yet.

Potential-cluster-miss. The document requested by the client is not in the cache of P_i and the local directory indicates it is cached by no cache server belonging to C_i . Analogously to previous case, the miss in C_i is potential because some cache servers in C_i could cache the document even if this information has not yet reached P_i .

When a local-hit occurs, P_i immediately forwards the document to the client and no data exchange among cache servers is activated at all. In case of a potential-cluster-hit, P_i selects one of the cache server that its local directory indicates as a potential holder of the document and attempts to fetch the document from that cache server. The selection policy should reduce the probability of false remote hit due to document invalidation [2]. In our prototype we have limited this problem by maintaining, together with location information, also Time-To-Live (TTL) information related to cached documents. Hence, we can select the remote cache server the local directory indicates as a holder of a copy of the document with not-expired TTL. In case of a remote hit on the selected cache server, the document is forwarded to the client. In case of a remote miss, the Web server is contacted. Potential-cluster-misses are managed by activating the second cooperation mode of 2TC as we shall discuss in the following section.

3.2 Query Cooperation

The query cooperation mode (QC) is activated by P_i when experiencing potential-cluster-misses. The goal is to exploit information contained in the local directories of the peers, which is collected through IC. QC works as follows. P_i selects a subset Q_i of S and sends a query message to the cache servers in Q_i . The subset Q_i will be referred to as the *query set* of P_i . The payload of the query is the URL of the requested document. Upon the receipt of the query, any recipient cache server P_k checks the content of its cache. If the requested document is in the cache, P_k replies with a hit message whose payload is its own Internet address. If the document is not in the cache, P_k checks its local directory to see whether some cache servers in C_k have the document in their cache. In the positive instance, P_k selects one of those cache servers and replies to P_i with the Internet address of the selected cache server. As for the case of potential-cluster-hits, the selection policy we implemented in our prototype is TTL based. This should reduce the probability of remote false hits due to document invalidation [2]. In the negative instance, P_k replies to P_i with a miss message indicating that P_k does not cache a copy of the docu-

ment and, to the best of its knowledge, no cache server in C_k does cache it. If no hit is received by P_i , then the document is fetched from the Web server. If at least one hit is received, then P_i attempts to fetch the document from the cache server specified by the hit message. If a remote miss on that cache server occurs, the Web server is contacted.

Depending on the degree of reliability of the communication protocol for query/reply messages, a time-out mechanism could be used to determine when a potential-cluster-miss should be forwarded to the Web server. Typically, in existing cooperation protocols the discovery phase is implemented at the transport layer through UDP messages. Similarly to ICP, our prototype of 2TC uses UDP and uses a time-out set to 2 seconds.

A challenge of the QC mode is the choice of the query set Q_i associated with P_i . Our solution is as follows. The cache server P_i is associated with a maximum number of queries $max_num_query_i$ and a maximum distance value max_dist_i . Therefore, the objective is the exploration of the cache content of as much cache servers as possible by using up to $max_num_query_i$ queries with the constraint that no query must be sent to a cache server P_k such that $max_dist_i < \Delta(P_i, P_k)$. The choice of the values of $max_num_query_i$ and max_dist_i impacts network and cache server overhead, and consequently latency of the discovery phase. Solutions for the selection of suitable values of $max_num_query_i$ and max_dist_i could be envisaged by using empirical data related to classical ICP. Once set the previous constraints, the choice of the query set Q_i becomes a classical maximum coverage problem, which is a generalization of the set cover problem [9]. The solution can be approximated through a simple greedy algorithm such as the one proposed in Figure 1. Basically, the algorithm adds a new cache server P_k in the query set Q_i if its distance from P_i is less than or equal to the maximum value max_dist_i , and its inclusion in the query set maximizes the number of additional cache servers whose cache content can be explored through the query message.

3.3 Properties of 2TC

The global hit ratio and the latency time of the hits are the two main metrics to evaluate performance of cooperation protocols. 2TC aims at optimizing the system behavior with respect to both metrics. Specifically, 2TC does not penalize hits of high value that is, those associated with very close cache servers, as these hits do not require query/reply exchange. This contributes to keep low the average latency. In addition, 2TC looks for hits in more far cache servers, which contributes to increase the global hit ratio. If cache server P_i selects an “adequately” small value for the parameter max_dist_i that determines the maximum distance for the recipients of a query message, then also these hits can

Choice of Q_i :

```

 $Q_i = \emptyset$ ;
repeat
  <determine the set
     $A_i = \{P_k : (P_k \notin Q_i) \wedge (\Delta(P_i, P_k) \leq max\_dist_i)\}$  >
  if  $A_i = \emptyset$  then break;
  <select  $P_k \in A_i$  such that
     $\cup_{P_h \in Q_i} S_h \cup S_i \cup S_k$  has maximal cardinality>
   $Q_i = Q_i \cup \{P_k\}$ 
until  $|Q_i| < max\_num\_query_i$ 

```

Figure 1. Algorithm for the Choice of the Query Set.

be served in a fast way. This further contributes to limit the latency perceived by the clients.

4 Performance of 2TC

4.1 Evaluation Technique

Our testing environment consists of ten machines: 6 PentiumII 300 MHz (running LINUX), 2 Sparc Stations 10, 1 Sparc Station 20 and 1 Sparc Station Ultra 10 (running Solaris). Each machine hosts one cache server with a cache space equal to 1 GB. The experimental platform emulates a wide area network environment with two groups of 3 close machines, and one group of 4 close machines. Since all the machines belong to the same isolated Ethernet network, wide area network transmission delays are artificially emulated. We consider two scenarios:

- In Scenario A, the average round-trip-time between cache servers running on machines within the same group is set to 40 msec. while it is set to 80 msec. for cache servers belonging to different groups.
- In Scenario B, the average round trip time between cache servers running on machines in distinct groups is set to about 200 msec.

For each cache server P_i , the γ_i value has been fixed at round trip time of about 50 msec. Therefore, we get cache servers running on machines in the same group have completely overlapping C_i clusters that is, they mutually inform each other about their cache content. max_dist_i has been fixed at round trip time 250 msec., and $max_num_query_i$ has been set to 2 for any cache server. The structure of the experimental platform is shown in Figure 2.

The source of the workload used for the experiments consists of 18 days worth of HTTP traces gathered from the

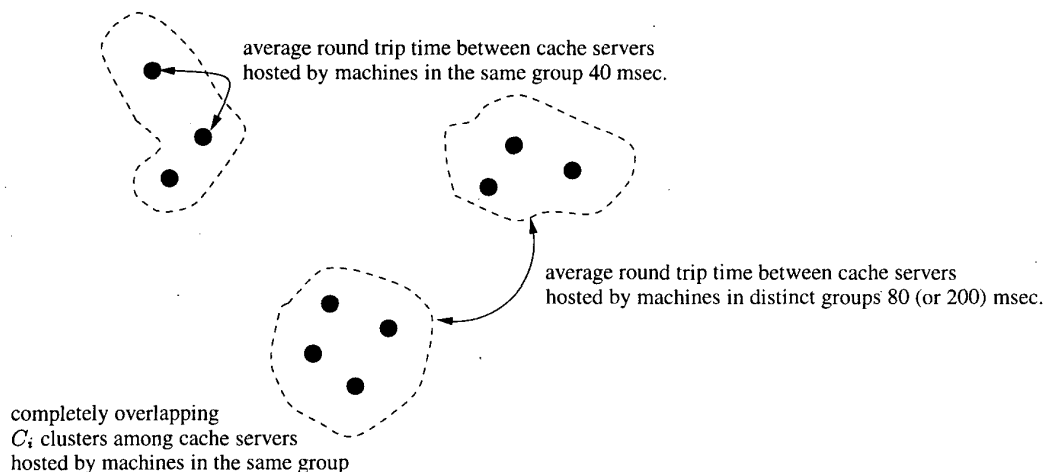


Figure 2. The Experimental Platform.

Home IP service offered by UC Berkeley to its students, faculty, and staff [10]. Home IP provides dial-up PPP/SLIP IP connectivity using 2.4 kb/s, 9.6 kb/s, 14.4 kb/s, or 28.8 kb/s wireline modems, or Metricom Ricochet (approximately 20-30 kb/s) wireless modems. These client traces were unobtrusively gathered through the use of a packet sniffing machine placed at the head-end of the Home IP modem bank. The tracing program used was a custom module written on top of the Internet Protocol Scanning Engine (IPSE) created by Ian Goldberg. Only traffic destined for port 80 was traced; all non-HTTP protocols and HTTP connections for other ports were excluded from these traces. We used the same trace for the setup phase and for the experimental phase. In particular, we used 1.5 million of requests from the trace to setup the cache content of the cache servers. In the setup phase, the destination cache server of any client request is selected uniformly among the 10 cache servers. About 8% of documents cached during the setup phase are replaced through the LRU-MIN policy [1].

In the experiments, we studied the performance of 2TC, QP and INF by using 1 million of other client requests from the same trace to determine the access pattern to the documents. The destination cache server for a request is uniformly selected among all the cache servers. Inter-arrival times between requests are determined on the basis of temporal information maintained in the trace. In our experiments, client processes are executed on a 4 CPUs Super-Sparc belonging to the same Ethernet network of the cache servers. To get comparable results for the different protocols, we have maintained the same request access pattern to

the cache servers. This was done by using the same set of seeds for random number generators in the setup phase and in all the experiments.

In our experiments, QP coincides with ICP (version 2) [17]. For what concerns INF and the IC part of 2TC, we use two time-out values to determine when changes in the cache content must be reported to the peers. Specifically, we study the performance of these protocols under the hypothesis that cache content notification to the peers occurs after 30 (or 100) seconds measured from the occurrence of the first not notified change. These two values allow us to observe system behavior in the case of frequent and less frequent notification. The notification protocol is incremental (only additions-to/deletions-from the cache are notified) and is implemented at the transport layer through the TCP protocol.

4.2 Experimental Results

We measured the following performance parameters:

- Percentage of connections refused by the cache servers due to possible overload.
- Local cache hit ratio.
- Local hit time, which is measured at the cache server contacted by the client.
- Protocol hit ratio that is, the percentage of local misses which are served through documents cached by some peer server.

Cooperation protocol	2TC (notif. timeout 30 sec.)	2TC (notif. timeout 100 sec.)	QP	INF (notif. timeout 30 sec.)	INF (notif. timeout 100 sec.)
Refused requests	0.0%	0.0%	0.0%	0.0%	0.0%
Local hit ratio	20.8%	20.8%	20.8%	20.8%	20.8%
Local hit time (msec.)	33.9	32.7	36.8	35.6	37.0
Protocol hit ratio	8.7%	8.6%	8.7%	8.6%	8.6%
Protocol hit time (msec.)	1888	1890	2041	1752	1778
False hit ratio	0.07%	0.07%	0.01%	0.07%	0.074%
Outgoing bytes per req.	319.16	318.84	781.75	574.48	574.45
Incoming bytes per req.	319.12	318.84	781.72	573.53	574.27

Table 1. Results for the Scenario A (Round Trip Time Equal to 40/80 msec.).

Cooperation protocol	2TC (notif. timeout 30 sec.)	2TC (notif. timeout 100 sec.)	QP	INF (notif. timeout 30 sec.)	INF (notif. timeout 100 sec.)
Refused requests	0.0%	0.0%	0.0%	0.0%	0.0%
Local hit ratio	20.7%	20.6%	20.6%	20.7%	20.6%
Local hit time (msec.)	34.9	35.3	34.8	37.8	37.1
Protocol hit ratio	8.6%	8.5%	8.7%	8.5%	8.5%
Protocol hit time (msec.)	4051	4063	4312	3839	3841
False hit ratio	0.07%	0.07%	0.01%	0.07%	0.08%
Outgoing bytes per req.	318.81	317.34	781.90	575.20	573.12
Incoming bytes per req.	318.73	318.10	781.87	574.65	573.98

Table 2. Results for the Scenario B (Round Trip Time Equal to 40/200 msec.).

- Protocol hit time that is, the average latency of local misses which are served through documents cached by other servers. Similarly to the local hit time, this parameter is measured at the cache server contacted by the client.
- False remote hit ratio.
- Average amount of outgoing and incoming bytes per request at each cache server due to information exchange associated with cooperation. Depending on the cooperation protocol, this parameter may include both query/reply messages and notification messages. This value is measured at the application level, therefore it does not include packet headers due to TCP, UDP or IP protocols.

Observed values are reported in Table 1 and in Table 2. Experimental results show that *local hit ratio* and *protocol hit ratio* of 2TC are very similar to those of INF and QP. This is important because it points out that 2TC has no negative impact on the effectiveness of cooperation for locating documents within the global cache. Moreover, for all scenarios and parameter combinations, 2TC exhibits latencies comparable to those achieved by the best performing protocol. This result confirms that 2TC does not deteriorate the

response time perceived by the clients. As expected, QP has lower false hit ratios than those achieved by both INF and 2TC. This is due to the fact that the type of cooperation provided by INF and 2TC may let document location information become stale. However, in all experimented instances, the percentage of false hits is negligible.

The most important result is that 2TC shows a drastic reduction of the amount of information exchanged among the peers for handling cooperation. For both configurations the amount of transmitted (received) bytes per request for UDP query (reply) messages and TCP notifications is equal to (or less than) 50% of the amount of bytes required by INF and QP for cooperation. The reduction of network traffic with no degradation in performance is a strong, promising feature for guaranteeing the scalability of a cooperation protocol. Therefore 2TC is likely to scale well for larger configurations, thus possibly allowing cache sharing with limited overhead among a very large number of cache servers.

5 Conclusions

In this paper we have proposed a cooperation protocol (2TC) for systems of dozens or hundreds of distributed Web cache servers. 2TC embeds the two main cooperation policies for Web caching systems because it uses an

Informed Cooperation approach among close cache servers and a *Query Cooperation* approach among distant cache servers. In such a way, 2TC can explore the content of several cache servers through a single query message. Experimental results carried out on real traces confirm that cooperation can be achieved with a strong reduction (up to 50%) of the amount of transferred information if compared to other cooperation protocols. Reduction of network traffic is one of the most promising feature for guaranteeing scalability of a cooperation protocol. The important contribution is given by the fact that such a consistent reduction of protocol overhead is achieved with no performance degradation in terms of latency and cache hit rate.

References

- [1] M. Abrams, C. Standridge, G. Abdulla, S. Williams and E.A. Fox, "Caching Proxies: Limitations and Potentials", *Proc. 4th Int. World Wide Web Conference*, Boston, MA, 1995
- [2] E. Cohen, and H. Kaplan, "The Age Penalty and its Effect on Cache Performance", *Proc. Usenix Symp. on Internet Technologies and Systems*, March 2001.
- [3] P.B. Danzig, R.S. Hall, and M.F. Schwartz, "A Case for Caching File Objects Inside Internetworks", *Proc. ACM Sigcomm '93*, pp. 239-248, 1993.
- [4] S.G. Dykes, C.L. Jeffrey, and S. Das, "Taxonomy and Design Analysis for Distributed Web Caching", *Proc. Hawaii Int. Conf. on System Science*, January, 1999.
- [5] L. Fan, P Cao, J. Almeida, and A.Z. Broden, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", *Proc. ACM Sigcomm '98*, pp. 254-265, Vancouver, B.C, 1998.
- [6] S. Gadde, M. Rabinovich, and J. Chase, "Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches", *Proc. 6th Workshop on Hot Topics in Operating Systems*, May 1997.
- [7] S. Gadde, J. Chase, and M. Rabinovich, "Directory Structures for Scalable Internet Caches", Tech. Rep. CS-1997-18, Dept. of Computer Science, Duke University, Nov. 1997.
- [8] S. Gadde, J. Chase, M. Rabinovich, "A Taste of Crispy Squid", *Proc. WISP '98*, 1998.
- [9] M.R. Garey, and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: W.H. Freeman & C., 1979.
- [10] S. D. Gribble, "UC Berkeley Home IP HTTP Traces", July 1997. Available at <http://www.acm.org/sigcomm/ITA/>.
- [11] Internet Middleware Corporation, "Harvest cached-3.0 User's manual", San Jose', CA.
- [12] The Harvest Group, "Harvest Information Discovery and Access System", <http://excalibur.usc.edu/>, 1994.
- [13] M. Rabinovich, J. Chase, and S. Gadde, "Not All Hits are Created Equal: Cooperative Proxy Caching Over a Wide Area Network", *Proc. 3rd International Web Caching Workshop*, Manchester, UK, June 1998.
- [14] Squid Internet Object Cache: <http://squid.nlanr.net>
- [15] R. Tewari, M. Dahlin, H. Vin, and J. Kay, "Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet", Tech. Rep. TR98-04, Dept. of Computer Science, Univ. Of Texas at Austin, Feb. 1998.
- [16] V. Valloppillil, and K.W. Ross, "Cache Array Routing Protocol. Version 1.0", <http://ircache.nlanr.net/Cache/ICP/draftvinod-carp-v1-02.txt>, 1997.
- [17] D. Wessel, and K. Claffy, "Internet Cache Protocol (ICP). Version 2", 1998. <http://ds.internic.net/rfc/rfc2186.txt>