

Hybrid Cooperative Schemes for Scalable and Stable Performance of Web Content Delivery

Riccardo Lancellotti and Francesca Mazzoni
Michele Colajanni *

*Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia - Italy*

Abstract

Systems consisting of multiple edge servers are a popular solution to deal with performance and network resource utilization problems related to the growth of the Web numbers. After a first period of prevalent enthusiasm towards cooperating edge servers, the research community is exploring in a more systematic way the real benefits and limitations of cooperative caching. Hierarchical cooperation has clearly shown its limits. We show that the “pure” protocols (e.g., directory-based, query-based) applied to a flat cooperation topology do not scale as well. For increasing numbers of cooperating edge servers, the amount of exchanged data necessary for cooperation augments exponentially, or the cache hit rates fall down, or both events occur. We propose and evaluate two *hybrid* cooperation schemes for document discovery and delivery. They are based on a semi-flat architecture that relies on the grouping of the edge servers, and combines directory-based and query-based cooperation protocols. A large set of experimental results confirms that the combination of directory- based and query-based schemes increments the scalability of flat architectures based on “pure” protocols and guarantees more stable performance.

Key words: Internet caching architecture, Geographically distributed systems, Cooperation protocols, Document discovery, Performance evaluation

* Corresponding Author, *Email address:* colajanni.michele@unimo.it
URL: <http://weblab.ing.unimo.it/people/colajanni/>
Email address: lancellotti.riccardo@unimo.it,
mazzoni.francesca@unimo.it (Riccardo Lancellotti and Francesca Mazzoni).

1 Introduction

Web caching has evolved as the first way to address Web server and network resource utilization issues related to the growth of HTTP requests. The initial idea of using one edge server shows very low cache hit rates even in a context of homogeneous users. The proposed solutions aim to establish interactions among various cooperating edge servers. *Global caching* or *cooperative caching* architectures are used by public organizations (e.g., IRCache (1), NLANR (2)), Internet Service Providers (ISPs, e.g., AT&T (3)) and third party companies, such as Content Delivery Networks (e.g., Akamai (4), Speedera (5)).

After a first period of prevalent enthusiasm towards cooperating edge servers, the research community is exploring in a more systematic way the real benefits and limitations of cooperation. While there is no doubt that cooperative caching improves performance (6) when applied to a limited working set¹, the debate is open over the benefits of cooperation when applied to the entire Web content in a transparent way. We will consider the main results in Section 8, which discusses the related work.

In this paper, we focus on cooperative schemes for resource discovery and delivery that are traditionally based on two classes of “pure” protocols coming from the theory on distributed systems: *query-based* (15; 23; 28) and *directory-based* protocols (20; 13). We claim and demonstrate that any “pure” protocol is affected by some or all of the following drawbacks: it does not scale, its cooperation overhead increases too much as a function of the number of nodes, they are not stable because their performance (especially their cache hit rates) are too much dependent on the workload/system characteristics.

Hence, we propose and evaluate two main hybrid cooperation protocols (and a variation of one of them) that are based on a semi-flat organization of the edge servers. The idea is to achieve better cache hit rates and stable performance by splitting the resource discovery process in two steps, combining “pure” query-based and directory-based protocols. To this purpose we organize the cooperating edge servers in subsets belonging to two aggregation levels. The first level aggregation of the nodes denotes “groups” that are created on the basis of physical characteristics of the interconnections among the nodes. The second level aggregation denotes subsets with at least one representative node for each group. The goal is to find a resource in all the cooperating nodes, without having to contact all of them one by one. This is done in order to improve scalability, while the mixing of query-based and directory-based hopefully brings to a better stability than that of the “pure” protocols. We try to take advantage of the pros of each class of “pure” protocols and to alleviate

¹ This is the motivation of the success of many CDN companies that apply proactive and cooperative caching only to the content of their customer Web sites.

or even to eliminate the cons.

We have implemented a prototype for each scheme by modifying the Squid software (7). We find that hybrid cooperation schemes applied to a semi-flat topology can successfully reduce network traffic (especially over wide area links) without penalizing other metrics of interest for the end user. In particular, the schemes proposed in this paper show a better scalability than the “pure” protocols because they save network bandwidth for cooperation (one order of magnitude less than query-based protocols), guarantee same response time of query-based protocols, and do not penalize cache hit rates as it occurs to directory-based protocols applied to a flat topology of many cooperating edge servers. They also show a much better stability than the “pure” protocols because they are much more insensitive to network and workload characteristics.

The rest of this paper is organized as following. Section 2 gives qualitative motivations for the choice of hybrid cooperation schemes, by showing the limits of “pure” cooperation schemes. Section 3 gives qualitative motivations for the choice of the proposed cooperation schemes. Section 4 describes the two hybrid cooperation protocols proposed in this paper. Section 5 describes the workload models used in the experiments. Section 6 shows a performance evaluation of query-based and directory-based protocols. Section 7 reports the experimental results showing that hybrid protocols are the most viable solution for the performance stability of cooperative Web caching even over large numbers of cooperating edge servers. Section 8 discusses the related work. Finally, Section 9 presents some conclusions.

2 Limits of pure cooperation schemes

There are two main knobs to be handled in the design of a cooperation scheme among multiple edge servers: *cooperation topology* and *cooperation protocol*. Cooperation can be established along a vertical direction, namely hierarchical Web caching descending from the Harvest project (8), or along an horizontal direction, namely flat or distributed Web caching (9). In hierarchical architectures, a cache miss will result in looking for the resource to an upper level edge server (10). In flat architectures, every edge server is supposed to be at the same level of the others, and missed resources at one edge server are looked for in all cooperating nodes.

Hybrid architectures have been studied as well. Rodriguez et al. propose a hierarchical structure with cooperation among the edge servers at the same level (11). Tewari et al. investigate the advantages of cooperation using a hierarchical discovery mechanism based on partial cache index distribution

with a flat retrieval scheme, where an HTTP connection to a cooperating edge server is set up only in the case of hit detection (12).

Hierarchical architectures follow the idea of hierarchical Internet organization, with local, regional and international network providers. However, various studies have shown that a cooperation over N -tiers has scalability and coverage problems, especially for large sets of cooperating edge servers. There are so many results describing the drawbacks of pure hierarchical topologies especially for large N values (12; 13; 14; 15; 16), that in this paper it seems more convenient to limit the research space for alternative topologies to flat ($N = 1$ tier) and semi-flat ($N = 2$ tiers) architectures.

Any cooperation scheme among a set of distributed edge servers has to define a protocol to exchange some local state information. The two main and opposite approaches to disseminate state information are well defined in literature: *query-based protocols* in which exchanges of state information occur in response to an explicit request by an edge server, and *directory-based protocols* in which state information is exchanged among the cooperating nodes in a periodic way or at the occurrence of a significant event, with many possible variants in between.

The qualitative and quantitative analysis presented in (17), which we will also report in Section 6.1 and 6.2, show that document discovery carried out through a “pure” protocol in a flat topology does not scale well to more than a limited number of edge servers. Moreover, it does not guarantee stable performance in terms of cache hit rate and traffic overhead due to cooperation, because these performance metrics are too much sensitive to workload models and architecture configurations. In particular, a distributed summary-based protocol exhibits poor cache hit rates as strong information consistency is impracticable for large global caches. Analogously, the query mechanism underlying a query-based protocol could reveal ineffective for large global caches because of the message overhead of query broadcasts and time-out expirations.

We can conclude that the cause of limited scalability of query- and directory-based protocols is that they enforce the same cooperation method between any pair of edge servers. Moreover, the use of the same scheme makes the performance of the “pure” protocols heavily dependent on workload and system characteristics. It is not hard to find the perfect combination of parameters that makes one protocol apparently better than others. The real difficulty is to find a scheme that guarantees *stable* (even if not always best) performance in any condition. The limits on stability shown by pure flat schemes motivate the proposal of hybrid solutions.

3 Two-tier architectures

The key idea behind two-tier architectures is to provide a lookup mechanism that can carry out a discovery task over a set of edge servers without the need to contact every node in the set. This idea brings the need of a two-tier organization of the edge servers and we also need a protocol to take advantage of this two-tier topology. The philosophy of our two-tier architecture results in two main choices both in the topology of the architecture and in the lookup process.

- On the topological side, we introduce a partition on the set of edge servers based on physical characteristics. In particular we create *groups* of well-connected nodes. Then we choose a representative edge server for each group. We put these representatives all together, thus creating what we call a *representative subset*.
- The lookup process is divided in two steps, called *tiers*. Each tier can occur within a group or can involve a representative subset.

3.1 Two-tier topologies

We consider solutions for resource discovery based on an architecture where edge servers are logically organized in a two-tier topology ($N = 2$) that uses hybrid cooperation protocols combining query- and summary-based schemes.

The choice of a two-tier scheme derives from the observation that typically ISPs deploy caches at the points of presence (POPs), and corporate networks deploy caches within each of their locations. Moving the lookup process beyond $N > 2$ has been demonstrated not to be convenient because more than two steps tend to lengthen the response time (12), thus making often convenient to get the resource from the origin Web server.

To describe the two-tier logical organization, let us take as example the nine edge servers in Figure 1.

A *group* is a subset of “well connected” edge servers in terms of distance and network connectivity. The connectivity requirements for WAN network applications subject to variable traffic conditions are still open problems, although there are studies that demonstrate that routing in the Internet core tends to be more stable than that observed some years ago (18; 19). Independently of these issues, we can easily accept any edge server partition that achieves intra-group connections with typical lower latency, lower congestion and higher bandwidth than inter-group connections.

For instance, in Figure 1, we could define the following groups: $G_1 = \{A, B, C\}$, $G_2 = \{G, H, I\}$ and $G_3 = \{X, Y, Z\}$.

A *representative subset* is a subset of edge servers that includes one representative for each group. We create representative subsets in order to obtain an exchange of information only among the representatives of each group. That is we do not want to have a global information exchange. We already observed that pure protocols do not follow this approach: in summary-based cooperation every edge server must have some knowledge about the content of the neighbor cooperating nodes, while in query-based cooperation the query and response message exchange involves every edge server. The previous section suggests that a lookup process involving every edge server leads to poor scalability. On the other hand, we can achieve high scalability by providing a lightweight lookup mechanism that contacts only a subset of the edge servers and yet is able to locate resources also on cooperating nodes not directly involved in the lookup. Since there are many nodes in a group, each of them can be chosen as representative for its group, thus we can have many different representative subsets depending on how we choose the representative nodes. We can also have many representative subsets at the same time, that is a group has many representatives that work simultaneously.

For instance, in Figure 1, we define the following representative subsets for collaboration inside the groups: $RS_1 = \{A, G, X\}$, $RS_2 = \{C, I, Z\}$.

3.2 Cooperation protocols for two-tier topologies

We now describe in greater detail the dynamic lookup scheme used in two-tier cooperation. Two-tier architectures allow us to split the cooperative resource lookup process in two separate phases called *lookup tiers*. The first-tier lookup is done among a subset of nodes, be it a group, or one of the representative subsets, while the second-tier lookup, if necessary, is done among the servers of the other subsets. In this way, the first attempt is done by contacting only a few servers, thus bringing a small overhead on the network. Then, only if it is necessary, other edge servers are contacted as well. In any case, we want to avoid the need of contacting all the cooperating nodes one by one, because this would lead to poor scalability. Instead, we introduce a way of collaborating among the edge servers of one group. Thus, we need to contact only the representative in order to know if an edge server belonging to a certain group owns or not the requested resource.

We will call *intra-group cooperation* the one that occurs inside a group, while we will call *inter-group cooperation* the one that occurs among the groups by contacting their representatives.

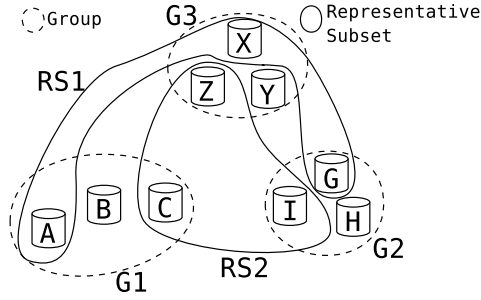


Fig. 1. Example of a two-tier architecture: groups and representative subsets.

Table 1

Possibilities in combining different mechanisms:

Q=Query-based protocol, S=Summary-based protocol

case	1st tier	2nd tier
a	Q	Q
b	Q	S
c	S	Q
d	S	S

Many protocols can be used as first-tier and/or second-tier lookup protocols, such as query-based and summary-based. A summary-based protocol exchanges small amounts of traffic for cooperation and guarantees low latency times for lookup (just a search in a table), but we will see that its cache hit rates are highly sensitive to various workload characteristics. On the other hand, a query-based protocol requires larger amounts of traffic for cooperation and has higher latency times, but its cache hit rates are high and typically more stable for different workloads.

We have already presented in Section 2 the limits of “pure” protocols applied to a flat topology, hence we now consider protocols that combine query- and summary-based lookup on a two-tier topology. There are four possibilities to evaluate.

Table 1 allows us to draw some conclusions to focus the research on the most promising combinations.

- **Case “a”** utilizes a query-based protocol for both phases of cooperative lookup. Hence, it becomes a “pure” flat query-based protocol, that is not of interest for our investigation. Furthermore, this brings to three consecutive latencies due to the three consecutive queries (looking up in the first group, contacting the representatives of the other groups and looking up in the other groups), thus resulting in unacceptably slow response times.
- **Case “d”** utilizes a summary-based protocol for both phases of cooperative

lookup. Hence, it becomes a “pure” flat summary-based protocol, that is not of interest for our investigation.

- **Case “b”** utilizes a query-based protocol for first tier lookup and a summary-based one for the second tier lookup. While the first-tier lookup is feasible by means of a query-based protocol, the problem arises when it is necessary to contact the representatives of each group. If we use a summary-based protocol, what we obtain is an answer that regards the representative edge server only. If we want an answer that regards the whole group, the representative edge server should send as reply all the digests of its group. The schema would result in a global digest exchange, thus being contrary to our basic idea of avoiding global exchanges.
- **Case “c”** utilizes a summary-based protocol for first tier lookup and a query-based one for the second tier lookup. A summary-based protocol for the first-tier lookup is immediately aware whether a valid copy of the resource exists in a subset of nodes. However, as the cache hit rate of the summary-based protocol can be low, it seems appropriate to integrate it with a query-based protocol for the second-tier lookup, that has the role of reinforcing hit rates and achieving more stable performance. This schema does not show evident limits hence, we use it as the basis of the hybrid cooperation protocols we propose in this paper. The doubt that we cannot solve is about the most convenient combination of query- and summary-based protocols as intra- and inter-group cooperation protocol. That is if the first-tier lookup should be an inter- or intra-group cooperation. Hence in the next section we consider hybrid cooperation schemes that combine in different ways the advantages of summary-based protocols with those of query-based protocols.

4 Two-tier cooperation protocols

In this section we consider two hybrid schemes and a variation of one of them. They share some common features. In every scheme the lookup process starts with a summary-based protocol and continues, when necessary, with a query-based one. *InterQ-IntraS* and *InterQ-IntraS-DM* (a variation of the first summary-based scheme) use a summary-based protocol for intra-group cooperation and a query-based protocol for inter-group cooperation. *InterS-IntraQ* uses the opposite approach that is, a query-based and a summary-based protocol for intra- and inter-group cooperation, respectively. This means that the first-tier lookup occurs among the nodes of a group (intra group) for *InterQ-IntraS* and *InterQ-IntraS-DM*, whereas the first-tier lookup is among the groups (inter group) for *InterS-IntraQ*. For instance, in Figure 1, the first-tier lookup occurs among *A*, *B* and *C* (or among *G*, *H* and *I*, or among *X*, *Y* and *Z*) for *InterQ-IntraS* and *InterQ-IntraS-DM*, and among *C*, *I* and *Z* (or

among A , G and X) for InterS-IntraQ.

Before going into a detailed description of the hybrid cooperation schemes, let us introduce some definitions we will use throughout the paper. A client request reaching an edge server of a two-tier cooperative architecture may experiment different effects. It may result in a *local hit* when a valid copy of the requested resource is found in the first contacted edge server: the resource is sent to the client and no cooperation is activated. Otherwise, a *first-tier hit* occurs when the requested resource is found in an edge server of the same group to which the first contacted edge server belongs to. A *global hit* occurs, after a *first-tier miss*, when the resource is found in an edge server by means of both tiers of cooperation. Finally, we have a *global miss*, if the resource must be retrieved from the origin Web server, because both lookup tiers fail in finding a valid copy of the resource in any edge server.

4.1 InterQ-IntraS cooperation protocol

The InterQ-IntraS scheme uses a summary-based protocol based on cache digests (20) for intra-group cooperation (the ‘‘Summary’’ part), that is inside a group, and a query-based cooperation protocol for inter-group cooperation (the ‘‘Query’’ part), that is when contacting the representative nodes. The main advantage of this scheme is that any edge server is informed about the content of all the cooperating edge servers of the same group. Hence, it seems convenient to limit the query-based cooperation on the second tier to one representative server for each group, called *group master*. Typically, we select the edge server offering superior computing power and better connections to the other groups to act as a master. Masters are edge servers that can be directly contacted by clients, as well as the other edge servers.

To describe this protocol we consider the example of the architecture shown in Figure 2.

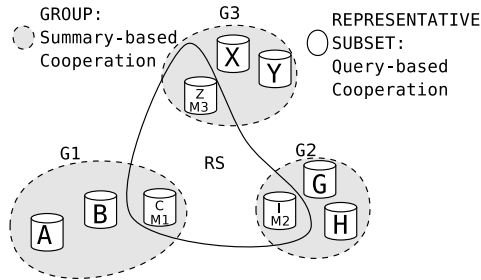


Fig. 2. Configuration example of a system using InterQ-IntraS cooperation

Let G_1 denote the group $\{A, B, C\}$, G_2 the group $\{G, H, I\}$ and G_3 the group $\{X, Y, Z\}$. The three nodes $\{C, I, Z\}$ compose the representative subset in which the second-tier lookup is carried out through a query-based protocol.

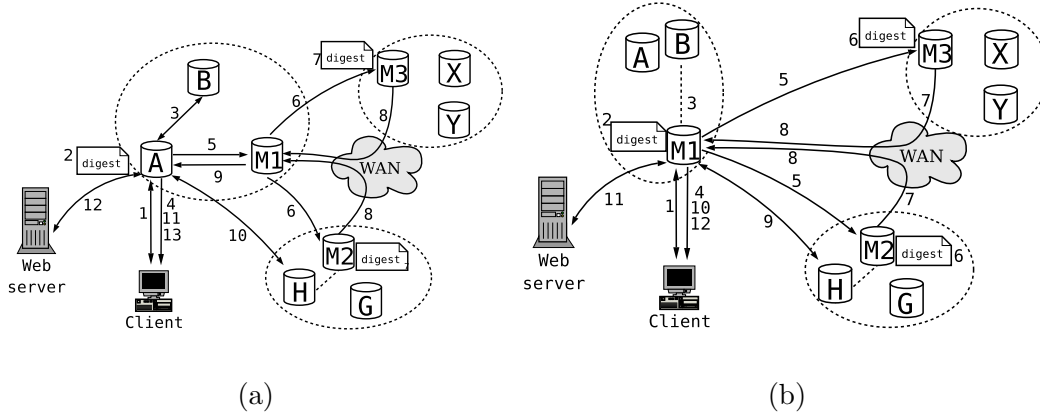


Fig. 3. InterQ-IntraS cooperation. A client requests resources to a non master proxy (a), or to a master proxy (b).

To better evidence the group masters, we have added the labels M_1 , M_2 and M_3 to the nodes C , I and Z , respectively. As you can infer, there are only two scenarios:

1. A client requests a resource by contacting a non-master proxy.
2. A client requests a resource by contacting a master proxy.

The former scenario is described in Figure 3(a). The client issues an HTTP request (step 1) for a given resource to a non master proxy, for instance A in Figure 3(a). A searches in its own cache. If it finds the requested resource, it sends the copy to the client. All this phase does not require the activation of any cooperation scheme. If A does not have a valid copy of the resource in its cache, then the first-tier lookup is activated (step 2): A looks up the digests to examine the content of the other edge servers belonging to its group. If A discovers the requested resource in B , then A sets up a TCP/IP connection with B to retrieve it (step 3). If the download is successful, A forwards the resource to the client (step 4). If B cannot return the resource or it cannot be found in any edge server belonging to the A group, A activates the second-tier lookup (step 5) by sending a query to his own master (M_1 in Figure 3(a)). In its turn, M_1 sends an ICP query (step 6) to each of the other masters (M_2 and M_3 in Figure 3(a)). The masters look up in their digests (step 7) and answer (step 8) to the requesting master (M_1) whether the resource can be found in any edge server belonging to their group. Let us suppose, for instance, that M_3 answers with a miss message, indicating none of its group edge servers owns a copy. On the other hand, M_2 answers with a hit message, indicating that an edge server, (H in Figure 3(a)), owns a valid copy of the requested resource. Then M_1 sends a message to A (step 9) containing the address of the server which resulted in a hit. A sets up an HTTP connection with H (step 10) to download the requested resource. If the resource retrieval is successful, A forwards it to the client (step 11), otherwise A contacts the origin Web

server (step 12) and then forwards the resource to the client (step 13).

The scenario where a client requests a Web page directly to a master proxy (e.g. M_1) is described in Figure 3(b). In this case, M_1 looks up in its cache (step 1) a valid copy of the resource. If there is a miss in the local cache, M_1 uses the digest search algorithm of its group to look for possible remote hits (step 2). Two possibilities exist:

1. There is a possible hit in its group, for instance in B . Then M_1 sets up a TCP/IP connection with B (step 3); it fetches the resource and forwards it to the client (step 4).
2. There is no hit in its group. The master sends an ICP query to the nearest masters (step 5). The successive behavior is similar to that described in the previous scenario (Figure 3(a)) from step 7 to step 13, keeping in mind that in Figure 3(b) the corresponding steps are numbered from 6 to 12.

InterQ-IntraS-DM cooperation protocol

The previous InterQ-IntraS scheme can achieve high cache hit rates, but it places a significant amount of work on the group masters. Each of them has to serve client requests and, at the same time, has to act as a gateway for query-based cooperation. Especially when masters do not have computational capacity higher than that of the other edge servers, the twofold role of edge and master server can easily congest these nodes and lead the entire cooperation system to perform poorly.

To address this bottleneck issue, we propose the idea of using *dedicated masters* that do not act as edge servers. This approach denotes a different architecture, called *InterQ-IntraS-DM*, that works similarly to the previous InterQ-IntraS scheme. The main difference is that now each master acts as a gateway for its group and as a directory for the other groups, but it cannot be directly contacted by clients. The limited congestion in master nodes is done at the expenses of a reduced number of edge servers available for client service. It is interesting to note that the InterQ-IntraS-DM scheme is fully interoperable with the InterQ-IntraS scheme. We also implemented a flexible scheme, where some masters are dedicated, while others can accept client requests, depending on the expected level of congestion. However, the evaluation of the variants of this scheme are outside the scope of this paper.

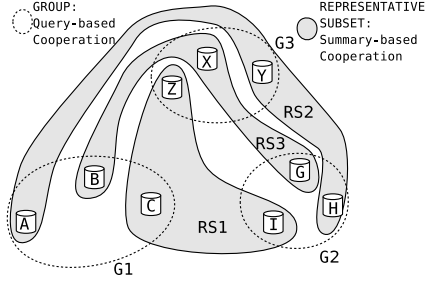


Fig. 4. InterS-IntraQ architecture

4.2 InterS-IntraQ cooperation protocol

The motivation for the *InterS-IntraQ* scheme comes from the observation that in the InterQ-IntraS schemes the large majority of traffic for cooperation (due to ICP messages) transits through the inter-group links, that are potentially slower and more expensive (for ISPs) than intra-group links. Hence, in *InterS-IntraQ*, inter-group cooperation is carried out through cache digests that generate a minor traffic. Communications inside a group, where nodes are connected through faster and less expensive links, are based on an enhanced version of ICP messages.

As a further positive effect of the InterS-IntraQ scheme, we can indicate that it bypasses the bottleneck related to the masters, and it can achieve a better load balancing because it uses multiple representatives of each group for the requests.

Figure 4 shows an example of this cooperation scheme by considering the same architecture shown in Figure 2. The three groups are $G_1 = \{A, B, C\}$, $G_2 = \{G, H, I\}$, and $G_3 = \{X, Y, Z\}$. We also define the following *representative subsets* for summary-based cooperation: $RS_1 = \{C, I, Z\}$, $RS_2 = \{A, H, Y\}$ and $RS_3 = \{B, G, X\}$. Each representative subset must contain at least one member of each group. In this version of InterS-IntraQ cooperation protocol, we require that the groups G_i and the representative subsets RS_i denote a partition of the initial set of nodes. This leads to the conclusion that the maximum number of representative subsets is equal to the minimum cardinality of the groups. Additionally, the minimum number of nodes inside a representative subset RS_i is equal to the number of groups in the system, because at least a member of each group must belong to a representative subset.

To describe how this cooperation scheme works, we explain what happens when an edge server receives an HTTP request from an end-user (step 1), as shown in Figure 5. The edge server searches in its local cache (step 2). If the requested resource is present, it is forwarded to the client (step 3), otherwise the first-tier lookup is activated. The edge server analyzes the digests of the nodes belonging to its representative subset (step 4). If any of the first-

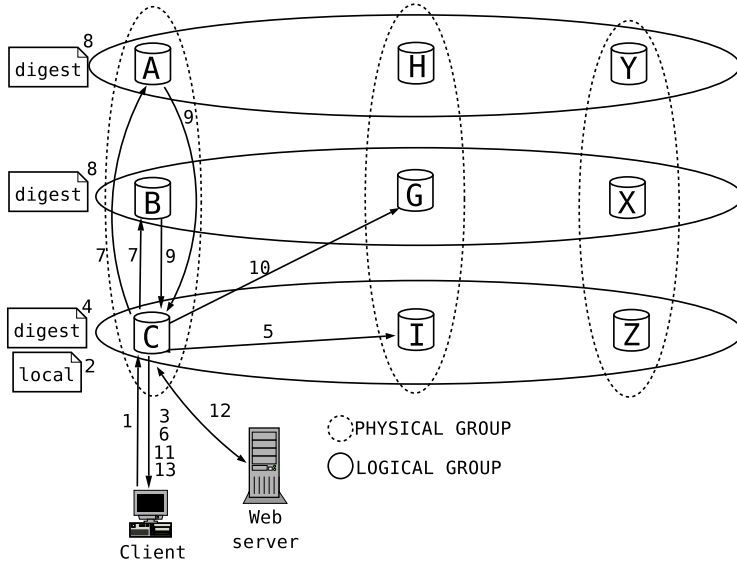


Fig. 5. InterS-IntraQ cooperation protocol: answer to an HTTP query from an end user.

tier cooperating nodes owns the resource, (e.g., I in Fig. 5) then the server fetches it (step 5) and forwards it to the end user (step 6). On the other hand, if the first-tier lookup comes out in a first-tier miss, then the second-tier lookup is activated and ICP queries are sent to the group nodes (step 7). It is important to observe that these nodes do not answer only for the content of their caches. Since they also know the content of their representative subset nodes by viewing the respective digest (step 8), every node can answer for its whole representative subset (step 9). For example, A in Fig. 5 answers for itself, for H and for Y. If any of the cooperating nodes owns the resource, the edge server activates an HTTP connection (step 10) to fetch it and forwards it to the end user (step 11). Otherwise, if no node has the resource, or it cannot be retrieved, then the edge server reaches the origin Web server (step 12), fetches the requested resource and forwards it to the end user (step 13).

5 Workload models for performance evaluation

The difficulty of defining a “typical” workload model is a well known issue, because studies on real traces show great differences. For the experiments, we prefer to use two synthetic workload models generated by Web-Polygraph version 2.5 (21). They intend to capture two “realistic” Internet scenarios, that are based on the model proposed for the second cache-off by IRCACHE (1). The basic workload is characterized by a high *recurrence* (that is, the probability that a resource is requested more than once), and small inter-arrival times for client requests. This latter characteristic contributes to degrade summary-based cooperation because of capacity and consistency misses, as observed in

Section 2. We can anticipate that the chosen workload models tend to penalize the summary-based protocol with respect to query-based cooperation. Nevertheless, for a more than fair comparison with the pure query-based protocols, in this paper we prefer to report results of a worst-case scenario for first-tier lookup, that is always based on a summary-based protocol.

Every experiment was done twice and data measures were collected only in the second run so to emulate a steady state scenario. It is worth to observe that an increment of the number of edge servers augments the global cache capacity, but also the size of the working set, because the number of clients is incremented proportionally. Preliminary tests, that we do not report here for space limitations, were done to tune some parameters of Squid. For example, for summary-based cooperation we find more convenient to use a digest rebuild period of 60 seconds to reduce the consistency miss effects due to frequent cache object replacements (note that the Squid default value is 60 minutes).

Workload 1 represents a set of heterogeneous users with different interests. This characteristic, together with the document popularity model, leads to a high spatial locality. Client requests show also some temporal locality, so that only 30% of the workload is active at a given time. Requests are referred to a mix of content types consisting of images (65%), HTML documents (15%), binary data (0.5%), others (19.5%). The workload model defines a set of *hot* resources (1% of the working set) that receive about 10% of the requests. The heterogeneity of the user interests is represented by the fact that only 50% of the requests is taken from a “public” set of pages common to all clients, while the remaining 50% are taken from a “private” set, different for each client. Each client is configured to visit more than once only 80% of the URLs and the object cacheability is 80%. HTML resources typically contain embedded objects.

Workload 2 is a modified version of Workload 1, where the client population is more homogeneous. Here, clients share the same interests, so spatial locality is reduced, while the overall access locality is augmented. Moreover, the popularity model is chosen so to increase the size of the hot fraction of the workload. This workload takes less advantage from local cache hit rates and puts more pressure on coordination. The modified main parameters of Workload 1 are indicated below. The hot set of the workload is larger (15% of the access are referred to a hot set corresponding to 5% of the URL-space). This creates a popularity distribution with a heavier tail. The working set size keeps growing trough the whole experiment (there is no temporal locality). The public fraction of requests is 100%, which means that every document in the workload is public, so that each resource can be requested by any client.

6 Performance evaluation of pure cooperation protocols

For a more conscious choice of the cooperation protocol we extensively test the “pure” schemes in a flat topology under different conditions. Only the most significant results are reported here. The goal is to study the parameters to which they are more sensitive. We consider three workload classes, two of them “realistic” (Workload 1 and 2), one “ideal” in the sense that it favors object caching. Unlike the previous two workload models, in the Ideal Workload the object cacheability is always 100%, the object lifetime is assumed infinite (there is no chance of stale hit), and there is a high probability (0.87) that an object is requested more than once. Other parameters are similar to those used for Workload 1 and 2. The two main performance results considered here are the *global cache hit rate* (Global HR) and *overhead for cooperation* (expressed as an average of the additional bytes transmitted per each delivered document). There are many parameters that can influence the performance of the cooperative protocols, and most of them are considered here: number of cooperating edge servers, cache capacity with respect to the working set, local cache hit rate determined by the characteristics of the workload, frequency of client request arrivals to the edge servers.

6.1 Limits of query-based protocols

Query-based protocols are conceptually simple. When an edge server experiences a local miss, it sends a query message to all of the cooperating edge servers² in order to discover whether one of them caches a valid copy of the requested resource. In the positive case, the recipient edge server replies with a hit message, otherwise it may reply with a miss message or not reply at all. This query-based solution was proposed by the Harvest project (8), and then implemented in NetCache (22) and Squid (7). The last two use the Internet Cache Protocol (ICP) as their query mechanism (23).

Let us summarize the main pros and cons of ICP shown by our experiments. On the positive side, ICP has a high cache hit rate that, in a local area, is nearly insensible to the number of involved edge servers (as shown in the fourth column of Table 2 for sets of 8, 15 and 30 nodes), to the cache capacity, and to the frequency of client requests (these experiments are described in greater details in Section 7.2). The dependency of the cache hit rate to the workload model was largely expected because the cacheability and volatility of the objects change. In the Ideal case, the global hit rate of ICP is close to 90%.

² Some implementations keep records of the node state, so the query is not sent to overloaded or temporarily off-line edge servers.

Table 2

Cache size per node [% of working set] for ICP protocol

Nodes	Fraction of cache capacity per node	Local HR	Global Hit Rate	Overhead [$\frac{\text{bytes}}{\text{req.}}$]
8	11%	58.92%	88.20%	474
15	5%	42.43%	88.07%	1520
30	3.5%	34.14%	87.71%	3583

The overhead for cooperation is the main drawback of ICP that is evidenced by our experiments. From Table 2 we see that the number of bytes required for cooperative document discovery increases more than linearly when the number of cooperating edge servers augments. This result occurs for any kind of workload, and it represents a limit to the scalability and stability of query-based protocols. Increasing the number of cooperating edge servers leads to scalability problems because there is an increment of both the number of ICP queries and the number of recipients of each query. The last column of Table 2 confirms this: when 30 nodes are involved, the traffic for cooperation is more than one third of the traffic generated by client requests. Increasing the cache size can alleviate this problem because the local hit rate augments, so less client requests start a cooperative lookup process. However, recent studies indicates that the working set seems to increase faster than the typical cache capacity (24).

The scalability and stability problems of query-based protocols are even more serious when they are applied over a geographic context. For example, in (13) it has been shown that cooperation overhead may become unacceptable even with less than 10 cooperating edge servers. Other limits to the scalability of query-based protocols are found in literature. One of the most interesting, described in (20), is related to the mechanism used to detect cache misses. During cooperative lookup, the first received hit message is used to locate the requested resource on a cooperating edge server. To detect a miss, instead, the edge server must wait for a reply by all cooperating nodes or for a time out. The consequence is that in query-based protocols cache misses are much slower to be detected than cache hits. Our experiments in Section 7.3 confirm this conclusion. Increasing the number of cooperating edge servers makes this problem even worse, especially if cooperation occurs among distant nodes. Moreover, the probability of packet loss or delay is increased as the number of cooperating edge servers augments, in a way proportional to $1 - (1 - P_{err1})^M$ (25), where P_{err1} is the probability of packet loss or delay when only one packet is sent and M is the number of cooperating nodes. In a geographic environment, where packet delay and loss are more likely, the response time of query-based protocols augment because of the higher numbers of cache misses.

6.2 Limits of directory- and summary-based protocols

Directory-based protocols are conceptually more complex than query-based schemes, especially because they include a large class of alternatives, of which the two most important are: the presence of one centralized directory or multiple directories disseminated over the cooperating edge servers; the frequency for communicating a local change to the directory/ies. It is impossible to discuss here all the alternatives that have been the topics of many studies. We limit our analysis to distributed directory-based schemes because it is common opinion that in a geographically distributed system any centralized solution does not scale, the central directory server may represent a bottleneck and a single point of failure, and it does not avoid the query delays during the lookup process.

In a distributed directory-based scheme, each edge server keeps a directory of which URLs are cached in every other cooperating node, and uses the directory as a filter to reduce the number of queries. Distributing the directory among all the cooperating nodes avoids the polling of multiple edge servers during the discovery phase, and in the ideal case makes object lookup extremely efficient. However, the ideal case is affected by large traffic overheads to keep the directories up-to-date. Hence, real implementations use multiple relaxation, such as compressed directories (namely, *summary*) and less frequent information exchanges to save memory space and network bandwidth, respectively. Examples of compression used to reduce the dimension of the transmitted messages are the Bloom filters (26), used by Summary Cache (13), and Cache Digests (20), that offer a form of lossy compression of the cache indexes in which a certain amount of false hits is allowed. These summary-based schemes weaken the consistency of distributed information. However, as a side result of this paper we have verified that summaries limit network overheads with surprising efficiency, even if frequent exchanges are carried out. For the experiments, we choose Cache Digests as a representative of the summary-based architectures, because of its popularity and its implementation in the widely adopted Squid.

Unlike ICP that has rather stable cache hit rates, our experiments show that the cache hit rate of the summary-based schemes is heavily dependent on many parameters. In particular, it diminishes as the number of cooperating edge servers increases, and even when the working set augments more than the cache capacities. Our tests with the same Ideal workload previously described (with a frequency of 10 client requests per second) show that the global hit rate decreases from 50% to 33% when the cooperative nodes pass from 8 to 30 (fourth column in Table 3). This instability is also due to the acceleration of the exchanges of objects between the nodes when the total working set augments more than the cache capacity (second column in Table 3).

Table 3

Cache size per node [% of working set] for CD protocol

Nodes	Fraction of cache capacity per node	Local HR	Global Hit Rate	Overhead [$\frac{\text{bytes}}{\text{req.}}$]
8	11%	31.83%	50.23%	3.31
15	5%	20.99%	37.01%	4.34
30	3.5%	18.79%	33.80%	5.41

Moreover, we found a (stronger than expected) correlation between the cache hit rate of Cache Digest and the frequency of client request arrivals. In Table 4 we report the impact of capacity miss on hit rate when 11% of the content can be held in each of the 8 edge servers. This table shows that the reduction of the cache hit rate when the frequency of client requests increases is significant: from 76% to 50%, when all other parameters are kept constant. The dependency of the Cache Digests hit rate on the number of cooperating edge servers and on the frequency of client request arrivals are confirmed by all our experiments. Actually, they are two effects of the same problem (out-of-date summary information) caused by capacity misses and consistency misses, respectively.

Capacity misses occur when a directory references an object that the edge server has previously discarded to make space to other objects. Larger numbers of cooperating edge servers augment the working set because more clients are served. If the capacity of each node remains the same, the replacement algorithm tends to be activated more often, with consequent higher numbers of capacity misses. In our experiments, each node can contain from 11% to 3.5% of the entire working set. As shown in the second column of Table 3, the capacity misses due to replacements reduce the hit rate when the cooperating edge servers augment.

Consistency misses are caused by the asynchronous propagation of summary contents that further reduce the accuracy of the directories, especially when the frequency of client request arrivals is much greater than that used for summary refresh.

On the positive side for the summary-based protocols, we have that the overhead for cooperation is really low and almost independent of the number of nodes: the traffic generated for cooperation is even three orders of magnitude lower than that caused by ICP (compare the last column of Table 3 against that of Table 2). Moreover, this cooperation overhead is quite stable with respect to many parameters: it is mainly a function of the cache capacity, because increasing the capacity augments the size of the digests being exchanged. Dependency on other parameters, such as digests rebuild frequency and number of edge servers, is much less evident.

Table 4

Global Hit Rate of Cache Digest as a function of the frequency of client requests

Client request frequency	Global Hit Rate
$\left[\frac{\text{req.}}{\text{sec.}}\right]$	
10	50%
5	72%
2	74%
1	76%

7 Performance evaluation of hybrid cooperation schemes

The InterQ-IntraS, InterS-IntraQ and InterQ-IntraS-DM cooperation architectures are implemented as modifications of the Squid 2.4 software (7). All prototypes were extensively tested to verify their scalability and to compare the stability of their performance with that of the “pure” query-based and summary-based protocols. The first set of experiments, carried out with edge servers placed on the same network segment, focuses on cache hit rates, cooperation overheads, and sensitivity to other parameters, such as cache capacity and workload models. The InterQ-IntraS-DM scheme was not tested in this context because measuring response times and congestion (the main issues addressed by this cooperation scheme) would be meaningless in a LAN. With a second set of experiments we measure the response times of client requests in a geographic scenario. This is to guarantee that the stability of hybrid schemes on cooperation overhead and cache hit rates are not achieved at the expenses of user response time. These experiments also include the InterQ-IntraS-DM cooperation protocol.

7.1 Scalability of the cooperation protocols

In Section 6 we have observed that “pure” cooperation protocols do not scale well over a limited number of cooperating edge servers even in the case of Ideal workload: traffic overheads and poor cache hit rates limit query-based and summary-based protocols, respectively. In this section, we aim to verify whether hybrid cooperation protocols based on a two-tier topology can provide a better scalability and more stable performance than “pure” protocols. Moreover, we are interested in finding which is the best hybrid scheme among the three proposed in this paper. To this purpose, we compare cache hit rates and cooperation overheads for an architecture with an increasing numbers of edge servers that are subject to Workload 1 and Workload 2. As a comparison testbed, we also include results for an architecture with the same number of

Table 5
Cache hit rates and overheads for Workload 2

Number of Nodes	Local HR	1st-tier HR	Global HR	Intra-Group	Inter-Group	Total	Relative
				Overhead	Overhead	Overhead	Overhead
				per request [$\frac{\text{bytes}}{\text{req.}}$]	per request [$\frac{\text{bytes}}{\text{req.}}$]	per request [$\frac{\text{bytes}}{\text{req.}}$]	per node [$\frac{\text{bytes}}{\text{req.}}$]
InterS-IntraQ							
8	38.38%	45.69%	60.16%	284.25	7.32	291.58	36.44
15	25.89%	37.87%	55.66%	427.38	18.10	445.48	29.70
30	15.13%	30.42%	51.04%	543.05	49.10	642.14	21.40
InterQ-IntraS							
8	41.00%	52.50%	60.13%	62.05	41.67	104.72	13.09
15	23.63%	36.16%	60.96%	83.40	118.87	202.36	13.49
30	13.49%	26.71%	60.26%	116.61	249.58	366.19	12.21
Cache Digest							
8	38.00%		53.05%			5.70	0.71
15	26.92%	n/a	44.29%	n/a	n/a	6.32	0.42
30	16.28%		33.77%			6.84	0.23
ICP							
8	40.54%		66.49%			780.84	97.60
15	27.07%	n/a	66.15%	n/a	n/a	1882.64	125.51
30	16.19%		65.20%			4500.42	150.14
No Cooperation							
8	49.64%		49.64%				
15	27.84%	n/a	27.84%	n/a	n/a	n/a	n/a
30	16.35%		16.35%				

nodes that do not cooperate for document discovery (namely, No Cooperation).

Table 5 and Table 6 summarize the results regarding Workload 2 and 1, respectively. In each table, the first three columns report the cache hit rates (local, first-tier and global), while the last three columns report the traffic overhead due to cooperation, that is indicated as additional bytes exchanged for each client request.

Actually, the most interesting results for evaluating the cooperation schemes are obtained for Workload 2. Hence, we derive the main conclusions from these experiments and use the results obtained with Workload 1 to confirm these results or show some light differences. We have already observed that increasing the numbers of edge servers leads to an increment of the working set size and a consequent reduction of the percentage of documents that can be cached in each node. For example, for Workload 2 these percentages decrease from 7.5% to 2% of the working set when the cooperating nodes pass from 8 to 30. But the most interesting aspect of the Workload 2 characteristics is that a larger number of nodes leads to a sensible reduction of the local cache hit

rates. From Table 5 we have that the local cache hit rates go from about 40% to about 15%. Some differences dependent on the cooperation scheme were expected, because an edge server belonging to a cooperative system receives requests from its clients and other edge servers. This alters the access patterns and in practice reduces the document locality. Indeed, the best local hit rates are obtained by the No-Cooperation scheme that preserves locality at most. Same behavior is shown in Table 6 referring to Workload 1, although in this case the local hit rates remain higher than those observed for Workload 2. The larger space of improvement for cooperation motivates our main interest for this latter model.

From Table 5 we can observe that the No-Cooperation scheme is not able to face the reduction of the local cache hit rates, and also Cache Digest shows many limits especially for higher numbers of cooperating edge servers. On the other hand, ICP and the other hybrid cooperation schemes compensate the effects of the local hit rate reduction with a stable and always over 50% global hit rate. InterQ-IntraS is better than InterS-IntraQ, but plain ICP is even better. The motivation for these results is related to the choice of the workload models that tend to penalize summary-based cooperation. Indeed, an increment of the working set size rises the frequency of object replacement, thus reducing the accuracy of the exchanged cache digests (there is an increment of capacity and consistency misses). Cache Digests is particularly sensitive to this and its hit rate (quite low even with only 8 cooperating nodes) is further reduced when the working set size augments. ICP is not affected by the frequency of cache replacements and this explains its good performance. However, it is remarkable how well the hybrid schemes are able to address the issues related to cache replacement even if the peculiarities of the workload model affect their Summary cooperation component occurring on the first-tier. In particular, InterQ-IntraS seems the most stable, because it overcomes the drawbacks due to the low first-tier rates by means of great performance on the second-tier. It is remarkable the case with 30 nodes, where its cache hit rates pass from 13% to 26% to 60%, that is the same value observed for lower numbers of cooperating edge servers. On the other hand, the InterS-IntraQ scheme finds a minor benefit from the query-based cooperation on the second-tier. The consequence is a reduction of the global cache hit rate of about 15%, that however remains much better than that of Cache Digests experiencing a decrease of about 38%.

The results reported in Table 6 related to Workload 1 confirm the previous conclusions: InterQ-IntraS and ICP show the most stable cache hit rates, followed by InterS-IntraQ and Cache Digest. Now, because of the higher spatial locality in request patterns, the hit rates are generally higher than those shown in Table 5. The main contribution to these results are due to the local hit rate (see No-Cooperation performance), which is increased by the greater locality caused by the higher percentage of “private” requests, as explained in the de-

Table 6
Cache hit rates and overheads for Workload 1

Number of Nodes	Local HR	1st-tier HR	Global HR	Intra-Group	Inter-Group	Total	Relative
				Overhead	Overhead	Overhead	Overhead
				per request [$\frac{\text{bytes}}{\text{req.}}$]	per request [$\frac{\text{bytes}}{\text{req.}}$]	per request [$\frac{\text{bytes}}{\text{req.}}$]	per node [$\frac{\text{bytes}}{\text{req.}}$]
InterS-IntraQ							
8	55.01%	57.60%	63.84%	332.15	7.14	339.29	42.41
15	39.81%	43.82%	52.55%	317.66	10.88	328.54	21.90
30	35.80%	44.11%	54.38%	427.40	24.46	451.86	15.06
InterQ-IntraS							
8	48.24%	55.40%	62.56%	70.44	82.84	153.28	19.16
15	40.77%	50.53%	63.62%	65.54	95.09	160.64	10.71
30	31.65%	55.12%	64.06%	88.37	192.83	281.20	9.37
Cache Digest							
8	31.47%		40.25%			3.67	0.46
15	30.52%	n/a	37.82%	n/a	n/a	5.11	0.34
30	29.14%		37.80%			5.48	0.18
ICP							
8	57.22%		69.46%			532.72	66.59
15	52.16%	n/a	68.24%	n/a	n/a	1238.13	82.54
30	45.12%		67.62%			2977.00	99.23
No Cooperation							
8	57.30%		57.30%				
15	52.28%	n/a	52.28%	n/a	n/a	n/a	n/a
30	45.45%		45.45%				

scription of the workload models. Cooperation tends to reduce locality, and this motivates the results of Cache Digest that are even poorer than those of No-Cooperation.

If we include the overheads due to cooperation in the analysis of scalability and stability, we can observe some interesting modifications in the ranking of the cooperation schemes. Once again, we use Workload 2 (Table 5) as a main reference to explain our conclusions, and report data for Workload 1 (Table 6) as a comparison testbed. The last four columns of these tables show cooperation overheads that are measured as the ratio between the bytes exchanged for cooperation and the number of client connections (hits and misses) received by the edge servers, normalized by the number of cooperating edge servers in the last column.

As expected, augmenting the number of edge servers increases the cooperation overhead, but not every scheme is affected in the same way. In particular, ICP generates the highest cooperation traffic even with 8 nodes, and continues to increase with respect to the number of cooperating edge servers in a much faster way than any other schemes do. It is interesting to note that with 30

nodes the ICP cooperation overhead (traffic generated only to cooperative lookup purposes) reaches 4.5 KB per requests, with an average document size of about 10 KB. On the other hand, Cache Digests shows the lowest cooperation overhead.

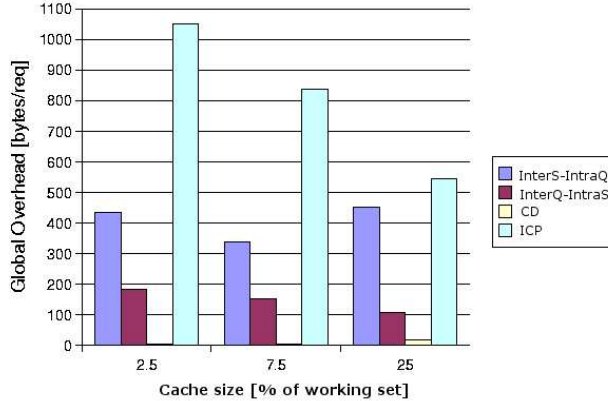
Hybrid schemes occupy an intermediate position, with a cooperation overhead higher than that of Cache Digest, but significantly lower than that of ICP. InterQ-IntraS halves the traffic of InterS-IntraQ, but the order of magnitude remains the same. It is more important to observe that the experimental results confirm the motivation of the InterS-IntraQ scheme. This approach is effective in reducing the usage of inter-group network resources, because it offers a summary-like traffic on the distant and more expensive links, while it moves more overheads on the intra-group links. The results observed for Workload 2 are substantially confirmed by the experiments based on Workload 1, as shown in the last four columns of Table 6. To appreciate the stability of the results of the cooperation schemes, we refer to the last columns of Tables 5 and 6, showing the overhead traffic per request relative to each node. It is important that for larger numbers of cooperating edge servers, InterQ-IntraS does not show practical modifications, the relative overheads of InterS-IntraQ and Cache Digests relative overheads tend to diminish, whereas continuous increments only occur for ICP.

7.2 Sensitivity analysis on cache capacity

Studying the sensitivity of the cooperation schemes with respect to the cache capacity evidences other interesting results, especially for the hybrid protocols. For these experiments we used the Workload 1, with only 8 nodes to reduce the advantages of the better scalability of two-tier cooperation. The considered cache capacity per node is equal to 2.5%, 7.5% and 25% of the working set size. As expected, the cache hit rate of all schemes increases with the cache capacity. With the highest capacity, InterQ-IntraS, InterS-IntraQ and ICP show close performance beyond 70%. Even Cache Digest improves over 50%, but this scheme remains penalized by the frequent object replacements.

The results related to the cooperation overhead, reported in Figure 6, are more interesting. Increasing the cache capacity brings two different effects: the local hit rate raises, but the digest size grows as well. The increase in local hit rates reduces the need of cooperation, so the overhead related to query-based cooperation decreases, whereas the overhead related to summary-based cooperation tends to augment, because of the bigger digests being exchanged. The overhead of Cache-Digest remains two orders of magnitude below that of ICP, but the increasing trend for the former and the decreasing trend for the latter was initially unexpected. Hybrid schemes, combining both types of coopera-

Fig. 6. Sensitivity of traffic overhead for cooperation to cache capacity



tion, have the most stable results with respect to cache capacity modifications, even if InterQ-IntraS performance tends to decrease more visibly.

7.3 Experiments on a geographic testbed

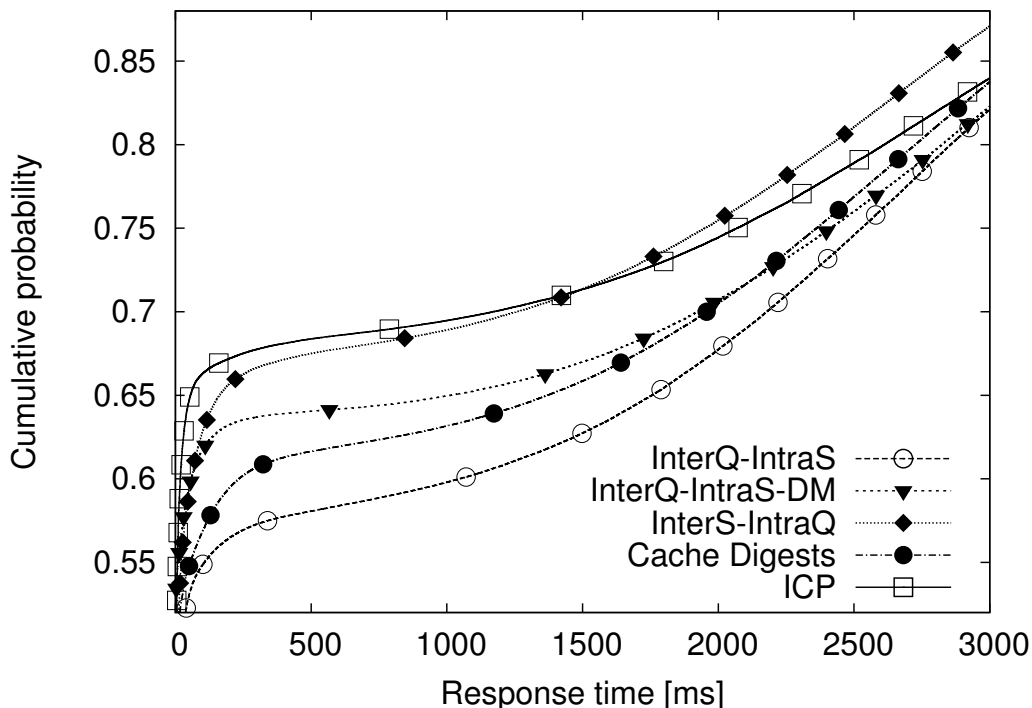
The last set of experiments was done in a geographic environment. The main goal is to verify whether the better scalability and stability of hybrid schemes (lower overheads than ICP with similar cache hit rates) have a negative influence on the user response time. We set up two groups of five nodes each, that were connected through some links and a geographic backbone. Ten network hops existed between the groups. We also installed one Web server in each location. For this test, we report the results referring to Workload 1, because they are representative of the other workload models as well. The experiments were performed with two different network conditions.

The first test was carried out on Sunday, with light network load, while the second test was done in conditions of heavy network load during a working day. The observed mean round-trip time was 50 ms and 300 ms, in the case of light and heavy network load, respectively.

We collected the data related to the client request service times from the Squid logs which can be used as an estimation for the user response time. We also monitored the network resource usage on the worst performing links. The worst, thus becoming the bottleneck, had a capacity of 2 Mbit/sec on one hop.

Figures 7 and 8 show the cumulative probability of the response times, for light and heavy network load, respectively. It is interesting to note the correlation between the cache hit rate and the user response time, that leads to the Bernoullian shape of the curves in Figures 7 and 8: requests are served very early or after a large amount of time.

Fig. 7. Response time (light network load)



Indeed, there is a big difference between client requests resulting in a hit (usually served in very short time) and requests occurring in a global miss (that usually experience a much higher latency). ICP shows the best response times for hit resources, but poorest results in the case of global misses because it has to wait for the slowest cooperating edge server. In Figure 7 the ICP curve is the highest at the beginning, but it is surpassed by that of InterS-IntraQ (around 1.5 seconds) and becomes the worst (after 4 seconds). These effects, even if already shown by our experiments, became much more evident in other not reported curves obtained for a workload model characterized by scarce locality and consequent lower hit rates. The conclusions on response times can be better appreciated by evaluating the 90-percentile of the request service time instead than looking at the curves that seem so close. As shown in Table 7, in the case of light network load, the 90-percentile for ICP is 3.7 seconds (the highest value), while for InterS-IntraQ it is 3.2 sec (the lowest value) and for InterQ-IntraS-DM it is 3.6 seconds. Cache Digests, because of its low hit rate tends to be slower in the first part of the curve, but the 90-percentile of its response time (3.5 seconds) is similar to that of the other cooperation protocols.

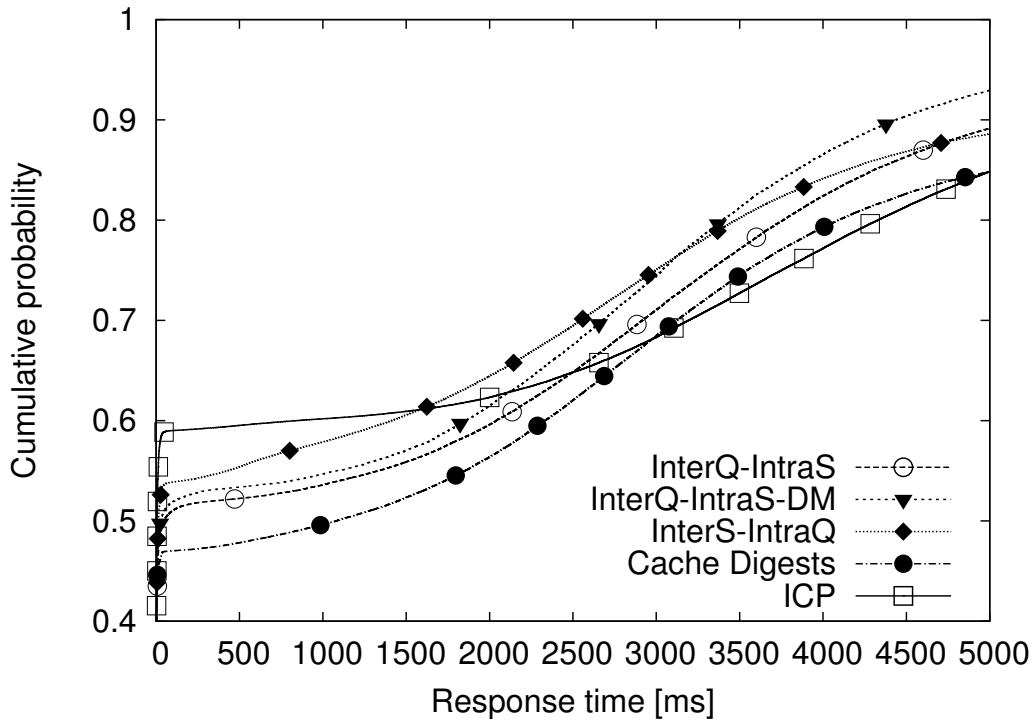
When the network is affected by heavy traffic (Figure 8), congestion can occur in many places of the architecture. The overall performance of the cooperative system is reduced: the percentage of client requests serviced within 100 ms is reduced from 66%-56%, depending on the cooperation model used, to 59%-47%. Furthermore, the cache hit rate decreases (from 68%-47% to 61%-

Table 7

90 percentile of response times in seconds

Protocol	Light network load [sec]	Heavy network load [sec]
InterQ-IntraS	3.60	5.19
InterQ-IntraS-DM	3.68	4.43
InterS-IntraQ	3.26	5.63
Cache Digests	3.51	7.30
ICP	3.70	6.23

Fig. 8. Response time (heavy network load)



36%) in conditions of heavy network load. The differences among the various hybrid schemes are reduced because the congestion occurring in the query-based cooperation overpowers the congestion of InterQ-IntraS protocol. As shown in Table 7 also the 90-percentile of the user response time augments: ICP shows once again poor performance (6.2 seconds), far more than that of the best InterQ-IntraS-DM remaining below 4.5 seconds. Cache Digests is the worst cooperation protocol in terms of 90-percentile (7.3 sec) when the network is congested. The motivation is that its lower hit rate augments the use of congested links to contact the Web servers. The other hybrid schemes, InterQ-IntraS and InterS-IntraQ, have an intermediate result with 90-percentile equal to 5.2 and 5.6 seconds, respectively.

The InterQ-IntraS scheme, that has shown the best results in the previous

sections referring to LAN-based experiments, seems the worst of the three hybrid schemes in a geographic scenario. Collecting user response time statistics in a LAN environment is senseless, while it is very useful in a WAN environment. That is why we notice the congestion occurring at the group masters only in this latter series of experiments. We can see from Figure 8 that, when the group masters also act as edge servers, about 15% of resources are served after a long time (up to 5 seconds). This problem is solved by the InterQ-IntraS-DM variant with dedicated group masters. Indeed, even with a smaller number of edge servers, InterQ-IntraS-DM improves the performance of the hybrid schemes: in the case of heavy network load, its 90-percentile is equal to 4.4 seconds, while the corresponding index is equal to 5.2 and 5.6 seconds for InterQ-IntraS and InterS-IntraQ, respectively, as shown in Table 7. Actually, this bad result for InterS-IntraQ is mainly due to its lower cache hit rate that, like for Cache Digests, is more affected by the characteristics of the workload considered in this experiment. InterQ-IntraS-DM is the best cooperation protocol in terms of 90 percentile in the case of a heavy network load. However, it is worth to observe that especially when the cache hit rate has a lower impact on response time (as it is for the case of light network load), InterS-IntraQ is a valid alternative. Its curve is often over the others in Figure 7, and even better than InterQ-IntraS-DM in some circumstances (see the first part of Figure 8). It degrades when its low cache hit rate influences the percentage of objects that must be retrieved from the origin Web server.

7.4 *Summary of the experimental results*

A large set of experiments, of which only a significant subset has been reported here, highlights some interesting results.

- Query-based protocols seem very effective in resource discovery because their mechanism is less subject to stale information. However, their high cache hit rates are affected by cooperation overheads that grow as the number of cooperating edge servers increases, and by traffic especially over distant and congested links. The tests in a geographic environment have evidenced the poor stability of results based on query mechanisms: they have low response time in case of hits, but they are much slower than other protocol in handling misses.
- Summary-based protocols cause negligible overhead for cooperation, but their cache hit rates are very sensitive to many workload and architecture parameters. For example, when the frequency of client requests is high and/or there is a frequent object replacement, it is not convenient to use summary-based protocols for cooperative discovery. Even in a geographic environment, their performance is mainly affected by low hit rates.
- Hybrid protocols achieve intermediate results, with hit rates close to query-

based protocols (especially for the InterQ-IntraS and InterQ-IntraS-DM schemes), and overheads growing much slower than those of query-based protocols as the number of cooperating edge servers increases. This twofold effect allows us to conclude that they have the best potential scalability. Tests in the geographic scenario have confirmed this hypothesis: two-tier protocols show good ability in dealing with both hit and miss objects. Their stable performance in the highly variable Web scenario is the most important merit of these hybrid schemes.

A final remark is in order about the general applicability of these results. The three main workload models considered in this paper derive from a careful selection of many other results. They have been included here because they bring the most important conclusions and certainly do not favor hybrid schemes. On the opposite, they tend to favor query-based cooperation with respect to summary-based cooperation. Actually, we can say that different workload characteristics never contributed to improve ICP performance shown here.

Other workload models can only improve summary-based performance (e.g., less frequent object replacements in caches, higher inter-arrival times for client requests) or deteriorate query-based results especially for the user response time (e.g., lower recurrence of requests leading to reduced local hit rates). The real important point is that the results of the hybrid schemes have been demonstrated to be more independent of the workload models, because they can backup the defacement of a cooperation protocol in the first-tier lookup with the improvement of the other protocol in the second-tier lookup.

8 Related work

Cooperative Web caching has been studied for a while. Cooperation can be used for different goals, but here we are more interested to resource discovery and delivery (or retrieval). The first idea for cooperation was the use of a hierarchy based on the physical Internet topology (8). In this approach, the phases of discovery and delivery are simultaneous. Hierarchical cooperation shows, however, multiple drawbacks especially for the lookup latency (11), and management issues (27).

Some of the problems of hierarchical caching can be solved by organizing the edge servers in a flat topology. Typically, in this approach the two actions of discovery and delivery are separated, as in the systems considered in this paper. Document retrieval for delivery is extremely simple and efficient, by requiring only an HTTP connection to the selected edge server or to the origin Web server. On the other hand, discovery can be performed according to

different schemes (namely distributed lookup schemes). In pure distributed schemes, the lookup process takes place in one step and is basically founded on query-based or directory (summary)-based mechanisms. In our hybrid distributed schemes, the lookup process takes place in two separate steps and is founded on a combination of query-based and summary-based mechanisms.

ICP (23) is the most popular query-based cooperation protocol and its pros and cons have been widely studied over times. The limited scalability of ICP has been documented in (15) by the author of the protocol himself, while its high overhead for cooperation was one of the main motivation for the proposal of a summary-based scheme, such as Summary Cache (13). In (20), Russkov et al. also note that ICP is slow in handling misses, thus motivating the proposal of Cache Digests, another well known summary-based protocol. Multicast has been proposed (for example (28) for ICP) to reduce cooperation overhead. However, this technique has serious limits to be applied to a geographically distributed system, because by now multicast addresses cannot usually cross the boundaries of autonomous systems.

In our paper we intensively tested these “pure” protocols and found a confirmation of the fact that they are not scalable. ICP because of the high overheads it puts on the network, Cache Digests because of its low hit rates.

As “pure” hierarchical and distributed approaches were found unsuitable for cooperation over a large scale, hybrid schemes have been proposed. Here, a hierarchical approach is combined with other forms of cooperation among sibling edge servers (that is, nodes sharing the same parent edge server). *Hybrid scheme* is a quite generic term that is usually referred to any cooperation mechanism that does not fit in the category of pure hierarchical and pure distributed cooperation. However, there are many possible combinations that derive from the lookup topology and the cooperative protocol. For example, an all-query-based scheme is proposed in (29), where traditional hierarchical caching is integrated with horizontal query-based cooperation among cooperating edge servers at the same level of the hierarchy. Tewari et al. (12) suggest a summary-based hierarchical approach by means of metadata that track where copies of files are stored in the hierarchy. A similar technique is proposed by Povey et al. (27), where only the lower level edge servers are responsible for storing documents, while upper level nodes maintain information about the contents of the lower level edge servers. The idea of a multi-step lookup over sets of different cardinality is related to the idea of two-step cooperation over a two-tier architecture proposed in this paper. The main difference with the results from our contribution is that the previous proposals rely on one “pure” lookup protocol, that is either query-based or directory(summary)-based, whereas we propose various hybrid combinations of protocols.

CRISP is another hybrid scheme in terms of protocols, proposed by Rabi-

novich et al. (30; 9). It combines a centralized directory-based protocol with a query-based approach. On the other hand, all the schemes proposed here rely on a distributed summary-based protocol. Indeed, the authors of CRISP observed that a centralized directory makes their architecture not scalable over a geographic network environment, although they outlined some alternative solutions for scalability (31). In particular, the two-step lookup process and node clustering choices have some similarities with our two-tier architecture, and especially with the InterQ-IntraS-DM scheme. The good performance of InterQ-IntraS-DM can be considered a demonstration of the validity of the ideas contained in (31).

Another important topic related to Web caching is the definition of measures of quality for Web caching and cooperative Web caching. Unlike many other papers that use a limited set of measures, we consider all the main metrics appeared in literature, such as cache hit rate, response time, overhead for cooperation. In all previous studies, object (or byte) hit rate is the main metric used to evaluate performance of a caching architecture. More recently, several authors have pointed out the importance of considering other performance metrics. For example, Fan et al. (13) introduce network overhead as a measure for system scalability, and the impact of cooperation on central memory and CPU usage. However, present architectures do not seem anymore to represent a bottleneck for edge servers with the exception of mass storage size for nodes holding large resources such as video data (24).

The user response time is another important measure of cache performance. For example, Russkov et al. (20) use this parameter to demonstrate that Cache Digests is preferable to ICP. In our paper the user response time is the most important metric to evaluate the performance of the hybrid protocols in a geographic environment.

Finally, we consider important to observe that our prototype experiments based on Web Polygraph (21) allowed us the highest flexibility on the choices of workload characteristics. We carried out important sensitivity analysis that are prevented to the large majority of papers based on trace-driven simulations. Indeed, we are aware of few sensitivity analysis of cooperative caching performance to workload characteristics. Williamson et al. focus on traditional proxy caching (32), while some results on cooperative Web caching is in (24).

9 Conclusions

The contribution of this paper is twofold.

- We show the limits of scalability and stability of pure query-based and

summary-based protocols for cooperative document discovery, and the strong dependency of their performance on different workload and architecture characteristics.

- We propose three hybrid cooperation mechanisms (*InterQ-IntraS*, *InterQ-IntraS-DM* and *InterS-IntraQ*) that combine in different ways query-based and summary-based protocols in a two-tier architecture.

Two-tier protocols obtain good hit rates and low overheads, without penalizing the user response times. Indeed, evaluating the 90 percentile of the user response times, these protocols are even better than the “pure” ones. Furthermore they are particularly stable and much more insensitive to network conditions than “pure” protocols do. These characteristics let the proposed protocols scale well even when there are many cooperating edge servers.

References

- [1] IRCache, Ircache project, – <http://www.ircache.net> (1995).
- [2] NLANR, National laboratory for applied network research, – <http://www.nlanr.net> (2002).
- [3] ATT, At&t, – <http://www.att.com> (2002).
- [4] Akamai, Akamai inc., – <http://www.akamai.com> (2002).
- [5] Speedera, Speedera inc., – <http://www.speedera.com> (2002).
- [6] B. Krishnamurthy, C. Wills, Y. Zhang, On the use and performance of content distribution networks, in: Proc. Internet Measurement Workshop, ACM SIGCOMM, 2001.
- [7] D. Wessels, Squid Programmers Guide (2002).
- [8] A. Chankhunthod, M. Schwartz, P. Danzig, K. Worrell, C. Neerdaels, A hierarchical internet object cache, in: Proc of Usenix Annual Technical Conference, 1995.
- [9] S. Gadde, J. Chase, M. Rabinovich, A taste of crispy squid, in: Proc. of Workshop on Internet Server Performance (WISP’98), 1998.
- [10] P. S. Yu, E. A. MacNair, Performance study of a collaborative method for hierarchical caching in proxy servers, Computer Networks and ISDN Systems (1998) 215–224.
- [11] P. Rodriguez, C. Spanner, E. Biersack, Web caching architectures: hierarchical and distributed caching, in: Proc. of Web Caching Workshop (WCW’99), 1999.
- [12] R. Tewari, M. Dahlin, H. Vin, J. Kay, Beyond hierarchies: design considerations for distributed caching on the internet, in: Proc. 19th International Conference on Distributed Computing Systems, IEEE, 1999.
- [13] L. Fan, P. Cao, J. Almeida, A. Z. Broder, Summary cache: A scalable wide-area web cache sharing protocol, in: Proc. of SIGCOMM ’98, 1998.
- [14] S. Gadde, J. Chase, M. Rabinovich, Directory structures for scalable internet caches, Tech. rep., Dept. of Computer Science, Duke University (1997).
- [15] D. Wessels, K. Claffy, Icp and the squid web cache (Aug. 1997).
- [16] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, H. M. Levy,

- On the scale and performance of cooperative web proxy caching, in: Proc. Symposium on Operating System Principles, 1999.
- [17] R. Lancellotti, F. Mazzoni, M. Colajanni, Scalability of cooperative algorithms for distributed architectures of proxy servers, in: Proc. of IADIS International WWW/Internet 2003 Conference, 2003.
 - [18] P. McManus, A passive system for server selection within mirrored resource environments using as path length heuristics, – <http://proximate.appliedtheory.com/> (Apr. 1999).
 - [19] F. S. K. Obraczka, Looking at network latency for server proximity, in: Proc. of the IEEE Globecom 2000, 2000.
 - [20] A. Rousskov, D. Wessels, Cache digests, Computer Networks and ISDN Systems 30 (22-23).
 - [21] A. Russkov, D. Wessels, Web polygraph, – <http://www.web-polygraph.org> (2000).
 - [22] P. Danzig, Nocache architecture and deployment, in: Proc. of 3rd W3 Cache Workshop, 1997.
 - [23] D. Wessels, K. Claffy, Internet cache protocol (icp), version 2, RFC 2186 (Sep. 1997).
 - [24] K. W. Lee, K. Amiri, S. Sahu, C. Venkatramani, On the sensitivity of cooperative caching performance to workload and network characteristics, in: Proc. of ACM Sigmetrics, 2002.
 - [25] A. Papoulis, S. U. Pillai, Probability, Random Variables and Stochastic Processes, McGraw-Hill, 2001.
 - [26] B. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (1970) 422–426.
 - [27] D. Povey, J. Harrison, A distributed internet cache, in: Proc. 20th Australian Computer Science Conference, Sydney, Australia, 1997.
 - [28] D. Wessels, K. Claffy, Application of internet cache protocol (icp), version 2, RFC 2187 (Sep. 1997).
 - [29] P. Rodriguez, C. Spanner, E. Biersack, Analysis of web caching architectures: hierarchical and distributed caching, IEEE/ACM Transactions on Networking.
 - [30] S. Gadde, M. Rabinovich, J. Chase, An approach to building large internet caches, in: Proc. Sixth Workshop on Hot Topics in Operating Systems (HotOS-VI), 1997.
 - [31] M. Rabinovich, J. Chase, S. Gadde, Not all hits are created equal: Cooperative proxy caching over a wide-area network, in: Proc. of Third International WWW Caching Workshop, 1998.
 - [32] C. Williamson, M. Busari, On the sensitivity of web proxy cache performance to workload characteristics, in: Proc. of IEEE Infocom, 2001.