Analysis of Web Caching Architectures: Hierarchical and Distributed Caching

Pablo Rodriguez, Christian Spanner, and Ernst W. Biersack, *Member, IEEE*

***Abstract*—Cache cooperation improves the performance of isolated caches, especially for caches with small cache populations. To make caches cooperate on a large scale and effectively increase the cache population, several caches are usually federated in caching architectures. In this paper, we discuss and compare the performance of different caching architectures. In particular, we consider hierarchical and distributed caching. We derive analytical models to study important performance parameters of hierarchical and distributed caching, i.e., client's perceived latency, bandwidth usage, load in the caches, and disk space usage. Additionally, we consider a hybrid caching architecture that combines hierarchical caching with distributed caching at every level of a caching hierarchy. We evaluate the performance of a hybrid scheme and determine the optimal number of caches that should cooperate at each caching level to minimize client's retrieval latency.**

***Index Terms*—Caching, performance, web.**

I. Introduction

THE WORLD Wide Web (WWW) provides simple access to a wide range of information and services. As a result, the Web has become the most successful application in the Internet. However, the exponential growth in demand experienced during the last years has not been followed by the necessary upgrade in the network/servers; therefore, clients experience frustrating delays when accessing Web pages. In order to keep the Web attractive, the experienced latencies must be maintained under a tolerable limit. One way to reduce the experienced latencies, the server's load, and the network congestion is to store multiple copies of the same Web documents in geographically dispersed Web caches.

As most of the Web content is rather static, the idea of Web caching is not new. The first WWW browsers, e.g., Mosaic [10], were able to cache Web objects for later reference, thus reducing the bandwidth used for Web traffic and the latency to the users. Web caching rapidly extended from a local cache used by a single browser, to a shared cache serving all the clients from a certain institution. Unfortunately, since the number of clients connected to a single cache can be rather small and the amount of information available in the Web is rapidly increasing, the cache's performance can be quite modest. The hit rate, i.e., the percentage of requests that can be served from previously cached document copies, at an institutional cache typically ranges between 30% and 50% [26].

The hit rate of a Web cache can be increased significantly by sharing the interests of a larger community [9]; the more people are accessing the same cache, the higher the probability that a given document is present in the cache. To increase the effective client population using a cache, several caches can cooperate. Two common approaches to implement a large scale cache cooperation scheme are ***hierarchical*** [9] and ***distributed*** [23], [30] caching.

With *hierarchical caching* caches are placed at different network levels. At the bottom level of the hierarchy there are client caches. When a request is not satisfied by a client cache, the request is redirected to the institutional cache. If the document is not present at the institutional level, the request travels to the regional cache, which in turn forward unsatisfied requests to the national cache. If the document is not present at any cache level, the national cache contacts directly the origin server. When the document is found, either at a cache or at the origin server, it travels down the hierarchy, leaving a copy at each of the intermediate caches. Further requests for the same document travel up the caching hierarchy until the request finds the document. There are several problems associated with a caching hierarchy: 1) every hierarchy introduces additional delays [30], [9]; 2) higher level caches may become bottlenecks and have long queuing delays; and 3) several copies of the same document are stored at different cache levels.

With *distributed caching*, no intermediate caches are set up, and there are only institutional caches at the edge of the network that cooperate to serve each others' misses. Since there are no intermediate caches that store and centralize all documents requested by lower level caches, institutional caches need other mechanisms to share the documents they contain. Some of these mechanisms are the following.

- Institutional caches can query the other cooperating institutional caches for documents that resulted in local misses (this is usually done using the Inter Cache Protocol ICP [34]). However, using a query-based approach may significantly increase the bandwidth consumption and the experienced latency by the client since a cache needs poll all cooperating caches and wait for the slowest one to answer.
- Institutional caches can keep a digest [27] or summary [12] of the content of the other cooperating caches, thus avoiding the need for queries/polls. Content digests/summaries are periodically exchanged among the institutional caches. To make the distribution of the digest/summary more efficient and scalable, a hierarchical infrastructure

Manuscript received June 8, 1999; revised May 1, 2000; recommended by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Ammar. The work of P. Rodriguez was supported by a Training and Mobility for Researchers (TMR) fellowship from the European Commission. This work was partially supported by EURECOM's industrial partners Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

The authors are with Institut EURECOM, 06904 Sophia Antipolis Cedex, France (e-mail: erbi@eurecom.fr).

Publisher Item Identifier S 1063-6692(01)06852-2.

of intermediate nodes can be used [23], [30]. However, this hierarchical infrastructure only distributes information about the location of the documents but does not store document copies.

- Institutional caches can cooperate using a hash function [31], [16] that maps a client request into a certain cache. With this approach, there are no duplicated copies of the same document in different caches and there is no need for caches to know about each other's content. However, having only one single copy of a document among all cooperating caches limits this approach to local environments with well-interconnected caches.

Distributed caching, as well as hierarchical caching, are already a fact of life in much of the Internet [5]. In the U.S., NLANR provides a hierarchical caching architecture to handle highly popular information [3]. In Europe, many countries have also deployed caching hierarchies to reduce the number of requests that traverse the highly congested transoceanic links [3], [28]. Distributed caching is used in several cache cooperating schemes [34], [27], [31], and several researchers have proposed to deploy a large scale distributed caching cooperation [23]. Moreover, distributed caching has recently become very relevant with the appearance of new applications that allow the distribution of different content (Web pages, images, music) using only end-host caches [2], [1].

In this paper, we develop analytical models to study and compare the performance of both hierarchical and distributed caching. We derive models to calculate the latency experienced by the clients, the bandwidth usage, the disk space requirements, and the load generated by each cache cooperating scheme. Using analytical models, we can explore the different tradeoffs of the different cache cooperating schemes and simulate different scenarios. To make our model as realistic as possible, we take a set of reasonable assumptions and parameters from recent literature and try to validate our results with real data when possible.

Regarding latency, we find that hierarchical caching has lower connection times than distributed caching. Thus, caching document copies at the access points of intermediate Internet Service Providers (ISPs) reduces the connection time compared to the case where there is no support from intermediate caches in the network and there are only edge caches. We also find that distributed caching has lower transmission times than hierarchical caching, since most of the traffic flows through the less congested lower network levels. In addition to the latency analysis, we study the bandwidth used by hierarchical and distributed caching. We find that hierarchical caching has lower bandwidth usage than distributed caching, since hierarchical caching uses intermediate caches in the network. A hierarchical caching scheme that uses support from intermediate caches in the network mimics a multicast distribution at the application level [25] and is much more efficient in terms of bandwidth than a distributed caching scheme that only places caches at the edge of the network. However, distributed caching distributes the traffic better, using more bandwidth in lower network levels. Further analysis of hierarchical and distributed caching shows that the disk requirements for distributed caching are much smaller than for hierarchical caching. More precisely, we find that an institutional cache only needs several gigabytes to store all accessed documents, while a top-level cache of a caching hierarchy requires hundreds of gigabytes to satisfy the needed capacity. We also find that distributed caching shares very well the total load of the system and does not generate hot spots with high load, as may be the case for hierarchical caching.

In addition to hierarchical and distributed caching, we also study and compare the performance of a hybrid scheme where caches cooperate at every level of a caching hierarchy using distributed caching [34], [3]. Our analysis of the hybrid scheme shows that the latency experienced by clients greatly varies depending on the number of caches that cooperate at every network level. Based on analytical results, we determine the optimal number of caches that should cooperate at every network level to minimize latency experienced by clients. We find that a hybrid scheme with an optimal number of cooperating caches at every level improves the performance of hierarchical and distributed caching, reducing latency, bandwidth usage, and load in the caches.

The rest of the paper is organized as follows. We first discuss some previous work and different approaches to hierarchical and distributed caching. In Section II, we describe our specific model for analyzing hierarchical and distributed caching. In Section III, we provide latency analysis for hierarchical and distributed caching. In Section IV, we present a numerical comparison of both caching architectures and also consider bandwidth, disk space, and cache's load. In Section V, we analyze the hybrid scheme. In Section VI, we summarize our findings and conclude the paper.

A. Related Work

Hierarchical Web caching was first proposed in the Harvest project [9] to share the interests of a large community of clients and has already been implemented in several countries [3]. In the context of distributed caching, NLANR designed the Internet Cache Protocol (ICP) [34] and the HTCP [32] protocol, to support discovery, retrieval, and management of documents from neighboring caches, as well as parent caches. Another approach to distributed caching is the Cache Array Routing Protocol (CARP) [31], which divides the URL space among an array of loosely coupled caches and lets each cache store only the documents whose URL hashes to it. In [16], Karger *et al.* propose a similar approach to CARP that uses DNS servers to resolve the URL space and allows replication of popular content in several caches. Povey and Harrison also proposed a large-scale distributed Internet cache [23], in which upper level caches are replaced by directory servers that contain location hints about the documents kept at every cache. A hierarchical infrastructure is used to facilitate a scalable distribution of these location hints. Tewari *et al.* propose a similar approach to implement a fully distributed Internet cache where location hints are replicated locally at the institutional caches [30]. In the central directory approach (CRISP) [14], a central mapping service ties together a certain number of caches. In Summary Cache [12], Cache Digest [27], and the Relais project [19], caches exchange messages indicating their content, and keep local directories to facilitate finding documents in other caches.

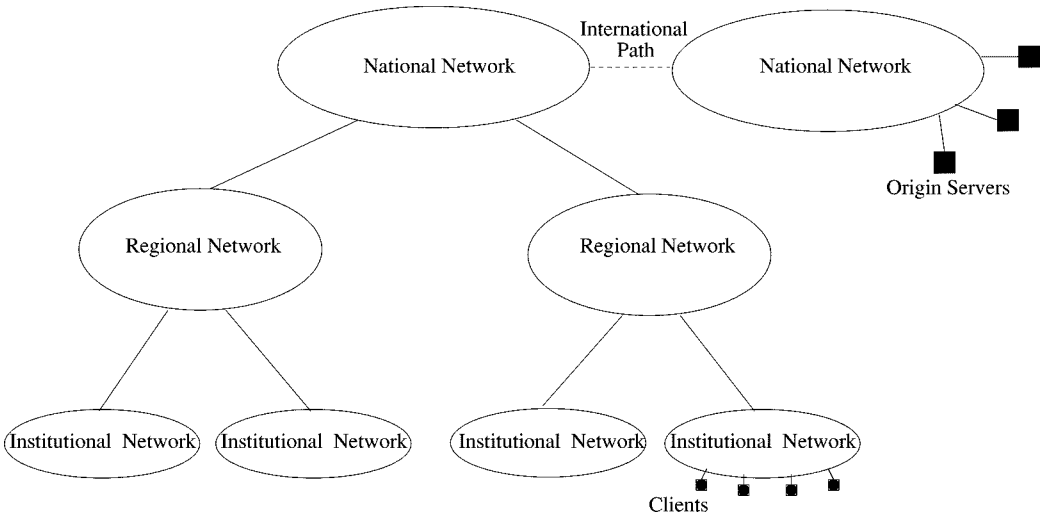


Fig. 1. Network topology.

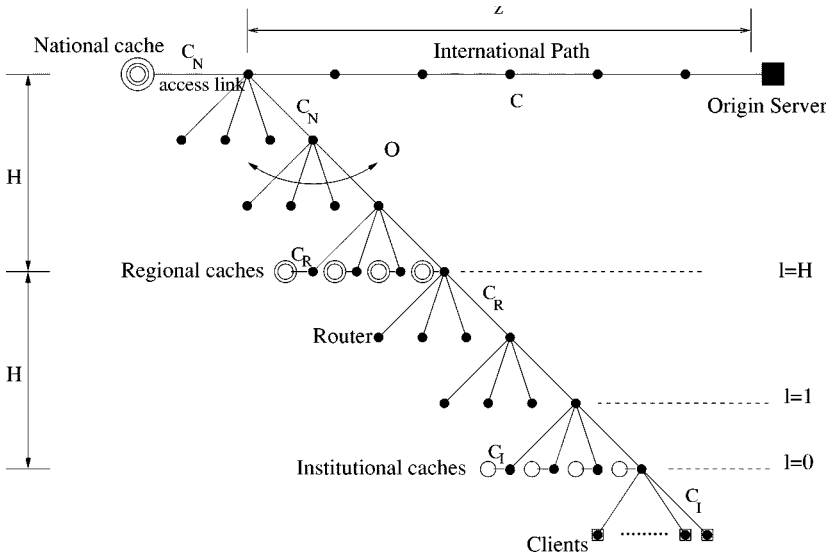


Fig. 2. Tree mode, showing cache placement.

Concerning a hybrid scheme, ICP [34] allows for cache cooperation at every-level of a caching hierarchy. Rabinovich *et al.* [24] proposed to limit the cooperation between neighbor caches to avoid fetching documents from distant or slower caches, if they could have been retrieved directly from the origin server at a lower cost.

II. The Model

A. Network Model

As shown in Fig. 1, the Internet connecting the server and the receivers can be modeled as a hierarchy of ISPs, each ISP with its own autonomous administration.

We shall make the reasonable assumption that the Internet hierarchy consists of three tiers of ISPs: institutional networks, regional networks, and national backbones. All clients are connected to the institutional networks; institutional networks are connected to the regional networks; regional networks are connected to national networks. National networks are also connected, sometimes by transoceanic links. We shall focus on a model with two national networks, with one of the national networks containing all of the clients and the other national network containing the origin servers.

We model the underlying network topology as a full O-ary tree, as shown in Fig. 2. Let O be the nodal outdegree of the tree. Let H be the number of network links between the root node of a national network and the root node of a regional network. H is also the number of links between the root node of a regional network and the root node of an institutional network. Let z be the number of links between a origin server and root node (i.e., the international path). Let l be the level of the tree. $0 \leq l \leq 2H + z$, where $l = 0$ is the institutional caches and $l = 2H + z$ is the origin server. We assume that bandwidth is homogeneous within each ISP, i.e., each link within an ISP has the same transmission rate. Let C_I, C_R, and C_N be the *transmission rate* of the links at the institutional, regional, and national networks. Let C be the bottleneck rate on the international path.

B. Document Model

Denote N as the total number of documents in the WWW. Denote S as the size of a certain document. We assume that documents change periodically every update period Δ. Requests for document i, $1 \leq i \leq N$ in an institutional cache are Poisson distributed with average request rate $\lambda_{I,i}$. Therefore, the total request rate for document i is $\lambda_{\text{tot},i} = \lambda_{I,i} \cdot O^{2H}$, which is also Poisson distributed. We consider that each document is requested independently from other documents, so we are neglecting any source of correlation between requests of different documents. Let β_I be the request rate from an institutional cache for all N documents, $\beta_I = \sum_{i=1}^{N} \lambda_{I,i}$. β_I is Zipf distributed [7], [35], that is, if we rank all N documents in order of their popularity, the ith most popular document has a request rate $\lambda_{I,i}$, given by

$$\lambda_{I,i} = \beta_I \frac{\sigma}{i^{\alpha}},$$

where α is a constant that determines how skewed the Zipf distribution is, and σ is given by

$$\sigma = \left(\sum_{i=1}^{N} \frac{1}{i^{\alpha}} \right)^{-1}.$$

The Zipf distribution will be more or less skewed depending on how homogeneous the client population is; the more homogeneous the client population is, the higher the skew factor α. Therefore, by tuning the parameter α, we can model communities of different degrees of heterogeneity. At the regional and national caches, requests for very popular documents are filtered by intermediate-level caches.

C. Hierarchical Caching

Caches are usually placed at the access points between two different networks to reduce the cost of traveling through a new network. As shown in Fig. 2, we make this assumption for all of the network levels. In one country there is one national network with one national cache. There are O^H regional networks and every one has one regional cache. There are O^{2H} local networks and every one has one institutional cache.

Caches are placed on height 0 of the tree (level 1 in the cache hierarchy), height H of the tree (level 2 in the cache hierarchy), and height $2H$ of the tree (level 3 in the hierarchy). Caches are connected to their ISP's via *access links*. We assume that the capacity of the access link at every level is equal to the network link capacity at that level, i.e., C_I, C_R, C_N, and C for the respective levels. The hit rate for documents at the institutional, regional, and national caches is given by hit_I, hit_R, hit_N, and is defined as that percentage of requests found at a given cache level or at any cache level below.

D. Distributed Caching

In the distributed caching scheme, caches are only placed at the institutional level of Fig. 2 and no intermediate copies are stored in the network. Institutional caches keep a copy of every document requested locally. To share local document copies among different institutional caches, institutional caches periodically exchange metadata information about the documents that they keep. We assume that metadata information is instantaneously updated at every institutional cache every time that a new document is fetched in any cache.

E. Properties and Limitations of the Model

The model presented up to now tries to make a set of reasonable assumptions to reflect some of the most important parameters that influence the performance of a cache cooperating scheme. The fact that we use an analytical model instead of trace-driven or real simulations has some drawbacks but also several advantages that we try to outline next.

- The full O-ary tree used to model the network is the base of our latency and bandwidth analysis. O-ary trees have been found to be good models for the Internet [21], [22]. However, the latency and bandwidth results presented in this paper should not be considered as absolute results but rather as relative results to compare the relative behavior of different cache cooperation protocols.
- The network model assumes a hierarchical topology of the Internet, with different tiers of ISPs. This may not be always the case, since there are situations where clients are directly attached to high-level ISPs or origin servers are directly connected to institutional ISPs. However, a careful reader should note that these scenarios can be easily modeled by modifying the height or the number of tiers of the distribution tree and changing the analysis accordingly.
- The assumptions used to calculate the load in the caches, the hit rates, and the disk space, are not based on any tree model for the network. They are only based on well-known probabilistic distributions taken from the literature (e.g., Poisson requests for the same document [15], Zipf distribution). Thus, the results obtained for the load in the caches, the hit rates, and disk space are very close to those ones reported with real data.
- To compare hierarchical and distributed caching in terms of latency and bandwidth we assume homogeneous client communities. While this may be considered as a limitation of our model, in Section IV-D we also show how slight modifications of the analysis can easily model heterogeneous client communities. For other important performance parameters, such as the hit rate and the disk space, we vary the skew factor of the Zipf distribution to analyze the impact of heterogeneous client communities.

III. Latency Analysis

In this section, we model the expected latency to obtain a document in a caching hierarchy and in a distributed caching scheme. We use a similar analysis to the one presented in [25]. The total latency T to fetch a document can be divided into two parts, the *connection time* T_c and the *transmission time* T_t. The connection time T_c is the time since the document is requested by the client and the first data byte is received. The transmission time T_t is the time to transmit the document. Thus, the average total latency is given by

$$E[T] = E[T_c] + E[T_t].$$

A. Connection Time

The connection time depends on the number of network links from the client to the cache containing the desired document copy. Let L_i be the number of links that a request for document i travels before it is satisfied in the caching hierarchy or in the origin server. We assume that the operating system in the cache gives priority at establishing TCP connections compared to other already existing TCP connections. Let d denote the per-hop propagation delay. The connection time in a caching hierarchy T_c^h is given by

$$E[T_c^h] = 4d \sum_{l \in \{0,\, H,\, 2H,\, 2H+z\}} P(L_i = l)(l+1)$$

where the $4d$ term is due to the three-way handshake of a TCP connection that increases the number of links traversed before any data packet is sent. To account for the distance between the client and the institutional cache, we include one more link in the connection time.

Now, let L_i be the network level such that the tree rooted at level L_i is the smallest tree containing copy of document i. The connection time in distributed caching T_c^d is given by

$$E[T_c^d] = 4d \cdot \sum_{l=0}^{2H} P(L_i = l) \cdot (2l+1) + 4d \cdot P(L_i = 2H+z) \cdot (2H+z+1).$$

In distributed caching, a request first travels up to network level l and then down to the institutional cache with a document copy, thus accounting for $2l$ links.

We now calculate the distribution of L_i, which is the same for hierarchical and distributed caching. To obtain $P(L_i = l)$, we use $P(L_i = l) = P(L_i \geq l) - P(L_i \geq l+1)$. Note that $P(L_i \geq l)$ is the probability that the number of links traversed to meet the document is equal to l or higher. To calculate $P(L_i \geq l)$, let τ denote the time into the interval $[0, \Delta]$ at which a request occurs. The random variable τ is uniformly distributed over the interval, thus we have

$$P(L_i \geq l) = \frac{1}{\Delta} \int_0^{\Delta} P(L_i \geq l | \tau)\, d\tau \tag{1}$$

where $P(L_i \geq l|\tau)$ is the probability that there is no request for document i in the subtree rooted at level $l-1$ during the interval $[0, \tau]$

$$P(L_i \geq l|\tau) = e^{-O^{l-1}\lambda_{I,i} \cdot \tau}. \tag{2}$$

Combining (1) and (2), we get

$$P(L_i \geq l) = \frac{1}{O^{l-1}\lambda_{I,i} \cdot \Delta} (1 - e^{-O^{l-1}\lambda_{I,i} \cdot \Delta}). \tag{3}$$

B. Transmission Time

We now calculate the transmission time to send a document in a hierarchical caching versus distributed caching. The transmission time of a document depends on the network level L_i up to which a request travels. Requests that only travel through low network levels will experience low transmission times. Requests that travel up to high network levels will experience large transmission times. We make the realistic assumption that the caches operate in a cut-through mode, rather than a store-and-forward mode, i.e., when a cache begins to receive a document, it immediately transmits the document to the subsequent cache (or client) while the document is being received. We expect capacity misses to be a secondary issue for large-scale cache architectures because it is becoming very popular to have caches with huge effective storage capacities. We, therefore, assume that each cache has infinite storage capacity.

We now proceed to calculate the transmission time for hierarchical caching $E[T_t^h]$, and the transmission time for distributed caching $E[T_t^d]$. $E[T_t^h]$ and $E[T_t^d]$ are given by

$$E[T_t^h] = \sum_{l \in \{0,\, H,\, 2H,\, 2H+z\}} E[T_t^h | L_i = l] \cdot P(L_i = l)$$

$$E[T_t^d] = \sum_{l=0}^{2H+z} E[T_t^d | L_i = l] \cdot P(L_i = l)$$

where $E[T_t^h|L_i = l]$ and $E[T_t^d|L_i = l]$ are the expected transmission times at a certain network level for hierarchical and distributed caching. To calculate $E[T_t^h|L_i = l]$ and $E[T_t^d|L_i = l]$, we first determine the aggregate request arrival rate at every network level l for hierarchical caching β_l^h and for distributed caching β_l^d.

For hierarchical caching, the aggregate request arrival rate at every network level β_l^h is filtered by the hit rates at the lower caches. Thus, the aggregate request arrival rate generated by hierarchical caching at a link between the levels l and $l+1$ of the network tree is given by

$$\beta_l^h = \begin{cases} \beta_I, & l = 0 \\ O^l \beta_I \cdot (1 - hit_I), & 0 < l < H \\ O^l \beta_I \cdot (1 - hit_R), & H \leq l < 2H \\ O^{2H} \beta_I \cdot (1 - hit_N), & 2H \leq l < 2H + z. \end{cases}$$

Hit rates at every network level can be calculated using the popularity distribution of the different documents, (i.e., Zipf) and the distribution of L_i

$$hit_l = \sum_{i=1}^{N} \left(\frac{\lambda_{I,i}}{\beta_I} \cdot P(L_i \leq l) \right).$$

For distributed caching, the aggregate request arrival rate at a link between levels l and $l+1$ is filtered by the documents already hit in any institutional cache belonging to the subtree rooted at level l, hit_l. Furthermore, in distributed caching, the traffic between levels l and $l+1$ increases due to the percentage of requests not satisfied in all the other neighbor caches but stored in any cache in the subtree rooted at level l, $hit_N - hit_l$. Thus, every subtree rooted at level l receives an additional traffic equal to $O^l \beta_I \cdot (hit_N - hit_l)$. Therefore, the request rate between levels l and $l+1$ in distributed caching is given by

$$\beta_l^d = O^l \beta_I \cdot ((1 - hit_l) + (hit_N - hit_l))$$

for $0 \leq l < 2H$ and by $O^{2H} \beta_I \cdot (1 - hit_N)$ for $2H \leq l < 2H + z$.

TABLE I
PERCENTAGE OF DOCUMENTS HIT AT DIFFERENT CACHING LEVELS FOR SEVERAL UPDATE INTERVALS ($\alpha = 0.8$)

Δ	7 days	10 days	15 days
hit_I	40%	42%	44%
hit_R	59%	61%	63%
hit_N	78%	79%	81%

To calculate the transmission time, we model the routers and caches on the network path from the sending to receiving host as M/D/1 queues. The arrival rate at a given network level for hierarchical and distributed caching is given by β_l^h and β_l^d. The service rate is given by the link's capacity at every network level (i.e., C_I, C_R, C_N, and C). We assume that the most congested network link from level l to the clients is the link at level l. Delays on network levels lower than l are neglected. Let $\tilde{S}$ be the average document size of all N documents. The M/D/1 queuing theory [17] gives

$$E[T_t^h|l] = \frac{\tilde{S}}{C_l - \beta_l^h \cdot \tilde{S}} \cdot \left(1 - \frac{\beta_l^h \cdot \tilde{S}}{2C_l}\right)$$

for the delay at every network level in a caching hierarchy, and

$$E[T_t^d|l] = \frac{\tilde{S}}{C_l - \beta_l^d \cdot \tilde{S}} \cdot \left(1 - \frac{\beta_l^d \cdot \tilde{S}}{2C_l}\right)$$

for the delay at every network level in a distributed caching scheme.

IV. HIERARCHICAL VERSUS DISTRIBUTED CACHING: NUMERICAL COMPARISON

To quantitatively compare hierarchical caching against distributed caching, we pick some typical values for the different parameters. The following parameters will be fixed for the remainder of the paper, except when stated differently. The network tree is modeled with an outdegree $O = 4$ and a distance between caching levels $H = 3$, yielding $O^H = 64$ regional and $O^{2H} = 4096$ institutional caches. The distance from the top node of the national network to the origin server is set to $z = 10$.

We consider $N = 250$ million Web documents [6], which are distributed following a Zipf distribution with α between 0.64 and 0.8 [7]. We consider the total network traffic generated from the institutional caches $O^{2H}\beta_I$ to be equal to 10 000 document requests per second [8]. We fix the average document size of a Web document as $\tilde{S} = 10$ kB [8]. We vary the update time Δ between several days and one month. We consider 10% of documents to be noncacheable [30]. All these values correspond to the same point in time (year 1998), and must be scaled accordingly for other years.

Hit rates hit_I, hit_R, and hit_N can be calculated with the formulas derived in Section III-B. Table I summarizes hit rates at different caching levels for homogeneous client communities and different update intervals. We see that the hit rate achieved at an institutional cache is about 40%, at a regional cache about 60%, and at the national cache about 80%. These values are quite similar to those reported in many real caches [30], [26], where trace-driven analysis of real caches report hit rates of 35%–50% at an institutional cache, 50%–60% at a regional cache, and 60%–70% at a national cache. The value that differs the most between the trace-driven results and our analytical results is the hit rate at the national cache; the analytical results show a 10% higher hit rate than in a real national cache. However, if we pick the appropriate parameter α, i.e., $\alpha = 0.64$, to model the heterogeneous client communities connected to the national cache, we obtain hit rates at the national cache about 70%, which are closer to the real ones.

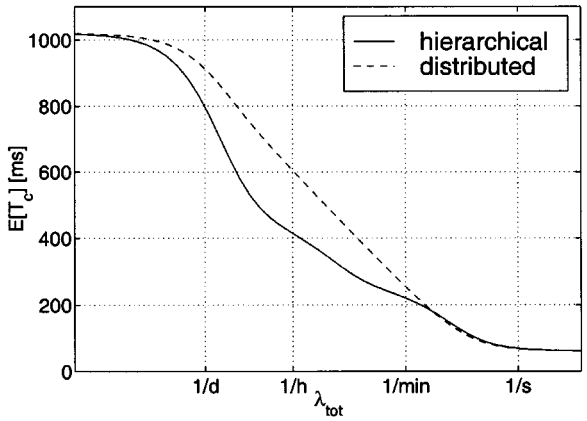


Fig. 3. Expected connection time $E[T_c]$ for hierarchical and distributed caching as a function of the document's popularity $\lambda_{\rm tot}$ ($\Delta = 24$ h, $d = 15$ ms).

A. Connection Time

The connection time is the first part of the perceived client latency when retrieving a document. We assume that it only and depends on the network distance to the document [20]. In Fig. 3, we show the connection time for distributed and hierarchical caching for a generic document with a documents' popularity $\lambda_{\rm tot}$.[1] We consider an update period $\Delta = 24$ h; larger update periods would result in more requests in a period Δ, which would displace the curves for hierarchical and distributed caching toward the left of the x axis. However, the relative performance of distributed and hierarchical caching would still be equivalent.

First, we observe that for an unpopular document (small $\lambda_{\rm tot}$), both hierarchical and distributed caching, experience high connection times because the request always travels to the origin server. As the number of requests for a document increases, the average connection time decreases since there is a higher probability to hit a document at caches closer than the origin server. For all the documents which $\lambda_{\rm tot}$ ranges between 1 request/day and 1 request/min, a hierarchical caching scheme provides connection times that are about 1.5 times smaller than a distributed caching scheme. Thus, placing document copies at the regional and the national caches in the middle of the network reduces the expected network distance to hit a document compared to the case where there is only support from institutional caches at the edge of the network. When a document is popular, the connection times of hierarchical and distributed caching are very similar, since there is a very high probability that the document is hit in an institutional cache.

[1]To make the abstraction of considering a generic document, we eliminate the sub-index i and use $\lambda_{\rm tot}$ instead of $\lambda_{{\rm tot},\,i}$.

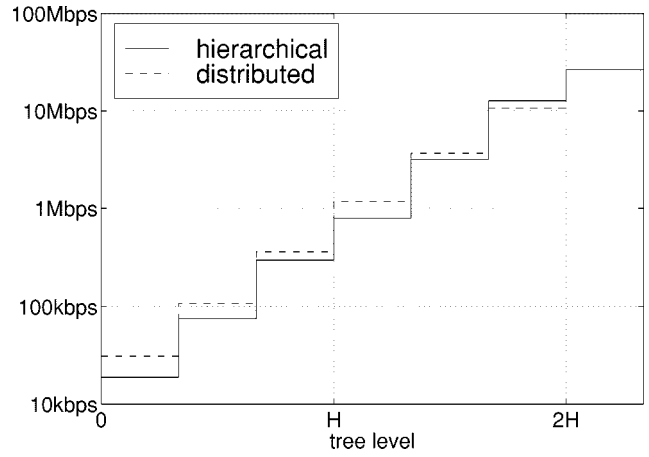


Fig. 4. Network traffic generated by distributed and hierarchical caching at every tree level ($\Delta = 24$ h).

B. Transmission Time

The second part of the overall latency is the time it takes to transmit the Web document itself. To calculate transmission time, we first show the distribution of the traffic generated by distributed caching β_l^d and hierarchical caching β_l^h at every network level (Fig. 4). We observe that distributed caching practically doubles the used bandwidth on the lower levels of the network tree, and uses more bandwidth in much of the national network. However, traffic on the most congested links, around the root nodes of the national network, is reduced to half. Distributed caching uses the low level links that interconnect institutional caches, thus generating more traffic in the less-congested low network levels.

Once we have presented the traffic generated at every network, level we calculate the transmission time for two different scenarios: 1) the national network is not congested and 2) the national network is highly congested. We set the institutional network capacity to $C_I = 100$ Mb/s. We consider the same network link capacities at the regional and national network, $C_N = C_R$. We do not fix the regional or national network link capacities to certain values, but consider only the degree of congestion $\rho = (\beta_{2H}^h \tilde{S})/C_N$ under which these links operate (i.e., we vary the utilization of the top national network links in the hierarchical caching scheme ρ). The international path is always very congested and has a utilization of $(\beta_I O^{2H}(1 - hit_N)\tilde{S})/C = 0.95$.

In Fig. 5(a), we show the transmission time for the case where the national network is not congested ($\rho = 0.3$). The only bottleneck on the path from the client to the origin server is the international path. We observe that the performance of hierarchical and distributed caching is similar because there are no highly congested links in the regional or national networks. In Fig. 5(b), we compare the performance of hierarchical and distributed caching in a more realistic scenario where the national network is congested, $\rho = 0.8$ (higher values of ρ would result in a worse performance of hierarchical caching). We observe that both hierarchical and distributed caching have higher transmission times than in the case when the national network is not congested [Fig. 5(a)]. However, the increase in the transmission time is much higher for hierarchical caching than for distributed

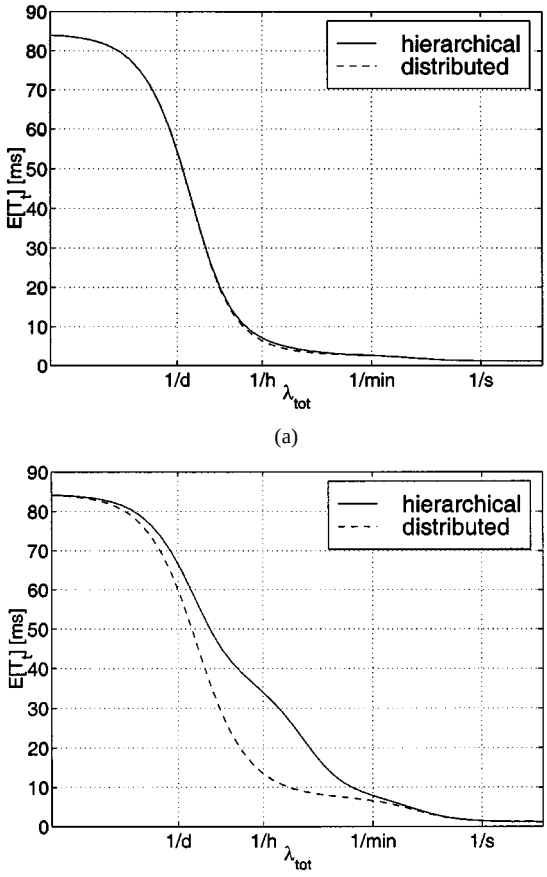


Fig. 5. Expected transmission time $E[T_t]$, for hierarchical and distributed caching. $\Delta = 24$ hours. (a) Not-congested national network: $\rho = 0.3$. (b) Congested national network: $\rho = 0.8$.

caching (even greater differences are obtained for higher degrees of congestion ρ). Distributed caching gives shorter transmission times than hierarchical caching (about a factor of 2.5 for average popular documents) because many requests are satisfied at less congested lower network levels. Similar results are also obtained in the case where the access link of the national cache is very congested. In this situation, the transmission time in distributed caching remains unchanged, while the transmission time in hierarchical caching increases considerably [29].

C. Total Latency

The total latency is the sum of the connection time and the transmission time. For large documents, the transmission time is more relevant than the connection time. For small documents, the transmission time is very small and the connection time tends to dominate. Next, we present the total latency for an average popular document with different document sizes S in both hierarchical and distributed caching. We will study the total latency in the scenario where the top nodes of the national network are highly congested. Fig. 6 indicates that hierarchical caching gives lower latencies for documents smaller than 200 kB because hierarchical caching has lower connection times than distributed caching. However, distributed caching gives lower latencies for higher documents because distributed caching has lower transmission times than hierarchical caching.

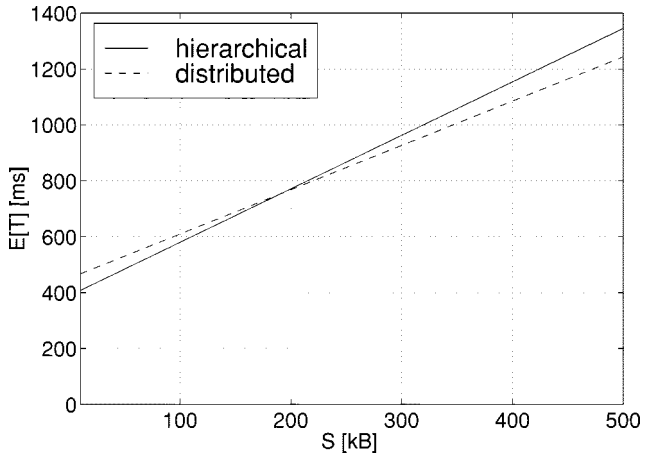


Fig. 6. Average total latency as a function of the document size S. National network is congested: $\rho = 0.8$.

The size threshold depends on the degree of congestion in the national network. The higher the congestion, the lower is the size threshold from which distributed caching has lower latencies than hierarchical caching.

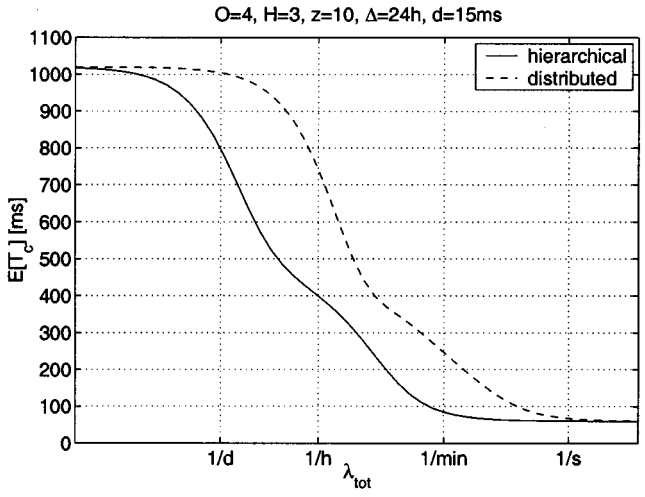
The results show that there is no simple caching scheme that is best for all scenarios. In Fig. 5, we show how a cache cooperating scheme that dynamically select the number of cooperating caches on a per document basis optimizes the total latency experienced by a client.

D. Heterogeneous Client Communities

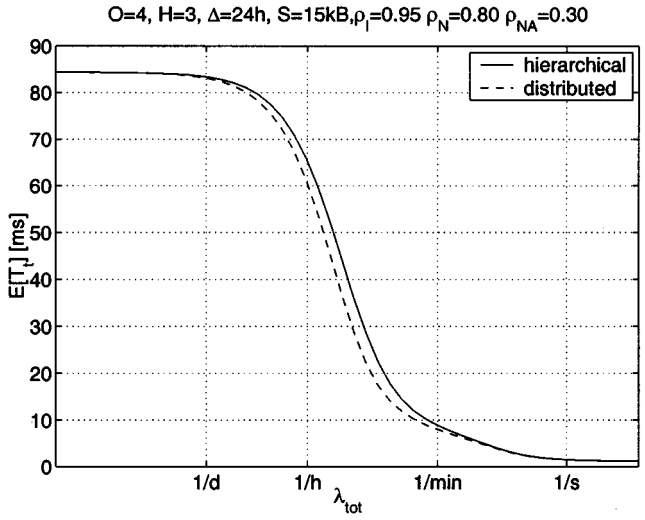
We study the latency performance of hierarchical and distributed caching in the case of heterogeneous client communities. To model heterogeneous client communities we consider the case where there are only clients in the two extreme institutional caches of the network that are interested in a certain document, and no other clients share the same interest. We consider this extreme case to obtain the largest differences between distributed and hierarchical caching under heterogeneous client communities. To model such a scenario, we modify the latency analysis and recalculate the distribution of the number of links to hit a document $P(L_i = l)$ (e.g., the probability to hit a document at the regional cache or lower is the same than the probability to hit a document at the institutional cache, since there are no institutional caches under the same regional cache that share the same request pattern).

In Fig. 7(a), we show the connection time for hierarchical and distributed caching under heterogeneous client communities. We observe that the connection time for an average popular document under distributed caching is about two times higher than the connection time of hierarchical caching. Comparing Fig. 7(a) with Fig. 3, we see that having heterogeneous client communities clearly increases the connection time for distributed caching since cache cooperation may happen among very distant caches. However, the performance of hierarchical caching almost remains unchanged since requests can always be satisfied from intermediate caches in the middle of the network that are close to the clients.

In Fig. 7(b), we show the transmission time for hierarchical and distributed caching. We have considered the case where the



(a)



(b)

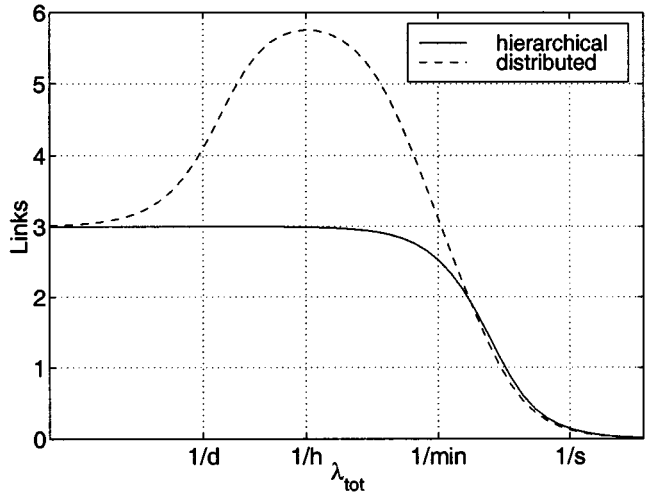
Fig. 7. Latency performance of hierarchical and distributed caching under heterogeneous client communities (only the two extreme institutional caches in the network share the same request pattern). $\Delta = 24$ hours, $d = 15$ ms, $\rho = 0.8$. (a) Expected connection time. (b) Expected transmission time.

national network is congested. Since requests for both hierarchical and distributed caching need to travel very high up in the distribution tree, the transmission time is similar for both. Comparing Figs. 7(b) and 5(b), we see that for heterogeneous client communities the transmission time of distributed caching increases significantly since requests need to travel through higher network levels, which are highly congested.

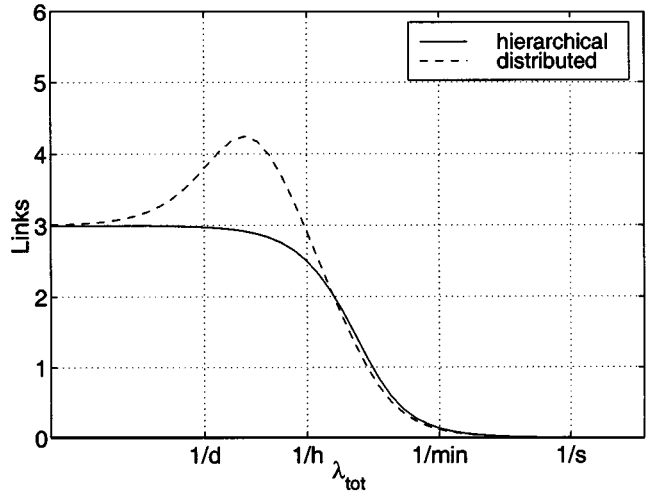
E. Bandwidth Usage

Next, we consider the ***bandwidth usage*** in the regional and national network for both caching architectures. Note that the bandwidth used on the international path is the same for both architectures, since all requests not satisfied at any cache level travel through the international path.

We calculate the bandwidth usage as the expected number of links traversed to distribute one packet to the clients. Fig. 8(a) shows the bandwidth usage of hierarchical and distributed caching in the regional network. We see that distributed caching uses more bandwidth than hierarchical caching in the regional



(a)



(b)

Fig. 8. Bandwidth usage on the regional and national network, as a function of the document request rate ($\Delta = 24$ h). (a) Regional network. (b) National network.

network since many documents are retrieved from distant institutional caches, traversing many links. However, recall that the network traffic is better distributed, with most of the bandwidth being used on lower less-congested levels of the network tree Section IV-B. On the other hand, hierarchical caching uses fewer links in the regional network since documents are hit in the regional cache.

A similar behavior occurs in the national network [Fig. 8(b)]; distributed caching uses more bandwidth than hierarchical caching. However, the bandwidth usage of distributed caching is lower in the national network than in the regional network, since many institutional caches can be reached only traversing few national network links.

The fact that hierarchical caching uses less bandwidth than distributed caching in the national and the regional network shows that a well-formed caching hierarchy implements an application-level multicast distribution [25], which is much more efficient in terms of bandwidth than using the only support of institutional caches at the edge of the network. Therefore, approaches like yoid [13] or napster [2] that only use the support from edge caches/hosts to provide a content distribution architecture have much worse performance in terms of bandwidth usage than other approaches that use cooperating hosts/caches inside the network, as does hierarchical caching.

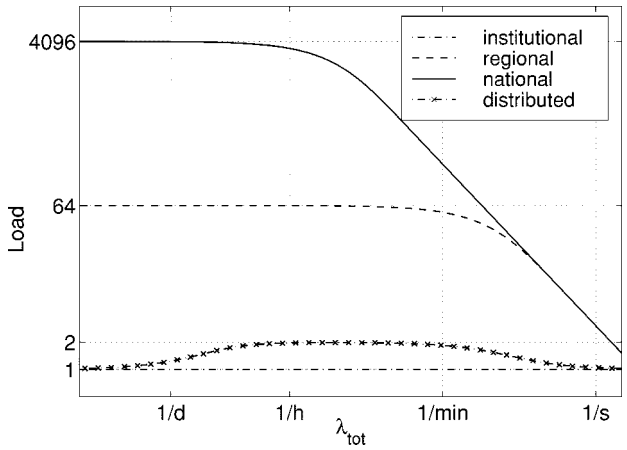


Fig. 9. Document request rate at the different caches with respect to the institutional request rate λ_I ($\Delta = 24$ h).

F. Cache Load

In order to estimate the load on the caches, we calculate the filtered request rate at every cache level for hierarchical and distributed caching. For a caching hierarchy, the filtered request rates are given by

$$\lambda^h_{I,i} = \lambda_{I,i}$$
$$\lambda^h_{R,i} = (1 - P(L_i \le 0)) \cdot O^H \lambda_{I,i}$$
$$\lambda^h_{N,i} = (1 - P(L_i \le H)) \cdot O^{2H} \lambda_{I,i}.$$

In distributed caching, the request rate at the institutional cache $\lambda^d_{I,i}$ is given by the local request rate $\lambda_{I,i}$ at the institutional cache, plus the external request rate arriving from the other cooperating institutional caches. The external request rate from the other cooperating institutional caches can be calculated using the percentage of requests that are not hit locally in an institutional cache and that are hit in any other institutional cache $(P(L_i \le 2H) - P(L_i \le 0))$. Recall that the percentage of local misses that are not satisfied among O^{2H} institutional caches is uniformly shared among the O^{2H} institutional caches. Therefore, the load at an institutional cache in distributed caching is given by

$$\lambda^d_{I,i} = \lambda_{I,i} + (P(L_i \le 2H) - P(L_i \le 0)) \cdot \lambda_{I,i}.$$

In Fig. 9, we calculate the load in the caches for both caching architectures as λ^h_l/λ_I and λ^d_l/λ_I, respectively, for different document popularities. With distributed caching, the request rate at an institutional cache doubles for a large range of document popularities. With hierarchical caching, the request rate at the regional and national caches increases by a factor of 64 or 4096, respectively, for documents that are not very popular. For very popular documents, the request rate at the national and regional caches is reduced since documents are hit at lower cache levels decreasing the load at higher cache levels.

Considering all N documents with different popularities, we can estimate the total incoming request rates at every cache

TABLE II
TOTAL REQUEST RATE AT EVERY HIERARCHY LEVELS FOR DIFFERENT NETWORK TOPOLOGIES, $\Delta = 15$ DAYS

	institutional	regional	national
O=4, H=3	0.2 req/s	6.4 req/s	122 req/s
O=3, H=3	0.2 req/s	2.7 req/s	25 req/s

level. In Table II, we show the load at every cache level of a caching hierarchy for different network topologies. In the first row of Table II, we consider the case where there are 64 children caches connected to one parent cache. We see that the national cache needs to handle request rates that are about 20 times higher than the request rates at a regional cache, and about 600 times higher than the request rates at an institutional cache. We have also considered the situation where the number of children caches connected to a parent cache is smaller, i.e., 27 children caches ($O = 3$, $H = 3$). In this scenario, we see that the load that the regional or national caches need to support is reduced. However, it is still between 13–125 times higher than the load at an institutional cache.

G. Disk Space

In this section, we analyze the disk requirements of hierarchical and distributed caching. In order to compare the disk space required in both caching schemes, we calculate the expected disk space to store one document in hierarchical caching D^h and in distributed caching D^d. We calculate the disk space required as the average Web document size S times the average number of copies present in the caching infrastructure. The average number of copies in the caching infrastructure can be calculated using the probability that a new document copy is created at every cache level. Thus, the average disk space D^h to store one document in hierarchical caching is

$$\begin{aligned} E[D^h] = &\tilde{S} \cdot O^{2H} \cdot (1 - e^{-\lambda^h_{I,i}\Delta}) \\ &+ \tilde{S} \cdot O^{H} (1 - e^{-\lambda^h_{R,i}\Delta}) \\ &+ \tilde{S} \cdot (1 - e^{-\lambda^h_{N,i}\Delta}) \end{aligned}$$

and, in distributed caching, it is given by

$$E[D^d] = \tilde{S} \cdot O^{2H} \cdot (1 - e^{-\lambda^d_{I,i}\Delta}).$$

Fig. 10 shows the ratio of the disk space used by hierarchical to distributed caching for different document popularities. We see that hierarchical caching always uses more disk space than distributed caching. For unpopular documents, hierarchical caching uses three times more disk space than disatributed caching, since document copies are present at the three levels of the caching hierarchy instead of being only at the institutional caches. For popular documents, the difference between hierarchical and distributed caching are reduced and hierarchical caching uses about 1.6% more disk space than distributed caching. The reason for this is that the extra number copies that hierarchical caching keeps with respect to distributed caching is given by the number intermediate cache ($O^H + 1$), which is a small number compared to the total number of caches (O^{2H}).

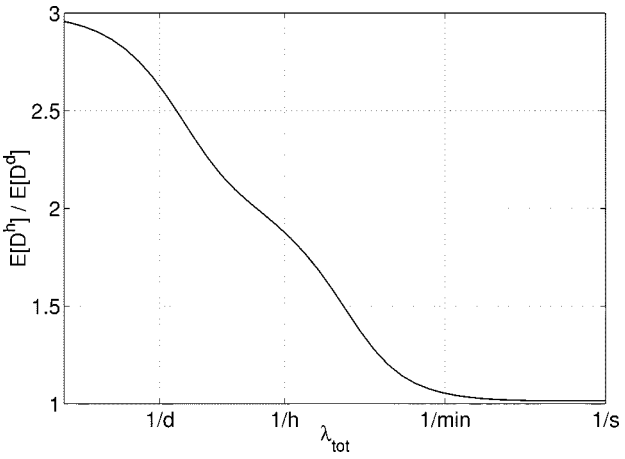


Fig. 10. Hierarchical-to-distributed disk-space ratio for one document, as a function of the document request rate.

TABLE III
INFINITE CACHE SIZE AT DIFFERENT LEVELS OF A CACHING HIERARCHY ($\tilde{S} = 10$ kB)

β^h_l, α	5 days	10 days	15 days
1 req/s α=0.8	3.6 GB	7 GB	10 GB
20 req/s α=0.64	111 GB	158 GB	369 GB
100 req/s α=0.64	316 GB	558 GB	759 GB

Given the distribution of document requests at level l for hierarchical caching $\lambda^h_{l,i}$, and the average Web document size $\tilde{S}$, we can derive the *infinite cache size* for each level of a caching hierarchy $D^h_{l,\infty}$, and for distributed caching D^d_∞, i.e., the disk space needed to keep a copy of every requested document while the document is up-to-date. For distributed caching, the infinite disk space is the same as the infinite disk space needed at the institutional caches of a caching hierarchy $D^d_\infty = D^h_{I,\infty}$. The infinite cache size for a cache on level l is given by

$$D^h_{l,\infty} = \tilde{S} \cdot \sum_{i=1}^{N} (1 - e^{-\lambda^h_{l,i}\cdot\Delta}).$$

To consider multiple scenarios with heterogeneous client communities with different Zipf parameters α and generic request rates β_l, we define the request rate $\lambda^h_{l,i}$ for document i at level l as $\lambda^h_{l,i} = \beta^h_l(\sigma/i^\alpha)$. Table III presents the infinite cache space for different request rates β^h_l, different Zipf parameters α, and different update periods Δ. We took values of α and β^h_l that correspond to those of a real institutional, regional, and national cache of a caching hierarchy [18], [7], [26], [3]. As the period Δ increases, the disk requirements also increase since the probability that a document is requested in a period Δ increases. At the limit, for an infinite update period Δ and a large enough client population, the cache would need a disk space equal to $N \cdot S$ kB to store all requested N documents.

From Table III, we first see that an institutional cache with a request rate equal to 1 request/s and skew factor $\alpha = 0.8$ requires a disk space between 3.6–10 GB. For a regional cache with a higher request rate (20 requests/s) and smaller skew factor (0.64), the disk space increases by a factor of 22–37. For a national cache with a request rate equal to 100 requests/s, the disk

space required to store all documents increases to several hundreds of gigabytes. These analytical values that have been obtained with our model are very close to those ones reported in different trace-driven simulations with similar parameters β_l^h and α [7], [15], [11]. For instance, in [11], an infinite disk space of 20 and 100 GB is given for request rates of 3 and 13 requests/s, respectively. In [33] it is suggested a disk space capable of storing at least three days of data. Assuming that we want to store a week of data, this rule would give us [for the three values of β_l^h used in Table III and the same skew factor (0.8)], and a disk space of 5.7, 115, and 570 GB. Note that for smaller skew factors α, the disk requirements will be higher.

As we have seen in this section, the performance requirements of high level caches in terms of disk space and cache load are very high, which makes it difficult to deploy a centralized top-level cache. As a result, many top-level caches use several machines to distribute load and disk requirements. These machines cooperate in a distributed fashion [3] using some of the existing cache sharing schemes [31], [12], [27].

Distributed caching can decrease the retrieval latency of large documents and reduce the bandwidth usage at high network levels. However, a large-scale deployment of distributed caching encounters several problems. When a document is fetched from a neighbor institutional cache, the experienced latency depends not only on the link bandwidth of the requesting institutional cache, but also on the link bandwidth of the neighbor cache contacted. So in a distributed caching scheme, investing into higher capacity links will not result in any benefit when documents are hit in neighbor caches with smaller connection capacities. In order to increase the local hit rates, local ISPs could increase the disk space of their cache, and thus store more documents. However, in distributed caching, the more documents a given institutional cache stores, more requests it receives from neighbor institutional caches. Thus, investing in larger disks to save bandwidth and reduce latency can eventually result in *more* incoming traffic and *longer* queuing delays at the local cache. Nevertheless, distributed caching can be used in a smaller scale where caches are interconnected at short distances with plentiful bandwidth, e.g., among caches in a metropolitan area or in a campus where cache cooperation is easier and there are no administrative issues.

V. A Hybrid Caching Scheme

In this section, we consider a hybrid caching scheme where a certain number of caches k cooperate at every network level of a caching hierarchy. With hybrid caching, when a document cannot be found in a cache, the cache checks if the document resides in any of the cooperating caches. If multiple caches happen to have a document copy, the neighbor cache that provides the lowest latency is selected. If the document does not reside among the considered caches at a given level, the request is then forwarded to the immediate parent cache or to the origin server.

The number of caches that should be considered at every cache level to locate a document copy should be limited to avoid fetching a document from a distant cache that could have been fetched faster from a parent cache or the origin server. We now investigate the optimal number of caches that should cooperate at every cache level to minimize the document retrieval latency. Due to space limitations, we omit the detailed analysis for hybrid caching, which can be found in [29]. For hybrid caching, the probability that a document is found among k cooperating institutional caches can be calculated using (3) and replacing O^l institutional caches by k cooperating caches. At the regional level, to calculate the probability that a document is found among k cooperating regional caches, O^l should be replaced by $O^H \cdot k$.

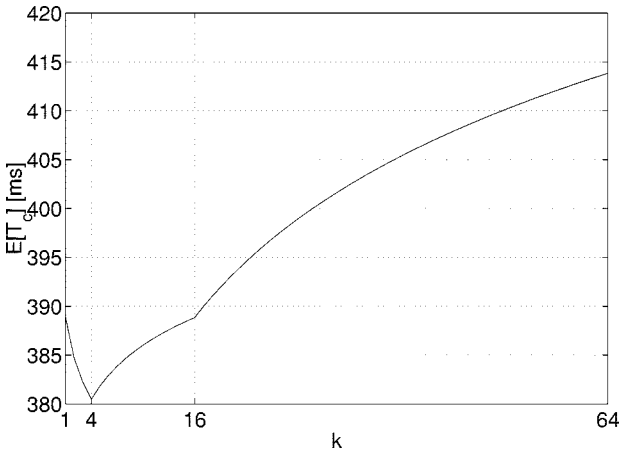


Fig. 11. Connection time as a function of the number of cooperating caches at every cache level in a hybrid scheme.

A. Connection Time

Next, we present the connection time in a hybrid scheme as a function of the number of cooperating caches k at every level. The number of cooperating caches at every cache level can range from one (no cooperation) to $O^H = 64$ (all neighbor caches with the same parent). Fig. 11 shows the average connection time for all N Web documents, depending on the number k of cooperating caches. Based on the network model presented in Section II, we observe that when the number of cooperating caches is just one or two, the connection time is about 385–390 ms, since most requests are satisfied from the parent cache. When the number of cooperating caches is four, the connection time reaches a minimum. If more distant caches are considered, the connection time increases again since documents are fetched from caches that have higher response times than the parent cache. There is, therefore, an optimal number of caches that should cooperate at every cache level to minimize the connection time. The optimal number of cooperating caches k_c that minimizes the *connection time* is the number of caches that are at closer network distances and therefore have lower response times than the parent cache, i.e., $k_c = O^{\lfloor H/2 \rfloor}$ in our network model.

Fig. 12 presents the connection time for distributed caching, hierarchical caching, and a hybrid scheme with the optimal number of cooperating caches k_c. We observe that a hybrid scheme with k_c cooperating caches has much lower connection times than distributed caching and even lower connection times than hierarchical caching (about 1.15 times less) for documents that have between 1 request/h and 1 request/s.

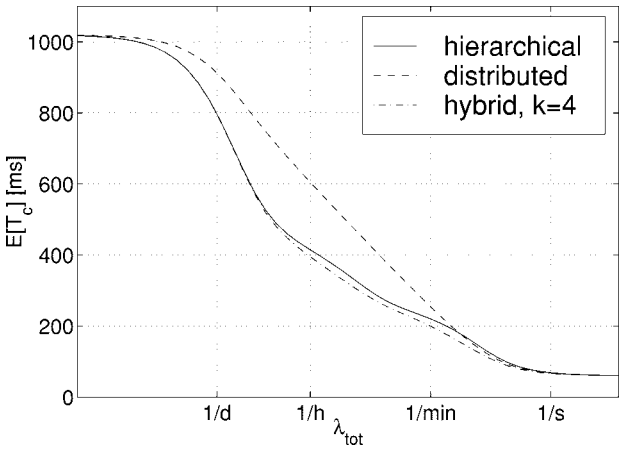


Fig. 12. Connection time in a hybrid caching scheme with the optimal number of cooperating caches k_c.

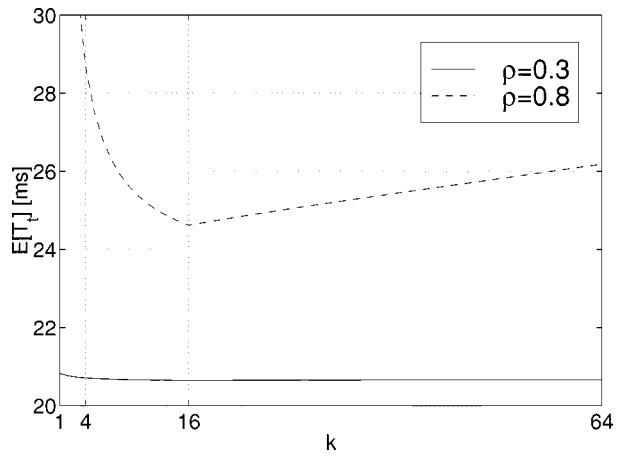


Fig. 13. Average transmission time as a function of the number of cooperating caches in a hybrid scheme. National network is not congested: $\rho = 0.3$. National network is congested:$\rho = 0.8$.

B. Transmission Time

Next, we analyze the transmission time in a hybrid scheme and calculate the optimal number k_t of cooperating caches at every network level that minimizes the transmission time.

In Fig. 13, we plot the transmission time in a hybrid scheme for all N Web documents, depending on the number of cooperating caches at every cache level. We consider the case where the top links of the national network are not congested ($\rho = 0.3$); that is, the only bottleneck in the path from the origin server to the client is the international path, and the case where the top links of the national network are congested ($\rho = 0.8$). Similar results are also obtained for the case that the access link of the national cache is congested [29].

We observe that, for a noncongested national network, varying the number of cooperating caches k at every cache level hardly influences the transmission time (from 20.7 to 20.5 ms). However, when the national network is congested, the transmission time strongly depends on the number of cooperating caches at every cache level. If the number of cooperating caches is very small, there is a low probability that the document can be retrieved from close neighbor caches. Since the parent cache contains all documents requested by its children caches, there is a higher probability that the document is retrieved from the parent cache traversing the highly congested top-level links. As the number of cooperating caches increases, the probability to hit the document at close neighbor caches connected by fast links increases, and thus, the transmission times are lower. If the number of cooperating caches increases over a certain threshold ($k_t = 16$ in our model), the transmission time starts increasing, since documents can be hit in distant neighbor caches, which can be interconnected through highly congested top-level links. The optimal number of cooperating caches k_t that minimizes the experienced *transmission time* depends on the number of cooperating caches reachable with out using the congested links. In the case where the top-level links of the national network are congested, the optimal number of cooperating caches at every cache level is $k_t = 16$. This value corresponds to the number of regional caches that can cooperate without traversing the national top-level links, which is $k_t = O^{H-1}$ in our network model.

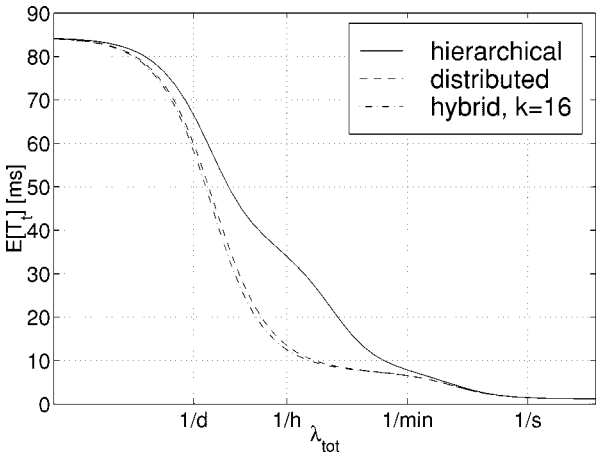


Fig. 14. Transmission time for a hybrid caching scheme with the optimal number of cooperating caches k_t as a function of the document's popularity. National network is congested: $\rho = 0.8$.

In Fig. 14, we present the transmission time $E[T_t]$ for a hybrid scheme with the optimal number of cooperating caches k_t, distributed caching, and hierarchical caching. We observe that a hybrid scheme with k_t cooperating caches has a transmission time which is about 2.9 times lower than for hierarchical caching. We also observe that a hybrid scheme has a transmission times that is about 1.12 times lower than for distributed caching since it reduces even more the traffic around the high network levels [29].

Thus, by dynamically choosing the number of cooperating caches, a hybrid scheme can have as low connection times as hierarchical caching, and as low transmission times as distributed caching.

C. Total Latency

Depending on the document size there is an optimal number of cooperating caches at each cache level that minimizes the total latency. For small documents the optimal number of cooperating caches is close to k_c, since choosing k_c cooperating caches minimizes the connection time. For large documents the optimal number of cooperating caches is close to k_t, since choosing k_t cooperating caches minimizes the transmission

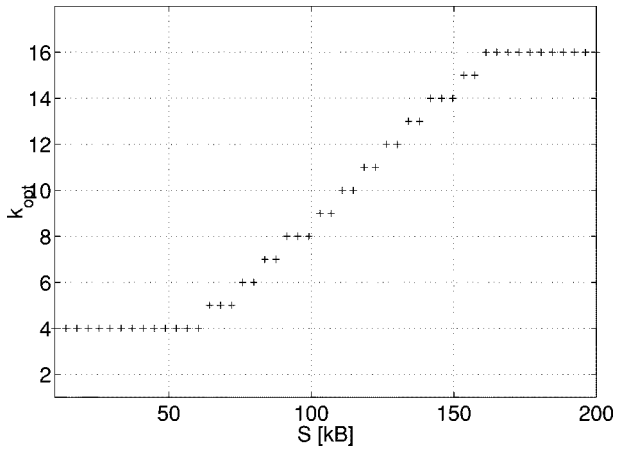


Fig. 15. Optimal number of cooperating caches $k_{\rm opt}$, as a function of the document size S. National network is congested: $\rho = 0.8$.

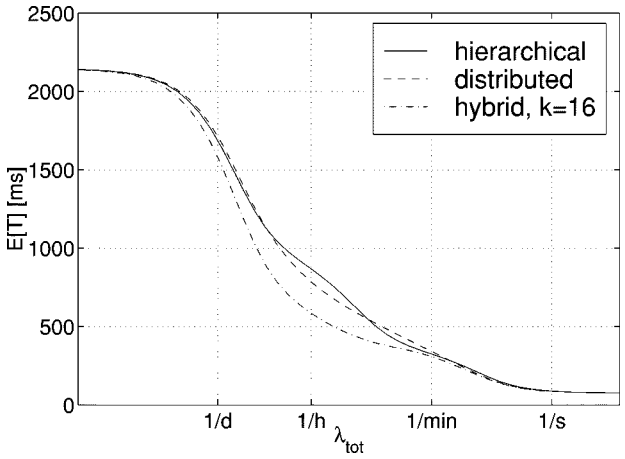


Fig. 16. Total latency to retrieve a large document in a hybrid caching scheme with the optimal number of cooperating caches $k_{\rm opt} = k_t = 16$. National network is congested: $\rho = 0.8$ ($S = 200$ kB).

time. For any document size, the optimal number of cooperating caches k_{opt} that minimizes the total retrieval latency is such that $k_c \leq k_{\rm opt} \leq k_t$.

In Fig. 15, we plot the optimal number of caches $k_{\rm opt}$ that should cooperate at every network level to minimize the total retrieval latency depending on the document size. We choose the case where the top-level links of the national network are highly congested, thus the optimal number of caches that minimizes the transmission time is $k_t = 16$. In Fig. 15, we observe that $k_{\rm opt}$ ranges between $k_c = 4$ and $k_t = 16$. For documents smaller than several kilobytes, only $k_c = 4$ caches should cooperate at every cache level. For documents larger than several cents of kilobytes, $k_t = 16$ caches should cooperate at every cache level.

In Fig. 16, we show the total retrieval latency for a large document ($S = 200$ kB) and the optimal number of cooperating caches $k_{\rm opt} = k_t = 16$. We see that a hybrid scheme with the optimal number of cooperating caches at every cache level has lower overall retrieval latencies than distributed and hierarchical caching for a large range of document's popularities (about a factor of 1.5).

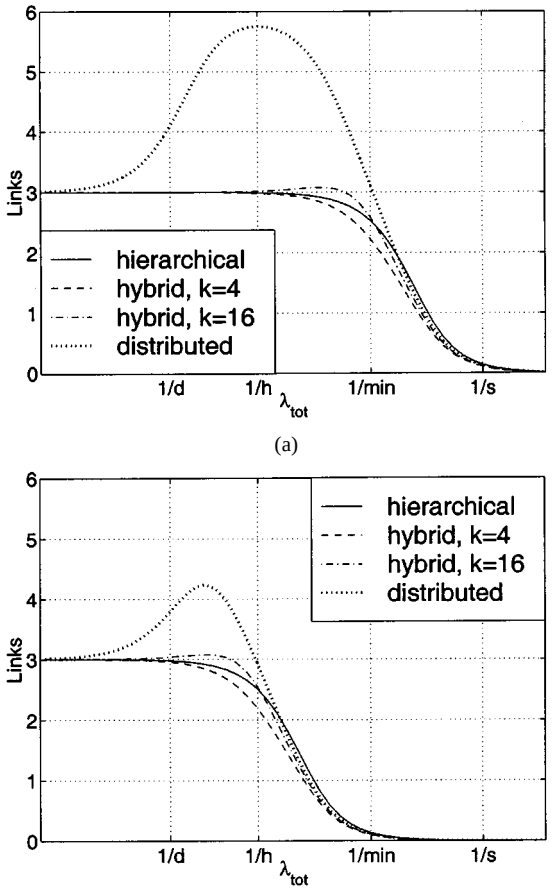


Fig. 17. Bandwidth usage on the national and regional network for the hierarchical, distributed and hybrid caching schemes. (a) National network. (b) Regional network.

D. Bandwidth

Having caches cooperate at every cache level not only influences the latency experienced by the clients, but also modifies the bandwidth patterns inside the network. In Fig. 17(a) and (b), we compare the bandwidth usage of a hybrid scheme for different degrees of cooperation with the bandwidth usage of hierarchical and distributed caching in the regional and national networks.

We see that the bandwidth usage of hybrid caching is smaller than the bandwidth usage of distributed caching for a large range of document popularities in the national, and the regional network. The reason for this result is that having intermediate caches in the network can greatly reduce the bandwidth usage, emulating a multicast distribution at the application-layer. The performance of hybrid caching with respect to hierarchical caching depends on the number of cooperating caches. In particular, when there are $k = k_c = 4$ caches that cooperate at every level, the bandwidth usage of a hybrid architecture is even lower than the bandwidth usage of hierarchical caching (about a factor of 1.2 for average popular documents). Note that having $k_c = 4$ cooperating caches at a caching level minimizes the number of network hops traversed by a request. When $k = k_t = 16$ caches cooperate to minimize the transmission time in the case of congested links, the bandwidth usage of hybrid caching increases slightly (about a factor of 1.3 for

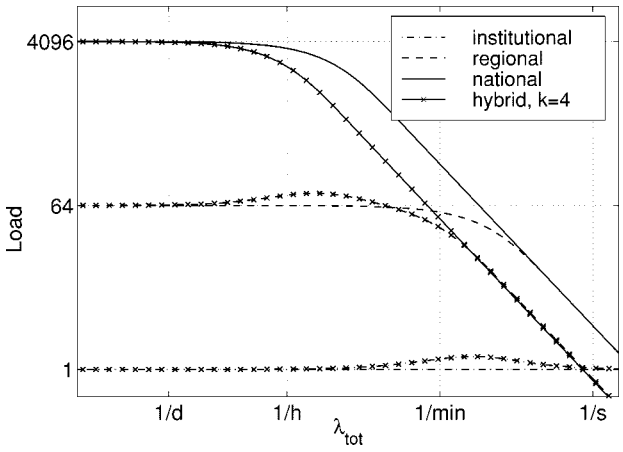


Fig. 18. Document request rate at the different caches with respect to the institutional request rate λ_I for hierarchical caching and for hybrid caching with $k = k_c = 4$ cooperating caches (crossed lines). $\Delta = 24$ h.

average popular documents) compared to a pure hierarchical configuration since distant caches may be contacted to avoid traversing the congested links.

E. Cache Load

A hybrid architecture redistributes the load at every cache level. Institutional and regional caches receive more requests from neighboring cooperating caches and the national cache receives fewer requests since more requests are satisfied among cooperating caches.

Fig. 18 shows the load at every cache level for hierarchical and hybrid caching with $k = 4$ (for the load generated by distributed caching refer to Fig. 9). We see that a hybrid scheme considerably reduces the request rate for popular documents compared to hierarchical caching since many documents are hit in neighbor caches. For average popular documents, the load at the regional and institutional caches increases slightly, since every cooperating cache serves a part of all the requests hit among the k cooperating caches. For a higher number of cooperating caches k, the results would not vary significantly, since the load is shared between a higher number of caches.

VI. Conclusion

We have developed analytical models to analyze the performance of hierarchical and distributed caching in terms of latency, hit rate, disk space, bandwidth usage, and load in the caches. We have found that a hierarchical cache scheme using intermediate caches in the network:

- reduces the expected network distance to hit a document;
- decreases the bandwidth usage by effectively implementing a multicast distribution at the application level;
- reduces the administrative concerns compared to a distributed architecture that only uses caches at the edge of the network.

However, a hierarchical caching architecture needs powerful intermediate caches or intelligent load-balancing algorithms to avoid high peaks of load in the caches that will result in high client latency. On the other hand, distributed caching has very good performance in well-interconnected areas without requiring any intermediate cache levels. Nevertheless, the deployment of distributed caching on a large scale encounters several problems, such as large network distances, high bandwidth usage, and administrative issues.

We have also analyzed a hybrid scheme where caches cooperate at every network level of a caching hierarchy. We found that a hybrid caching scheme can combine the advantages of both hierarchical and distributed caching, reducing the connection time as well as the transmission time. However, the number of caches that should be considered to locate a document copy at each level depends on the number of hops from the client to the caches, the congestion of the network, the parent cache/server load, and the document's size [29].

Our results on hierarchical and distributed caching not only apply to the area of Web caching, but also to some new applications that have recently become very popular to distribute content only using end client's caches. Programs like napster [2], gnutella [1], or scour [4] provide a distributed caching architecture to distribute images, mpg3 files, and other kinds of information that is stored at individual client caches. These applications use a central repository where all clients specify the documents that they cache. Clients query this repository to find where the information resides, and obtain a list of the cooperating clients who have a copy of the document, the location of the cooperating clients, and its connectivity. Among all the cooperating clients with a document copy, a client manually selects the cooperating client that is geographically closer or the cooperating client with the highest bandwidth connectivity. Then, the client caches the information locally and makes it available to other cooperating clients. In this way, popular content is rapidly replicated from one client to another, thus increasing the probability to find a popular document in a nearby client. Proposals like yoid [13] also consider to distribute content using a multicast distribution with the only support of client's caches. The results presented in this paper show that placing some intermediate caches in the network can significantly improve the performance of those applications that distribute content using the only support of client's caches, reducing the bandwidth usage in the network and the response time to the clients.

Acknowledgment

The authors like to thank the anonymous reviewers for their detailed comments that helped considerably in improving the paper.

References

[1] Wego Systems, Inc.. (1999) Gnutella. [Online]. Available: http://gnutella.wego.com
[2] Napster, Inc.. (2001). [Online]. Available: http://www.napster.com
[3] National Lab of Applied Network Research (NLANR). [Online]. Available: http://ircache.nlanr.net/
[4] CenterSpan Communications Corp.. (2001) Scour. [Online]. Available: http://www.scour.com
[5] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "World Wide Web caching: The application-level view of the internet," *IEEE Commun. Mag.*, pp. 170–178, June 1997.
[6] K. Bharat and A. Broder, "A technique for measuring the relative size and overlap of public web search engines," in *Proc. 7th Int. WWW Conf.*, Brisbane, Australia, Apr. 1997, pp. 379–388.

[7] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "On the Implications of Zipf's law for web caching," in *Proc. IEEE INFOCOM'99*, New York, Mar. 1999.
[8] E. A. Brewer, P. Gauthier, and D. McEvoy, "The long-term viability of large-scale caching," in *Proc. 3rd Int. WWW Caching Workshop*, Manchester, U.K., June 1998.
[9] A. Chankhunthod *et al.*, "A hierarchical internet object cache," in *Proc. 1996 USENIX Technical Conf.*, San Diego, CA, Jan. 1996.
[10] K. Claffy and H.-W. Braun, "Web traffic characterization: An assessment of the impact of caching documents from NCSA's web server," in *Electronic Proc. 2nd World Wide Web Conf.'94: Mosaic and the Web*, 1994.
[11] B. M. Duska, D. Marwood, and J. Feeley, "The measured access characteristics of World-Wide-Web client proxy caches," in *Proc. USENIX Symp. Internet Technologies and Systems*, Dec. 1997.
[12] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," in *Proc. SIGCOMM'98*, Feb. 1998, pp. 254–265.
[13] P. Francis. (2000) Yoid: Extending the Internet multicast architecture. Unfereered report. [Online] Available http://www.aciri.org/yoid/docs/index.html
[14] S. Gadde, M. Rabinovich, and J. Chase, "Reduce, reuse, recycle: An approach to building large internet caches," in *Proc. 6th Workshop on Hot Topics in Operating Systems (HotOS–VI)*, May 1997.
[15] S. Gribble and E. Brewer, "System design issues for Internet middleware services: Deductions from a large client trace," in *Proc. USENIX Symp. Internet Technologies and Systems*, Dec. 1997.
[16] D. Karger, A. Sherman, A. Berkhemier, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web caching with consistent hashing," in *Proc. 8th Int. World Wide Web Conf.*, May 1999.
[17] L. Kleinrock, *Queuing Systems, Volume I: Theory*. New York: Wiley, 1975.
[18] A. Mahanti, C. Williamson, and D. Eager, "Traffic analysis of a web proxy caching hierarchy," *IEEE Network Mag.*, May–June 2000.
[19] M. Makpangou, G. Pierre, C. Khoury, and N. Dorta, "Replicated directory service for weakly consistent replicated caches," in *Proc. ICDCS'99 Conf.*, Austin, TX, May.
[20] P. McManus. (2001) A passive system for server selection within mirrored resource enviroments using as path length heuristics. [Online]. Available: http://proximate.appliedtheory.com
[21] J. Nonnenmacher and E. W. Biersack, "Performance modeling of reliable multicast transmission," *Proc. IEEE INFOCOM'97*, Apr. 1997.
[22] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Scaling of multicast trees: Comments on the Chuang–Sirbu scaling law," in *Proc. ACM SIGCOMM'99*, Harvard, MA, Sept. 1999, pp. 41–51.
[23] D. Povey and J. Harrison, "A distributed Internet cache," in *Proc. 20th Australian Computer Science Conf.*, Sydney, Australia, Feb. 1997.
[24] M. Rabinovich, J. Chase, and S. Gadde, "Not all hits are created equal: Cooperative proxy caching over a wide-area network," in *Proc. 3rd Int. WWW Caching Workshop*, Manchester, U.K., June 1998.
[25] P. Rodriguez, K. W. Ross, and E. W. Biersack, "Distributing frequently-changing documents in the Web: Multicasting or hierarchical caching," in *Proc. Computer Networks and ISDN Systems: Selected Papers 3rd Int. Caching Workshop*, 1998, pp. 2223–2245.
[26] A. Rousskov, "On performance of caching proxies," in *Proc. ACM SIGMETRICS*, Madison, WI, Sept. 1998.
[27] A. Rousskov and D. Wessels, "Cache digest," in *Proc. 3rd Int. WWW Caching Workshop*, June 1998, pp. 272–273.
[28] N. G. Smith, "The UK national Web cache—The state of the art," in *Proc. Computer Networks and ISDN Systems*, vol. 28, 1996, pp. 1407–1414.
[29] C. Spanner, "Evaluation of web caching strategies: Distributed vs. hierarchical caching," Masters thesis, Univ. Munich/Institut Eurøcom, Sophia Antipolis, France, Nov. 1998.
[30] R. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay, "Beyond hierarchies: Design considerations for disturbed caching on the Internet," in *Proc. ICDCS '99 Conf.*, Austin, TX, May 1999.
[31] V. Valloppillil and K. W. Ross. (1998) Cache array routing protocol v1.1. Internet draft. [Online]. Available: http://ds1.internic.net/internet-drafts/draft-vinod-carp-v1-03.txt
[32] P. Vixie and D. Wessels, "RFC 2756: Hyper text caching protocol," (HTCP/0.0), Jan. 2000.
[33] D. Wessels. (1996) Squid Internet object cache. [Online]. Available: http://www.nlanr.net/Squid
[34] D. Wessels and K. Claffy, "Application of Internet cache protocol (ICP), version 2," Internet Engineering Task Force, Internet Draft:draft-wessels-icp-v2-appl-00. Work in Progress., May 1997.
[35] G. K. Zipf, *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Reading, MA: Addison-Wesley, 1949.

Pablo Rodriguez received the Ph.D. degree from the Institut EURECOM, Sophia Antipolis, France, and the Masters degree in telecommunication engineering from the University of Navarra (UPNA), Pamplona, Spain. His thesis work was conducted at King's College, London, U.K.

While at King's College, he also worked on optical communications.

Christian Spanner received the M. S. degree in computer science from the Technische Universität München, Munich, Germany. His thesis work was conducted at the Institut EURECOM, Sophia Antipolis, France.

Ernst W. Biersack (M'88) received the M.S. and Ph.D. degrees in computer science from the Technische Universität München, Munich, Germany.

Since March 1992, he has been a Professor in Telecommunications at the Institut EURECOM, Sophia Antipolis, France.

Dr. Biersack is a member of the Association of Computing Machinery (ACM).