Limitations and Benefits of Cooperative Proxy Caching

Sandra G. Dykes [*] and Kay A. Robbins [†‡]

Abstract

Cooperating proxy caches are groups of HTTP proxy servers that organize to share cached objects. This paper develops analytical models for proxy cooperation based on speedup in user response time. Speedup expressions are derived for the cooperation upper bound, a proxy mesh, and a three-level proxy hierarchy. The equations allow comparisons of fundamental design choices by separating the delivery organization from the discovery mechanism. For the mesh and hierarchy models, the discovery mechanisms include ideal discovery, ICP query, and distributed metadata directories. Equations are evaluated using parameter estimates from experiments and cache trace analysis. Results indicate that proxy cooperation is marginally viable from the standpoint of average user response time, and that the miss penalty for the hierarchy renders it less viable than the mesh. Proxy cooperation can, however, reduce the variability in user response time and the number of long delays. A trace-driven simulation shows that caching constraints have little effect on cooperation performance due to request filtering by lower level caches.

Keywords **– Web proxy, Web caching, cooperative cache, cache hierarchy**

---

[*]S. Dykes was with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249 USA. She is now with Southwest Research Institute, San Antonio, TX 78250 (email: sdykes@swri.org)

[†]K. Robbins is with the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249 USA (email: krobbins@cs.utsa.edu).

[‡]This work was supported by National Science Foundation Grant CDA-9633299 and by a fellowship from the NASA/Texas Space Grant Consortium. An earlier version of this paper appeared in *Proceeding of IEEE INFOCOM 2001*.

1 Introduction

Caches are defined as fast, temporary stores for commonly used items. High speed memory units cache data from main memory; main memory units cache sections of disk files; and local disks cache documents from network file servers. Caching succeeds in these examples because each cache level is progressively faster and the data exhibit locality of reference. Client caching on the Web succeeds at the browser level (L1) and the local proxy level (L2) for the same reasons. Web objects are often re-requested by the same user or by other users on the same local network, and the local caches have an inherent speed advantage over remote Web servers because they do not require a remote network transfer.

At the next cache level (L3), proxies cooperate to share objects across the cooperation set. If a cache miss occurs at the local proxy, that proxy may retrieve the object from a another proxy cache instead of the origin server. While researchers agree that caches on the client browser and local intranet improve performance [1, 24], there is considerable debate over the benefits of proxy cooperation and the relative performance of different cooperation architectures [8, 17, 30, 37, 38]. The viability of proxy cooperation is unclear for two reasons. First, lower level caches filter duplicate requests from the L3 request stream, thereby reducing the potential L3 hit rate. Assuming a local proxy has sufficient disk space, it infrequently re-issues requests for cacheable objects. Consequently, there is little duplication of cacheable objects within individual proxy request streams and the L3 hit rate is essentially limited to the overlap between proxy streams. The second limitation on cooperation success is the lack of an inherent speed advantage. Unlike browers and local proxies, a remote proxy has no fundamental advantage over a remote Web server because both deliver the object over a wide-area network subject to congestion and unpredictable delays. In general, there is no guarantee that the cache site has a faster file system, higher bandwidth, or a better network route to the client. When cache and server sites have similar resources and communication costs,

proxy cooperation actually increases the average user delay due to overhead costs and cache miss penalties.

A group of cooperating proxies can realize a speed advantage if objects are available from multiple sites and the proxies employ a statistically effective method to predict the faster sites [9, 21]. However, average response time improves only if the hit rate and speed advantage are large enough to offset the cooperation overhead. Overhead for proxy cooperation stems from three separate cache functions: *discovery*, *dissemination* and *delivery*. Discovery refers to how a proxy locates cached objects. Dissemination is the process of selecting which objects are to be cached, and transferring those objects from the origin servers to the caches. Delivery defines how objects are delivered from the caching system to the client at the time of the request. Dissemination overhead is modeled separately in our analysis because we consider only pull-caching designs that disseminate objects through the delivery process. Cooperation architectures are classified according to their delivery methods. In direct delivery architectures, the L2 proxy retrieves the object directly from the origin server, whereas indirect delivery implies the object is channeled through one or more other proxies on the way. A delivery method does not necessarily mandate a particular discovery or dissemination mechanism. For example, cache hierarchy architectures using Squid software [33] can implement discovery with the query-based Internet Cache Protocol (ICP) [34, 35] or directory-based Cache Digests [29].

The goal of this work is to assess the viability of cooperative proxy caching with respect to user response time by developing analytical models and evaluating them with experimentally determined parameters. In addition to numerical performance estimates, the analytical expressions provide insight into the interaction of L3 hit rate, average cache speed advantage, and cooperation overhead. The metric for the analysis is speedup in end-user response time. The analysis derives models and numerical speedup estimates for the upper bound on performance and for mesh and

hierarchy cooperation architectures with different discovery mechanisms. To examine the effect of caching constraints, we include a trace-driven simulation that predicts hit rate and response time speedup for the case in which all objects are cacheable.

The paper is organized as follows. Section 2 describes the mesh and hierarchy architectures, and defines symbols and assumptions used in the analytical models. Speedup equations are derived in Section 3, experiments used to determine parameter estimates are explained in Section 4, and numerical results are presented in Section 5. Section 6 defines a response time viability condition and applies it to the cooperation models. In Section 7, we examine benefits of proxy cooperation by considering measures other than average response time. The effect of caching constraints is examined in Section 8, Section 9 discusses cache configurations outside the scope of our models, and Section 10 describes related performance studies. Section 11 concludes with additional comments and viability considerations.

2 Definitions and assumptions

2.1 Hierarchy and mesh architectures

In a cache hierarchy, a separate L3 tier of remote caches acts as parents to the local L2 proxies (Figure 1). Each L2 cache has a single parent and zero or more siblings, where sibling caches are peer L2 proxies with the same parent. Although cache hierarchies can have levels beyond L3 [27, 37], the hierarchical model in this paper assumes a three-level design similar to the National Laboratory for Applied Network Research (NLANR) [23] cache hierarchy. When an L2 miss occurs at the local proxy cache, it requests the object from either its parent or from a sibling cache. A cooperation hit occurs if this L3 request succeeds and the object is delivered from a proxy in the cooperation set. If the L3 request fails, the parent cache retrieves the object from the origin server and returns it to the local proxy. Thus for L3 misses, the hierarchy uses an indirect delivery

mechanism with two remote network transfers. Section 3.6 discusses the implications of indirect delivery on the L3 miss penalty for the hierarchy.

By contrast, a mesh architecture employs direct delivery for both L3 hits and misses, using a single remote network transfer in either case (Figure 2). An L3 hit occurs when the local proxy retrieves the object from a remote peer cache. An L3 miss occurs if the object is not available at any peer cache *or* if the local proxy does not know the object is available. In the event of an L3 miss, the local proxy sends its request to the origin server and retrieves the object directly.

In most mesh designs, local proxies maintain metadata directories containing information about the objects at peer caches. Although object delivery is direct, the metadata can be propagated in different ways. Tewari *et al.* [30] propose a hierarchical push structure for metadata delivery, while Fan *et al.* [12] and Rousskov *et al.* [29] suggest compressing the metadata for exchange with other proxies. Cooperative caching can also use hashing schemes for object discovery [5, 15, 26]. Our models treat any cost associated with metadata propagation or cache selection as discovery overhead.

2.2 Symbol definitions

Table 1 defines the symbols used in the viability analysis. Our models assume that all users direct their requests through a local (L2) proxy server, with associated speedups derived for the end user rather than the local proxy.

H_3 is the maximum achievable hit rate for a reference stream at the L3 level. This hit rate is characteristic of the sharing potential in the reference stream and is independent of the caching mechanism. As we will show in the next section, the upper bound on speedup in user response time depends solely upon H_3. In contrast, h_3 is the measured hit rate of a particular cache system. The two hit rates are related through an efficiency factor, $\mathcal{E}$, which reflects the fraction of false misses. False misses occur when a requested object is cached within the cooperation system, but

Symbol	*Definition*
h_2	Hit rate at the local proxy cache (L2)
h_3	Hit rate at the cooperation cache level (L3)
H_3	Maximum L3 hit rate
$\mathcal{E}$	Efficiency of cache discovery
T_{local}	Average time to retrieve an object from the local proxy cache
T_C	Average time to retrieve an object from a remote proxy cache
T_S	Average time to retrieve an object from the origin Web server
$T_{o,hit}$	Average cooperation overhead for an L3 hit
$T_{o,miss}$	Average cooperation overhead for an L3 miss
$T_{d,hit}$	Average discovery overhead for an L3 hit
$T_{d,miss}$	Average discovery overhead for an L3 miss

Table 1: *Symbols used in the response time and speedup expressions.*

discovery fails. $\mathcal{E}$ plays an important role in viability and is discussed in detail in Section 3. T_{local} is the average time for a user to retrieve an object from the local L2 proxy, and includes proxy processing time, proxy file system delay, and local network delivery time. T_S is the average time to retrieve an object from a remote Web server, which consists of server processing time, server file system delay, and the remote and local network delivery times. T_C is the average time to retrieve an object from a remote (L3) cooperative cache and is defined similarly to T_S.

Cooperation overhead for L3 hits and L3 misses is denoted by $T_{o,hit}$ and $T_{o,miss}$, respectively. This overhead contains discovery costs and any extra time the cache system adds to object delivery. For some models, it will be convenient to separate out the discovery overhead, denoted by $T_{d,hit}$ and $T_{d,miss}$. Independent terms are necessary because discovery overhead for hits and misses may differ under some designs.

2.3 Assumptions

The models developed in the remainder of the paper rely on the following assumptions.

Assumption 1 *The average time for an end user to retrieve an object from its local L2 cache is significantly less than the average time required for the local L2 cache to retrieve the same object from a remote L3 cache or from the origin server:* $T_{local} << T_S, T_C$.

Local proxies are, by definition, either on the same local network or at the gateway to the intranetwork of their L1 clients. Under this configuration, the network delay for transferring an object from the local proxy to its end users is negligible compared to network delays from remote Internet sites. In some cases, the file system on the server may be faster than the file system on the local proxy. However, we assume that the server file system will seldom be fast enough to compensate for the slower network delivery, and therefore the assumption holds for the average times. When the local network delivery is much faster than remote network delivery, the L2 proxy and its end users experience approximately the same server response times. That is, T_S is similar for both the L2 proxy and the end user.

Assumption 2 *The average ratio of L3 cache response time to server response time,* T_C/T_S*, is constant with respect to hit rate, object size and cooperation set size.*

In reality, increasing either the L3 hit rate or the cooperation set size shifts the load from the Web servers to the proxies, increasing system and network contention for the caches while decreasing contention at the servers. This shift would improve Web server performance at the expense of cooperation performance. Consequently, violating this assumption causes proxy cooperation to be less viable than our results suggest. In Section 4.2, we address the dependence of T_C/T_S on object size. Our measurements show that daytime ratios of T_C/T_S are relatively independent of object size for small and moderate size objects.

Assumption 3 *Web servers, L2 caches and L3 caches are distributed randomly across the network.*

Cache placement affects both hit rate and cache response time. Response time, T_C, improves if caches are topologically close or lie enroute to the origin server. However, restricting requests to a subset of proxies lowers the cooperation hit rate, h_3. Because T_C and h_3 are not independent, modeling different cache placements would require estimates of the pair $< T_C, h_3 >$ for each placement. Assuming random locations eliminates the need for multiple $< T_C, h_3 >$ measurements and allows use of average cache response time, T_C, and hit rate, h_3.

Assumption 3 excludes special cases in which the parent caches are placed at border routers between regional networks and backbone providers, or cases in which parent caches are located at popular transoceanic links. We return to this issue in Section 9. While this assumption affects the mesh and hierarchy models, it does not affect the ideal case: placing proxy caches at optimal locations will not alter the upper bound on response time speedup.

3 Cooperation models and speedup equations

Response time speedup, S, is the ratio of task time without cooperative proxy caching to task time with cooperative proxy caching:

$$S = \frac{T_{without}}{T_{with}}. \quad (1)$$

Without cooperation, all local proxy (L2) misses are sent directly to the origin Web server and the average response time, $T_{without}$, is given by:

$$T_{without} = h_2 T_{local} + (1 - h_2) T_S. \quad (2)$$

With cooperation, the fraction of L2 misses satisfied by L3 proxies is h_3 and the fraction sent to the origin Web server is $1 - h_3$. Proxy cooperation adds overhead costs which may differ for L3 hits and misses. Incorporating overhead terms for L3 hits, $T_{o,hit}$, and L3 misses, $T_{o,miss}$, the expression

for average response time with cooperation becomes:

$$T_{with} = h_2 T_{local} + (1 - h_2) \times [\, h_3(T_C + T_{o,hit}) + (1 - h_3)(T_S + T_{o,miss}) \,]. \tag{3}$$

Assumption 1 allows us to neglect the contribution from local proxy hits, $h_2 T_{local}$, in Eqs. 2 and 3 if h_2 is not approximately equal to 1. Measurements support this assertion, with reported hit ratios for local proxies ranging from 0.2 to 0.7 [7, 24, 37]. With Assumption 1, the cooperation response time and speed equations simplify to:

$$T_{with} = (1 - h_2) \times [\, h_3(T_C + T_{o,hit}) + (1 - h_3)(T_S + T_{o,miss}) \,] \tag{4}$$

$$S = \frac{1}{h_3(\frac{T_C + T_{o,hit}}{T_S}) + (1 - h_3)(1 + \frac{T_{o,miss}}{T_S})}. \tag{5}$$

3.1 Discovery efficiency $\mathcal{E}$

The observed cooperation hit ratio, h_3, depends upon both the inherent sharing potential in the request stream and upon the effectiveness of the resource discovery mechanism. If a local proxy cannot find the cached copies, the request generates a false miss and lowers the L3 hit rate. A trade-off exists between the success of a discovery mechanism and its overhead. Suppose, for example, that each proxy maintains a local metadata directory containing information about remotely cached objects. Updating the directories more frequently reduces the number of false misses, but the update traffic and processing time can interfere with concurrent cache requests.

To separate the effects of false misses and discovery overhead in the models, we introduce the *discovery efficiency*, $\mathcal{E}$. For a maximum achievable hit rate, H_3, and a realized hit rate, h_3, the discovery efficiency, $\mathcal{E}$, is the fraction of potential hits for which the requesting proxy successfully locates the cached copies:

$$\mathcal{E} = \frac{h_3}{H_3}, \qquad 0 \leq \mathcal{E} \leq 1. \tag{6}$$

$\mathcal{E} = 0$ when cached copies exist, but in all cases the system fails to locate them. $\mathcal{E} = 1$ when discovery is perfect. Stated differently, the fraction of false miss is given by the expression $1 - \mathcal{E}$.

Query discovery methods such as ICP have an efficiency of 1, assuming all query messages arrive successfully.[1] Discovery efficiency for cache directories depends upon the match between the set of requested objects and the information in the cache directory: $\mathcal{E} = 1$ when cache directories contain all necessary metadata, and $\mathcal{E} = 0$ when the cache directories contain no useful metadata. Using discovery efficiency, the speedup equations can now be written in terms of the maximum hit rate for the workload, H_3, and the architecture-dependent overhead and efficiency terms:

$$T_{with} = (1 - h_2) \times [\, \mathcal{E} H_3 (T_C + T_{o,hit}) + (1 - \mathcal{E} H_3)(T_S + T_{o,miss}) \,] \tag{7}$$

$$S = \frac{1}{\mathcal{E} H_3 (\frac{T_C + T_{o,hit}}{T_S}) + (1 - \mathcal{E} H_3)(1 + \frac{T_{o,miss}}{T_S})}. \tag{8}$$

3.2 Upper bound

The upper bound on average cooperation speedup, corresponds to the case in which remote proxy caches are infinitely fast ($T_C = 0$), there is no overhead ($T_{o,hit} = T_{o,miss} = 0$), and discovery is perfect ($\mathcal{E} = 1$). These assumptions result in an expression for the maximum possible speedup:

$$S_{max} = \frac{1}{1 - H_3}. \tag{9}$$

Eq. 9 reflects Amdahl's Law, which states that a performance gain is limited by the fraction of cases that can take advantage of the improvement. For cooperative proxy caching, this limit corresponds to the duplication in L2 proxy requests streams for cacheable objects.

[1] ICP discovery can produce a false hit if the peer cache purges an object after sending the query reply and before receiving the associated request. As this interval is usually less than a second and objects typically remain cached for hours or days, the probability of false hits is considered negligible.

3.3 Mesh

The mesh and hierarchy architectures define how objects are delivered from the origin server to the end user, but are not necessarily associated with a particular discovery mechanism. This section develops a general speedup equation for the proxy mesh that is valid for any discovery mechanism. Specifically, the equation does not assume use of cache directories, either centralized or distributed. In a mesh, the L2 proxy cache sends all missed requests directly to the origin server. Delivery of objects for a miss is the same as for the case with no proxy cooperation. Consequently, there is no delivery component to the miss penalty, and the overhead terms reduce to the discovery overhead: $T_{o,miss} = T_{d,miss}$ and $T_{o,hit} = T_{d,hit}$. Substituting these overhead terms into Eq. 8 yields the general speedup expression for the mesh:

$$S = \frac{1}{\mathcal{E}H_3(\frac{T_C+T_{d,hit}}{T_S}) + (1-\mathcal{E}H_3)(1+\frac{T_{d,miss}}{T_S})}. \tag{10}$$

3.4 Mesh: ideal discovery

To compute the upper bound on speedup for a mesh, we consider the case of ideal discovery. Ideal discovery is defined as perfect discovery ($\mathcal{E} = 1$) with no overhead ($T_{d,miss} = T_{d,hit} = 0$). For a mesh with ideal discovery, the speedup expression in Eq. 8 simplifies to an expression that depends only upon the maximum hit rate and the average response time ratio:

$$S_{mesh,Ideal} = \frac{1}{1 - H_3 + H_3\left(\frac{T_C}{T_S}\right)}. \tag{11}$$

3.5 Mesh: no overhead, imperfect discovery

In real discovery mechanisms, the overhead and efficiency terms are coupled. Here we model a discovery mechanism that sacrifices discovery success in order to reduce discovery overhead to

a negligible amount. For a mesh with no discovery overhead but imperfect discovery ($T_{d,miss} = T_{d,hit} = 0, \quad \mathcal{E} < 1$), the speedup expression in Eq. 8 becomes:

$$S_{mesh,\mathcal{E}} = \frac{1}{1 - \mathcal{E}H_3 + \mathcal{E}H_3\left(\frac{T_C}{T_S}\right)}. \tag{12}$$

A particular realization of this model is the use of distributed metadata directories with an off-line method for propagating directory updates. Each proxy maintains a local directory containing metadata for other cache sites. Discovery overhead depends upon the costs of local directory lookups and updates, and efficiency depends upon the frequency of the updates. From Assumption 1, local directory lookups are, on the average, significantly faster than remote network transfers, thereby reducing directory overhead to the cost of updates. Update overhead depends on whether update operations directly affect the latency of a cache request. For example, a cache may delay returning an object until it checks object freshness with the origin server. If prefetching is used, the server or a peer cache may piggyback extra information onto returned objects. A distributed directory cache can be modeled with Eq. 12 if all directory updates occur off-line. That is, update information is sent independently of cache requests and replies and does not compete for bandwidth or system resources. Section 4.4 shows how these assumptions can be approximately satisfied in practice using a nighttime directory update scheme.

Closer examination of Eq. 12 reveals that the mesh architecture guarantees equal or faster response times when discovery overhead is negligible and caches are at least as fast as servers ($T_C \leq T_S$). This guarantee holds for all values of the hit ratio and discovery efficiency.

3.6 Hierarchy

For L3 hits in a three-level cache hierarchy, the object is delivered to the local proxy from either its parent or a sibling cache with a transfer time T_C. However, object delivery for L3 misses involves

two remote network transfers: one from the origin server to a parent cache, and a second from the parent cache to the local proxy. Assuming randomly distributed parents (Assumption 3) and store-and-forward delivery, the total delivery time for an L3 miss is the sum of these two transfers. On the Internet, this additional network transfer can incur a significant cache miss penalty [30]. Pipelining can reduce the delivery cost for larger objects. However, most Web objects are under 4 KB [25] and the effectiveness of pipelining small objects is limited by the extra connection set-up time and TCP's slow-start mechanism. In [28], Rousskov estimates that 80% of all requests do not benefit from pipelining due to the small size of the object.

Delivery time for a hierarchy miss can then be written as $T_{S,parent} + T_C$, where $T_{S,parent}$ is the average delivery time from an origin server to a parent cache. If response times from Web servers are, on the average, the same for L2 and L3 caches, then $T_{S,parent} = T_S$ and the total delivery time for a miss is $T_S + T_C$. Although parent caches may be provisioned with higher bandwidth than local proxies, this assumption still holds if delivery bottlenecks occur primarily within the network or at the origin server, rather than at the caches' network access point.

With no cooperation, the delivery time from an origin server to a local proxy is T_S. For the assumptions above, the delivery time for an L3 miss is $T_S + T_C$, with a corresponding miss penalty of T_C. As with the mesh, the overhead for a cache hit depends only upon discovery: $T_{o,hit} = T_{d,miss}$. Overhead for a cache miss in the hierarchy, however, is the sum of the discovery overhead and the cache miss penalty: $T_{o,miss} = T_{d,miss} + T_C$. Substituting these expressions into Eq. 8 results in the speedup equation for the three-level cache hierarchy model:

$$S_{hier} = \frac{1}{\mathcal{E}H_3(\frac{T_C+T_{d,hit}}{T_S}) + (1-\mathcal{E}H_3)(1+\frac{T_{d,miss}+T_C}{T_S})}. \quad (13)$$

3.7 Hierarchy: ideal discovery

For a hierarchy with ideal discovery ($\mathcal{E} = 1, \quad T_{d,miss} = T_{d,hit} = 0$), Eq. 13 simplifies to:

$$S_{hier,Ideal} = \frac{1}{1 - H_3 + \frac{T_C}{T_S}}. \tag{14}$$

Unlike the mesh, a proxy hierarchy *increases* user response time unless $H_3 > T_C/T_S$. This condition arises from the need to overcome the miss penalty for indirect delivery: smaller hit ratios generate higher miss penalties which must be compensated for by a greater cache speed advantage.

3.8 Hierarchy: no overhead, imperfect discovery

The cache digest protocol supported by the Squid hierarchical cache [33] uses local metadata directories for resource discovery. If we assume, as we did for the mesh, that cache digests or similar metadata summaries are exchanged off-line with negligible interference and that there are no other updates, then $T_{d,hit} = T_{d,miss} = 0$ and the speedup is:

$$S_{hier,\mathcal{E}} = \frac{1}{1 - \mathcal{E}H_3 + \frac{T_C}{T_S}}. \tag{15}$$

3.9 Hierarchy: ICP discovery

In the ICP protocol [34, 35], the child cache sends an ICMP echo request (ping) to its parent at the same time it sends ICP queries to its siblings. If its parent's echo reply arrives before a sibling ICP hit reply, the child cache immediately sends the request to its parent. Discovery overhead for an ICP miss therefore equals the parent RTT, while the discovery overhead for an ICP hit is the minimum of the parent RTT and the sibling hit replies. Echo and ICP reply times are typically similar, so we assume the same ICP overhead is incurred regardless of whether the request results in an L3 hit or miss: $T_{d,hit} = T_{d,miss} = T_{o,ICP}$. As discussed in Section 3.1, ICP guarantees perfect

Parameter	*Definition*	*Estimate*	*Source*
H_3	Maximum cooperation hit rate (current caching constraints)	0.31	Sec 4.1
T_C/T_S	Response time ratio	0.30	Sec 4.2, Ref [9]
$\mathcal{E}_{Night}$	Discovery efficiency: directories with nighttime update	0.92	Sec 4.4
$T_{o,ICP}/T_S$	Overhead ratio for ICP discovery	0.10	Sec 4.3, Ref [12]

Table 2: *Empirical parameters used to estimate speedup for the various cooperation models.*

discovery if all ICP packets arrive successfully. For this case, $\mathcal{E} = 1$ and Eq. 13 simplifies to:

$$S_{hier,ICP} = \frac{1}{1 - H_3 + \frac{T_C + T_{o,ICP}}{T_S}}. \tag{16}$$

4 Parameter estimates

The speedup equations for the cooperation models developed in the previous section are parameterized by one or more of the terms H_3, $\mathcal{E}$, T_C/T_S, and $T_{o,ICP}/T_S$. In this section, we describe the experiments used to determine numerical estimates for these parameters and discuss the sensitivity of the computed speedups to inaccuracies in the parameter estimates. All parameter estimates are empirical values derived from studies reported here or in our previous work [9], with the exception of the ICP overhead term ($T_{o,ICP}$) from Fan *et al.* [12]. Table 2 summarizes the parameter estimates used in the various cooperation models and lists the associated references.

4.1 L3 hit rate: H_3

The upper bound on speedup for proxy cooperation depends only upon H_3, the maximum L3 hit rate. Our estimate for H_3 is computed from the recorded cache hits in traces from two L3 parent caches in the NLANR national cache hierarchy. NLANR traces were chosen for several reasons. First, NLANR caches use the ICP protocol, which guarantees perfect discovery (see Section 3.1).

Consequently, the traces reflect the maximum hit rate for the proxies in the cooperation set, assuming the sibling hit rate is also known. Second, the NLANR parent caches have a relatively large L2 cooperation set with a heterogeneous client population. Finally, traces from NLANR caches are publicly available, permitting the estimates to be corroborated by other researchers.

4.1.1 Traces

Traces in this analysis were recorded at the NLANR Boulder (BO1) and Urbana-Champagne (UC) caches and made publicly available by the NLANR project [22]. BO1 serves as parent cache to approximately 90 L2 proxies and UC serves as parent to approximately 65 L2 proxies. Traces record the cache action (hit or miss), as well as type of request, error status, and cache directives in the request header. We analyzed 16 BO1 and 16 UC traces collected from January 27 to February 11, 2000. Each trace spans one day and contains from 400,000 to 900,000 requests for a combined total of 9 million requests for BO1 and 11 million for UC. The analysis computes separate hit rates for each trace, which are averaged separately for BO1 and UC. Table 3 summarizes the statistics and average hit rates for these 32 traces. Only successful GET requests are included, since POST, HEAD and requests that generate an error status are neither cache hits nor cache misses. The average hit rate across 16 daily traces is 0.30 for BO1 and 0.24 for UC.

4.1.2 Correction for sibling hits

Hit rates computed from NLANR and other Squid traces underestimate H_3 because they do not include sibling hits between the L2 caches. As the parent cache is unaware of sibling requests, these do not appear in the parent's trace and are instead recorded in the *child's* trace. In [10], we derive the following expression to correct for sibling hits:

$$H_3 = \frac{H_{3p} + f_s/f_p}{1 + f_s/f_p} \tag{17}$$

	B01	*UC*
Number of traces	16	16
Length of each trace	1 day	1 day
Requests per trace	563956	691770
Successful GET requests per trace	276481	397363
Documents per trace	198363	290923
Observed hit rate		
Before sibling correction	0.30	0.24
After sibling correction	0.34	0.28

Table 3: *Statistics and observed hit rates for NLANR BO1 and UC cache traces, Jan 27 - Feb 11, 2000. Values in the table are averages for the 16 traces.*

where H_{3p} is the hit rate from the L3 parent trace, f_s is the fraction of L2 requests sent to sibling caches, and f_p is the fraction of L2 requests sent to the parent cache. Eq. 17 assumes that all requests sent to siblings are L3 cache hits, which follows from the ICP protocol (see Section 3.1). The parent hit rate, H_{3p}, is determined from the parent's trace, while the ratio of sibling requests to parent requests, f_s/f_p, is determined from the child traces. When most requests are sent to the parent cache, the denominator approaches unity and f_s/f_p approximates the sibling hit rate.

Ideally, sibling request ratios should be measured for the same set of L2 caches that appear in the BO1 and UC traces. As this data was unavailable, we estimate the sibling correction using L3 sibling data in the BO1 and UC traces, and L2 sibling data in an unrelated L2 trace. In the BO1 and UC traces, f_s/f_p ranges from 0.03 – 0.08, with a median of 0.05. For the L2 data, we used 31 daily traces from Uppsala University recorded during March, 2000 [32]. The average number of requests per trace is 548,000, for a total of approximately 17,000,000 requests. In the Uppsala traces, f_s/f_p ranges from 0.04 – 0.10, with a median value of 0.06. Although basing the sibling correction upon the f_s/f_p ratio from different cache sets assumes comparable ratios, the agreement between the NLANR value of 0.05 and the Uppsala value of 0.06 lends confidence to the estimate. Moreover, for $f_s/f_p = 0.06$, the L2 sibling hit rate computed from Eq. 17 is 4%, which is similar

to observed sibling hit rates reported by NLANR and elsewhere [22, 18].

Applying the correction with $f_s/f_p = 0.06$ increases the hit rate from 0.30 to 0.34 for BO1, and from 0.24 to 0.28 for UC. We average these corrected L3 hit rates from BO1 and UC to obtain the parameter estimate $H_3 = 0.31$.

4.2 Response time ratio: T_C/T_S

The benefit of using speedup is that cache and server response times appear as a ratio rather than as independent parameters. Because response time varies dramatically with client connectivity and current load, server and cache response times can only be compared for the same client set and load conditions. A ratio mitigates this problem by coupling server and cache measurements. A pair of server and cache response times is measured for the same client and at approximately the same time. Although absolute response times vary substantially across clients and time-of-day, we observe that the ratio T_C/T_S ratio remains relatively constant (Table 4).

The estimated response time ratio, T_C/T_S, is taken from our earlier experiments on server selection algorithms. A brief description of the experiments is reported here; more detail is available in [9]. In these experiments, clients have a choice of remote servers from which they can download an object. Various selection methods are measured, including random assignment, prior bandwidth or latency, and dynamic RTT probes similar to an ICMP echo (i.e., ping). Random assignment is analogous to the situation where there is no cooperative caching and clients must download the object from the origin server (i.e., T_S). Other selection methods represent the case of cooperative caching where the object is stored at multiple sites and the client chooses which site to use (i.e., T_C).

4.2.1 Experimental methodology

Experiments were run concurrently on clients located at Texas A&M University (A&M), the University of Houston (UH), and the University of Texas at San Antonio (UTSA). Client sites are located in different cities, connect to different regional networks, and use different primary Internet backbones. All clients have T1 or better connections and access the same set of servers in the experiment. To model a widespread national caching system, the server set must be well-distributed geographically and topologically. Servers also must be stable, popular, and reasonably well-connected. To fulfill these conditions, we used the official Web servers for U.S. state governments: `www.state.**.us`, where `**` denotes the state postal abbreviation. A few states were omitted because the test object moved during the experiment, or because the server did not respond to the dynamic probes. Figure 3 shows the locations of the three clients and the 42 servers in the study.

Direct comparison of response times from different sites requires objects of the same size at each site; however, we have no control over server content. Our solution is to use objects of approximately equal size and normalize response times. The normalization procedure is as follows. We measure the total response time T, and its components $T_{Connect}$, $T_{Latency}$, and $T_{Remaining}$. $T_{Connect}$ is the time to establish the connection, $T_{Latency}$ is the time from sending the GET request to the receipt of the first packet, and $T_{Remaining}$ is the time to receive the remaining reply packets. DNS time is avoided by using the IP address, as would normally be the case in a proxy cooperation set. We also record the total number of bytes, N, and number of bytes in the first read, $N_{Latency}$. For objects of approximately the same size, we expect similar values for $T_{Connect}$ and $T_{Latency}$, with differences appearing in $T_{Remaining}$. On congested networks, the bit rate for remaining reads is approximately independent of file size [9], allowing the response time to be normalized to a

reference size $\widehat{N}$:

$$\widehat{T} \;=\; T_{Connect} + T_{Latency} + T_{Remaining}\left(\frac{\widehat{N} - N_{Latency}}{N - N_{Latency}}\right) \tag{18}$$

Two objects were chosen on each server, an HTML file of approximately 6 KB and an image file of approximately 50 KB. Using Eq. 18, the measured response times are normalized to 6 KB for HTML files and 50 KB for image files.

Data was collected from March 1999 to May 1999, with over 1600 measurement sessions per client. A measurement session begins by randomly picks ten unique candidates from the pool of 42 servers, representing the scenario where an object is available at 10 remote caches. Within a session, each selection algorithm picks one site from these 10 candidates and the object is immediately downloaded. A session loops over the selection algorithms in random order, pausing three minutes between algorithm measurements to avoid creating network congestion.

4.2.2 Experiment results

To account for time-of-day variations in server load and network congestion, the data is split into daytime and nighttime categories. Figure 4 shows a representative plot of median response times for each 3 hour period. On weekdays, response time increases sharply around 9 am and drops around 5 pm. Accordingly, the daytime period is defined as M–F, 9am–5pm, and the nighttime period as nights and weekends.

Table 4 compares median response times when server sites are chosen randomly ($\widehat{T}_{Random}$) and when the client selects the "best" site using a RTT probe ($\widehat{T}_{Probe}$). Times are for data collected weekdays M–F, 9am–5pm, and are analyzed separatedly for the 6 KB and 50 KB file sizes. $\widehat{T}_{Probe}/\widehat{T}_{Random}$ represents the expected response time ratio T_C/T_S when an object is available at multiple remote caches. As shown in Table 4, this ratio varies from 0.19 for the UTSA client

	6 KB			50 KB		
	$\widehat{T}_{Probe}$	$\widehat{T}_{Random}$	$\frac{\widehat{T}_{Probe}}{\widehat{T}_{Random}}$	$\widehat{T}_{Probe}$	$\widehat{T}_{Random}$	$\frac{\widehat{T}_{Probe}}{\widehat{T}_{Random}}$
A&M	0.5	1.2	0.42	1.1	5.6	0.20
UH	0.7	2.1	0.33	3.8	10.1	0.38
UTSA	1.5	8.0	0.19	11.2	30.3	0.37
Geom. mean			**0.30**			**0.30**

Table 4: *Median response times (normalized) in seconds for the A&M, UH, and UTSA clients, M–F, 9am–5pm. The geometric mean of $\widehat{T}_{Probe}/\widehat{T}_{Random}$ provides the estimate for T_C/T_S.*

to 0.42 for the A&M client. Taking the geometric mean across clients yields a mean of 0.30 for both the 6 KB and 50 KB file sets. We use this value as the estimate of the response time ratio: $T_C/T_S = 0.30$.

4.3 ICP overhead ratio: $T_{o,ICP}/T_S$

As discussed in Section 3.1, ICP [34] is a query discovery mechanism commonly used in proxy hierarchies. In theory, query methods can also be applied in meshes. However, queries scale poorly because the number of replies is proportional to the number of peer proxies. Whereas this overhead is tolerable for sibling subsets in a hierarchy, it is unreasonable for the complete set of proxies in a mesh.

The query request/reply messages of ICP add latency to every request that is approximately equal to the fastest RTT in the proxy set (including the parent cache). Fan *et al.* [12] measured ICP overhead for four proxies on a LAN and found that ICP queries increase the average latency of an HTTP request by 8% to 12%. Based on these results, we estimate ICP overhead ratio to be $T_{o,ICP}/T_S = 0.10$.

4.4 Discovery efficiency: $\mathcal{E}_{Night}$

The mesh and hierarchy models in Sections 3.5 and 3.8 assume there is no discovery overhead, and that discovery is not perfect. These models can be realized by a discovery mechanism that maintains L3 cache directories at each local proxy and updates these directories off-line. Off-line implies that no update information is sent with cache requests or with returned objects, and that update traffic does not delay replies by competing with HTTP traffic for bandwidth or system resources.

4.4.1 Overhead for nighttime updates

Update interference can be reduced by restricting the directory updates to periods of low network and system demand. As shown in Figure 4, HTTP response times on the Internet are significantly faster at night than during the day. Statistics reported by the NLANR project [23] show an analogous pattern in the request rates at the NLANR parent caches, exhibiting clear peaks during daytime hours and troughs at night. Similar diurnal patterns are reported by Chandra *et al.* [4] and Wolman *et al.* [36] for Web caches and by Thompson *et al.* [31] for Internet traffic in general. Propagating directory updates during a quiescent nighttime period will not interfere with requests at other times of the day. Because most requests occur during the day, updating late at night exposes only a small fraction of requests to update interference. Moreover, networks typically have excess capacity at night, further reducing the potential for traffic interference. Consequently, an L3 discovery mechanism with distributed directories located on the L2 proxies using only nighttimes updates will have negligible discovery overhead.

4.4.2 Update simulation

The disadvantage of restricting updates to late night is that infrequent updates cause false misses (i.e., $\mathcal{E} < 1$) and lower L3 hit rates. To estimate the fraction of false misses incurred by infrequent updates, we use the 16 daily BO1 and UC traces in Table 3 to simulate hit rates for two update scenarios:

Continuous update: Cache metadata directories are updated as soon as the cache receives a new object. Continuous update produces the maximum hit rate ($\mathcal{E} = 1$).

Nighttime update: Cache metadata directories are updated only at midnight with information about objects cached on the prior day.

In the continuous update simulation, cache hits are requests in the trace that 1) are recorded as hits, and 2) have URLs previously requested in the current day's trace or in a prior day's trace. The first requirement eliminates false hits from the simulation while the second requirement calibrates for an inadequately warmed simulation cache. If the simulation cache contains the same objects as the real cache, the continuous update hit rate will equal the recorded hit rate.

In the nighttime update simulation, cache hits are requests in the trace that 1) are recorded as hits, and 2) have URLs requested in a prior day's trace. If an earlier request was received for the same object during the current day but none were received in prior days, the object would not yet be registered in the directory and the cache would suffer a false miss.

Figure 5 compares computed hit rates for the continuous update and nighttime update scenarios as a function of cache warming. The rise in both hit rate curves over the first few days shows the necessity of warming the simulation cache. Single day NLANR traces are prone to a large warmup effect: 60% to 70% of the recorded cache hits are for URLs that appear only once in that day's trace. Simulations that rely on a single day's trace risk incorrectly counting many real

hits as compulsory misses. Our results show this dramatically. Measured hit rates recorded in the Feb.11^{th} BO1 and UC traces are 0.30 and 0.24, respectively. However, the cold simulation cache causes the continuous update simulation to predict hit rates of only 0.08 for both traces, *missing* $2/3$ *to* $3/4$ *of the recorded hits.* After warming the simulation cache with 2 weeks of traces, the predicted hit rates rise to 0.27 for BO1 and to 0.22 for UC. As Figure 5 shows, most false misses are eliminated after warming with 3 days of traces, although simulated hit rates continue to rise slowly with additional warming. Reference [11] gives a detailed description of the cache warmup effect and trace correction methods.

For the nighttime update scenario, a warm simulation cache produces hit rates of 0.25 for BO1 and 0.20 for UC. Thus updating cache directories once daily (at midnight) preserves 91% – 93% of the cache hits. The results may seem higher than expected because cache digest developers suggest update frequencies of an hour or less [12, 29]. However, Maltzahn *et al.* [19] report a similar nighttime strategy for prefetching HTTP documents significantly reduces peak bandwidth usage without compromising cache consistency. Our results on diurnal load and update frequency suggest that L3 directories can achieve a discovery efficiency of $\mathcal{E} = 0.91$ to 0.93 with negligible discovery overhead.

5 Speedup results

We define proxy cooperation to be viable with respect to user response time if cooperation does not increase average delay for the end user. This definition corresponds to the requirement that $S \geq 1$. A speedup less than 1 is actually a slowdown in which users experience *longer* average delays as a result of cooperation. In this section, we predict numerical speedups for the different cooperation models by substituting parameter estimates from Table 2 into the equations derived in Section 3.

5.1 Results for upper bound

The upper bound on speedup in Eq. 9 applies to all cooperation architectures and depends only upon the maximum L3 hit rate. For the parameter estimate $H_3 = 0.31$, cooperation speedup cannot exceed 1.4. That is: $S_{max,Real} = 1.4$, where the *Real* subscript refers to real cache constraints associated with the observed L3 hit rates. Figure 6 explores the sensitivity of the results to inaccuracies in parameter estimate by plotting the speedup upper bound as a function of H_3 (Eq. 9). For hit rates under 0.5, speedup varies slowly, indicating that moderate errors in H_3 have relatively small effects on the result. Beyond this point, speedup improves more rapidly, asymptoting as hit rate approaches unity. The graph illustrates why caching is much more successful at high hit rates. It also points out that modest improvements in the cooperation hit rate have little effect on average user response time. In Section 8 we simulate the case where all objects are cacheable and find full cacheability increases hit rate from 0.31 to 0.40. However, this rise in the L3 hit rate produces only a modest increase in the upper bound: $S_{max,All\_Cacheable} = 1.7$.

5.2 Results for mesh and hierarchy

For the mesh and hierarchy architectures, we first compute speedups for the case of ideal discovery. Here speedup depends upon the maximum hit rate and the response time ratio T_C/T_S. Substituting the parameter estimates $H_3 = 0.31$ and $T_C/T_S = 0.30$ into Eqs. 11 and 14 gives speedup limits of $S_{mesh,Ideal} = 1.28$ and $S_{hier,Ideal} = 1.01$.

Now consider discovery by distributed metadata directories restricted to nighttime updates (Section 4.4). Directory overhead remains negligible, but efficiency drops to the empirical estimate of $\mathcal{E}_{Night} = 0.92$. For this case, Eqs. 12 and 15 produce speedups of $S_{mesh,\mathcal{E}} = 1.25$ and $S_{hier,\mathcal{E}} = 0.98$.

The lowest speedup results are for the cache hierarchy with the ICP discovery. Substituting the

overhead estimate of $T_{o,ICP}/T_S = 0.10$ into Eq. 16 yields the speedup $S_{hier,ICP} = 0.90$.

5.3 Comparison of upper bound, mesh, and hierarchy

The upper bound of $S_{Max,Real} = 1.4$ suggests that cooperation can improve user response time if the miss penalty and discovery cost are sufficiently low. The bar graph in Figure 7 compares this upper bound to speedups for the mesh and hierarchy models. A dashed line at $S = 1$ indicates the viability boundary; below this line cooperation degrades rather than improves average response time. As is evident from the graph, the mesh design clearly outperforms the hierarchy and appears viable for ideal discovery and for distributed directories with nighttime updates.

By contrast, the hierarchy is barely viable even when discovery is ideal, and falls below the boundary for the directory and ICP discovery models. The fundamental cause for the hierarchy's poorer performance is its higher L3 miss penalty. Because cooperative caching has a relatively large miss rate, the L3 miss penalty is heavily weighed and strongly influences cache performance.

Speedup results for the directory model depend on the estimated value of $\mathcal{E}$. If the trace analysis in Section 4.4 overestimates $\mathcal{E}$, it would overestimate speedup and possibly lead to incorrect conclusions. To explore this sensitivity, Figure 8 plots Eq. 12 to illustrate how speedup for the mesh varies with $\mathcal{E}$. Because speedup for the mesh also depends upon the T_C/T_S ratio, curves are plotted for the upper bound of $T_C/T_S = 0$ and for the measured value of $T_C/T_S = 0.30$. These curves show that large changes in efficiency have little effect on speedup: when $T_C/T_S = 0.30$, halving efficiency from 100% to 50% reduces speedup from 1.3 to 1.1. This leads to the observation that cooperating proxies can better reduce user delay by minimizing overhead rather than maximizing hit rate.

6 Viability conditions

We can gain insight into the causes for the speedup results by examining the condition $S \geq 1$ for different models. For the mesh-directory model using nighttime updates, the inequality holds if

$$\frac{T_C}{T_S} \leq 1. \tag{19}$$

To see this, require that $S_{mesh,\mathcal{E}}$ in Eq. 12 be ≥ 1 and simplify the inequality. The mesh-directory model reduces average delay and passes the viability test as long as caches can offer a small speed advantage.

For the hierarchy-directory model with nighttime updates, applying the condition $S \geq 1$ to Eq. 15 yields the condition:

$$\frac{T_C}{T_S} \leq \mathcal{E}H_3 \tag{20}$$

where $\mathcal{E}H_3$ is the actual hit rate realized by the cache system. To be considered viable with respect to response time, caches in a cooperation hierarchy must provide a speed advantage proportional to the realized hit rate. If cooperation hit rates were near 0.50, the cache system would have to be, on the average, twice as fast as the origin servers. Hit rates near the estimated maximum value of 0.30 require that the cache system respond 3 times as quickly as the origin servers.

For a proxy hierarchy using ICP discovery, the viability condition derived from Eq. 16 is:

$$\frac{T_C}{T_S} \leq H_3 - \frac{T_{o,ICP}}{T_S}. \tag{21}$$

Here the cache speed advantage must be large enough to overcome both the miss penalty and the added latency of ICP queries. The overhead term is independent of the cooperation hit rate, therefore overhead assumes greater important when hit rate is low. This inequality shows explicitly why overhead plays an important role in cooperation performance.

7 Benefits of cooperation

Reduction in average user response time is not the only criteria by which proxy cooperation should be judged. From a server's viewpoint, the main benefit of proxy cooperation is the reduction in server load. For the network, proxy cooperation may better distribute traffic, thereby reducing congestion near busy servers. In this section, we show that although proxy cooperation may not offer a large reduction in average response time, it can reduce the variability in the delay and reduce the number of pathologically long delays.

Response time distributions are notoriously heavy-tailed [20] and not well-characterized by a single value. Users are known to be sensitive to other aspects of the distribution, especially the variability in response time and number of long delays (i.e., the distribution tail). To determine how cooperation affects the tail of the response time distribution, we compute cumulative distribution functions (CDFs) for the $\widehat{T}_{Random}$ and $\widehat{T}_{Probe}$ data from the experiment described in Section 4.2, where $\widehat{T}_{Random}$ models the response time for origin servers while $\widehat{T}_{Probe}$ models the response time for cooperating caches.

Figure 9 shows the cumulative distribution functions (CDFs) for the *Random* and *Probe* response times at the Texas A&M client. When an object is available at only one site, the maximum response time is 18 minutes. When an object is available at 10 sites and the client uses a dynamic probe to select the site, the maximum response time falls to 2 minutes. Providing clients a choice of sites reduces the variability in response times and eliminate most of the extremely long delays. Similar CDFs were observed for the UH and UTSA clients. This data suggests that response times from a cooperative system would reduce the pathological behavior seen in the servers' distribution.

A low cooperation hit ratio dampens this improvement because multiple cache sites are useful only when there is an L3 hit. The middle curve in Figure 9 is a simulated CDF based on the empirical hit rate from Section 4.1. The simulated CDF was computed by selecting random points

Distribution	Model	Fraction of Response Times $> 55sec$	$> 108sec$	$> 197sec$
Random	Server only	0.10	0.05	0.01
Simulation	Cooperating caches, 31% hit rate	0.07	0.03	0.007
Probe	Cooperating caches, 100% hit rate	0.01	0	0

Table 5: *Comparison of distribution tails for the* Random *and* Probe *experiments, and for the simulated cooperation distribution with a 31% hit rate. (Texas A&M client, 50 KB objects, 10 cache sites).*

from the *Random* distribution with probability equal to the L3 miss rate (0.69), and from the *Probe* distributions with probability equal to the L3 hit rate (0.31). When L3 hit rates are taken into account, the length of the tail is not dramatically reduced. However, the total number of pathological delays is reduced by a factor approximately equal to the cooperation hit rate, as is the variability in user response time.

Table 5 compares the tails of the *Random*, *Probe*, and simulated proxy cooperation distributions. The table shows the fraction of delays longer than the 90th, 95th and 99th percentile of the *Random* distribution. For example, at the 90th percentile, 10% of the *Random* responses are longer than 55 seconds, compared with 1% of the multiple site/probe times and 7% of the simulated proxy cooperation times. These comparisons quantify the benefit cooperative proxy caching offers in reducing variability and the number of pathologically long delays.

8 Effect of caching constraints

Caches normally do not store dynamic objects such as query or cgi-bin responses, nor do they store objects when expressly forbidden by cache control directives in either the HTTP request header or HTTP reply header. The measured H_3 hit rate in Section 4.1 reflects current constraints on object caching. If more objects could be cached, the L3 hit rate would increase and cooperation might

appear more attractive. To estimate the effect of caching constraints on cooperation hit rate and the speedup upper bound, we assume all objects are cacheable and compute the hit rate from a trace-driven simulation using the BO1 and UC trace sets. A request generates a cache hit if either 1) the trace recorded a hit, or 2) the URL was previously requested by a *different* client.

The simulation filters repeat requests issued by the same client from the L3 request stream, counting them as neither hits nor misses. If an object could be cached, it would be cached at the browser and local proxy. Subsequent requests for the object from the same client would be satisfied by these lower level caches and thus removed from the L3 request stream. This filtering has a substantial impact on the hit rate for ideal cacheability: over 90% of duplicate requests for uncacheable objects in the BO1 and UC traces are repeat request from the same client. As a result, a simulation that counts repeat requests as cache hits substantially overestimates the L3 hit rate. With filtering of repeat requests, the average simulation hit rate[2] for ideal cacheability is 0.40. Without filtering, the repeat requests inflate this value to 0.53.

Comparing simulation results for ideal cacheability to measured results for real cacheability, we observe that removing cache constraints raises the average hit rates by nearly 0.10 (from $H_{3,Real} = 0.31$ to $H_{3,All\_Cacheable} = 0.40$). Ideal cacheability generates few addition L3 cache hits; the higher hit rate is instead due to the filtering of repeat requests. With filtering, the L3 caches receive fewer total requests, and almost all of the filtered requests produced cache misses in the measured trace. Table 6 summarizes the results for the measured and simulation L3 hit rates.

To estimate the effect of caching constraints on user response time, we substitute the simulation hit rate of 0.40 into Eq. 9 to obtain an upper bound of $S_{max,All\_Cacheable} = 1.7$. As this limit is not substantially larger than the upper bound for current caching constraints of 1.4, we conclude that caching constraints have a modest impact on average response time. Once again, this is due to the dampening effect of Amdahl's law in regions where the performance improvement is limited to

[2]Simulation hit rates are corrected for sibling hits as described in Section 4.1.

	B01	*UC*	*Mean*
Measured hit rate : *Real cache constraints*	0.34	0.28	0.31
Simulated hit rate : *All objects cacheable*			
No filtering of repeat requests	0.54	0.52	0.53
With filtering	0.45	0.36	0.40

Table 6: *Measured and simulated hit rates for proxy cooperation for BO1 and UC cache traces, Jan 27 - Feb 11, 2000. Hit rates are corrected for sibling hits as described in Section 4.1.2.*

less than half of the events.

9 Other cache configurations

We now look at several important cache configurations that violate the random proxy placement assumption of Section 2.

9.1 Border caches

Border caches are caches located at network link points such as transoceanic endpoints, satellite links, routers connecting local networks to regional networks or routers connecting regional networks onto the Internet backbone. These caches can act as two-way filters for traffic passing in either direction across the link point. Since all relevant references are directed through the link point on which the cache resides, no overhead is incurred for discovery. Border caches do not strictly fit into the framework we have described for cooperative caching. Because a border cache lies along the network path to the origin server, its effect on user response time depends on the balance between the additional cache processing and the downstream time to the origin server. Caches on congested or low-capacity links can greatly improve response time, while overhead becomes more critical on high-speed, high-bandwidth lines.

9.2 Local and regional cache federations

Local and regional L3 cache federations present special cases where cooperation can be beneficial. Local cache federations are cooperation sets of L2 proxy caches residing on the same intranet. Regional cache federations have cache cooperation sets residing on the same regional network.

Not all L2 proxies are equal in their caching capacity, file system speed, or the number of clients served. Wolman *et al.* [37] show that L2 hit rate is an increasing function of client population, with a sharp knee at 2500 clients and little improvement beyond 20,000 clients. Krishnan and Sugla [17] observed an improvement in hit rates of $0.7 - 28.9\%$ for cooperating intranet proxy caches, with larger improvements for caches with smaller client sets. Although their estimates show that cooperation overhead can introduce significant processor load and network traffic, the increased L2 hit rate reduces the amount of traffic between the regional intranet and the global Internet.

When an origin server is on the same regional network, L2 clients would probably do better by avoiding cooperation and directing their requests to the origin server. If the origin server is not on the regional network, both the request path from client to the origin server and from an L3 proxy to the origin server pass through an entry point onto the Internet backbone. L3 hit rates then translate into fewer requests leaving the regional network to traverse the backbone. The ratio of cache to origin server response time is also likely to be small in this case (i.e., $T_C/T_S << 1$), indicating that if cache management overhead can be kept low, improvements in average user response time may be significant.

9.3 Content distribution networks

Content distribution networks (CDNs) offer a rival technology to cooperative caching. Krishnamurthy *et al.* [16] report an order of magnitude increase in the number and percentage of popular

origin sites using CDNs between November 1999 and December 2000. Clearly CDNs are becoming an influential factor.

CDNs differ from cooperative caching in several important respects. Users access a CDN directly, therefore CDN sharing is available to all users rather than being restricted to clients served by a local proxy. Unlike caches, the primary constituency of a CDN are the content providers rather than the end users. This shifts the focus from end user response time to server availability and load balance. CDNs use a push strategy to distribute replicas of an origin server's content to CDN servers placed at strategic locations on the network. Client requests are redirected to an appropriate CDN server, effectively replacing the origin server with the selected CDN server. Because the CDN controls object replication and dissemination, content distribution should be considered a server replication or supply-side strategy rather than a demand-side caching scheme. Given these differences, CDNs do not fall under the framework described by our models.

10 Related performance studies

Many performance studies on proxy cooperation have focused on characterizing cache sharing and hit rates [2, 3, 7, 14, 37]. For example, Cao and Irani [3], Breslau *et al.* [2], and others report the hit rate for proxy caches grows in a log-like fashion with number of clients and number of requests. For proxies serving a small number of clients, the hit rate grows rapidly as clients are added. However, the rate of improvement soon flattens and, after a certain point, the hit rate becomes relatively insensitive to population size.

In [37], Wolman *et al.* develop analytical performance estimates based on a tiered network model. They consider three cooperation schemes: hierarchical, hash-based and directory-based. The hierarchical design is similar to the one described in this paper, with addition of multiple levels in the cache hierarchy and a fanout of d at each level. In the hash-based scheme, one and only one

of the cooperating proxies caches a document. The scheme emphasizes load balancing among caches and savings of memory required to cache objects. The directory-based scheme is similar to the proxy mesh modeled in this paper. For the parameter set used by the authors, the analysis predicts slightly better response times for the hierarchical organization. Wolman *et al.* [37] find that cache cooperation improves hit rates for user populations under 20,000, but offers little gain beyond that level. Their results suggest that, based on hit rate alone, large-scale cooperation makes sense only in high-bandwidth, low-latency environments, such as might be found in a metropolitan area network.

Rodriguez *et al.* [27] also have developed analytical models for caching based on a tiered organization of local, regional and national networks. They consider a hybrid scheme in which caches cooperate at every network level of a caching hierarchy, and determine the optimal number of caches that should cooperate at each level in order to minimize the client's response time.

Tewari *et al.* [30] compare response times for a proxy hierarchy with a proposed mesh design using experimental measurements and simulation studies. The authors conclude that on wide area networks, the extra network transfers incurred by the hierarchical L3 misses lead to poorer response times as compared to a mesh. They do not consider the case of no proxy cooperation and therefore do not address the viability issue.

Doyle *et al.* [6] describe how widespread deployment of caching (mainly of the border type) and CDNs is reshaping network delivery system traffic patterns. Lower level caches filter the request stream, shifting the load from popular static objects to less popular and/or uncacheable dynamic content. Gadde *et al.* [13] apply the Wolman model to study marginal benefit of higher level caching to overall hit rates. They also study the effects of client population on the cacheable hit rate, the fraction of requests for cacheable documents that hit in the cache. Krishnamurthy *et al.* [16] report an extensive empirical study of the effect of CDNs on user response time, and

provide a methodology for CDN customers to assess improvement in user response time for particular configurations. Their results indicate that the overhead incurred by using small DNS TLLs to balance server load offsets the corresponding performance gains.

Our approach differs from other analytical cooperation studies in that we do not predict absolute performance measures, but instead use a relative performance metric referenced to the same standard case of no cooperation. A relative metric simplifies the analytical models and facilitates the separation of terms in the equations that represent fundamental design approaches. Moreover, relative metrics allow comparisons over a wide range of network, resource, and load conditions. We concentrate on ideal, simplified models to show the limits of cooperation performance for all cases and limits for basic proxy organizations.

11 Conclusions

This paper develops simplified models of cooperative proxy caching in order to isolate the factors that influence performance. We believe that the primary motivation for cooperative caching is reduction in user-perceived response time because the constituency of L2 cache operators consists of end users. While cooperation may better distribute server load and network traffic, these effects are not directly observed by L2 cache operators. The same amount of HTTP traffic flows through the L2 cache regardless of whether the source is a cooperation L3 cache or an origin server.

The main conclusion of the study is that proxy cooperation offers at best a small speedup in average user response time because the request filtering by browser and local proxies limits the cooperation hit rate. For the L3 caches in this study, the average hit rate is 0.31 with a corresponding speedup upper bound of 1.4. Improving cacheability has little effect. If all objects were cacheable, the cooperation hit rate would increase to 0.40 and the maximum speedup to 1.7. These speedup limits represent the ideal case of infinitely fast caches, perfect discovery, and no overhead.

With more realistic cache/server speed differentials and caching overhead, the realizable speedup is closer to unity. The analysis indicates that it is more important to eliminate cooperation overhead than to maximize hit rate. Our observations about the effectiveness of cache digests with nighttime exchanges apply to both intranet and federated regional cooperative caching. One can virtually eliminate discovery overhead for a small price in attainable hit rate. Our equations show that this is an excellent trade-off, as response times suffer more from cache overhead than from a diminished hit rate.

Of the cooperation models considered here, proxy meshes offer more speedup than proxy hierarchies, which we attribute to the hierarchy's larger miss penalty. With perfect discovery and no discovery overhead, the hierarchy is just over the viability boundary. Results suggest that a proxy hierarchy using the ICP protocol is likely to *increase* rather than reduce user delays. Our assumptions, however, do not hold for special proxy hierarchies in which parent caches are located directly along the route between the most popular Web servers and the local proxy cache.

One argument for cooperative Web caching postulates that increasing the number of clients increases cache sharing and therefore cache hit rates. Our results suggest that the hit rate would have to improve substantially to move beyond the upward knee of the speedup curve shown in Figure 6. However, cooperative caching has two major benefits both rooted in its ability offer clients a choice of sites in order to avoid congested routers and servers. The direct benefit is reduced variability in response time and fewer long delays. The indirect benefit is a better distribution of traffic and server load leading to an improvement in overall Internet throughput. We believe cooperation efforts should shift focus from maximizing hit rates to increasing distribution of popular objects and improving site selection methods. Clearly intranet and regional cache federations are in a better position to improve performance in these areas than are randomly distributed caches.

Finally, cooperative proxy caching involves trust and coordination. Wide-scale cooperative

caching that is not administered by a central authority would be difficult to implement. Cooperating proxies must trust that the objects they accept from other caches have not been modified, and that cooperating partners will not act in a manner contrary to the common good. Unlike CDNs, the economic issues of who pays to maintain cooperative arrangements are unclear. These factors limit the potential for large-scale cooperative caching and indicate that caching is likely to be dominated by content providers and network service providers.

Acknowledgements

We would like to thank Jeff Mogul for his careful reading and insightful comments on the conference version of this paper, especially with regard to the assumptions used in our analysis. We would also like to thank our editor, Ernst Biersack, for a number of thoughtful suggestions that improved the descriptions of the experiment and the discussion. Computing resources for a portion of this work were generously provided by the University of Houston and Texas A&M University.

References

[1] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox. Caching proxies: Limitations and potentials. In *Proc. of the 4th Int'l. World-Wide Web Conf.*, pages 119–133, Dec. 1995.

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proc. of IEEE Infocom'99*, Mar. 1999.

[3] P. Cao and S. Irani. Cost-aware WWW proxy caching algorithms. In *Proc. of the 1997 USENIX Symp. on Internet Technology and Systems (USITS'97)*, pages 193–206, Dec. 1997.

[4] S. Chandra, C. S. Ellis, and A. Vahdat. Differentiated multimedia web services using quality aware transcoding. In *INFOCOM - Nineteenth Annual Joint Conference Of The IEEE Computer And Communications Societies*, Tel Aviv, Israel, 2000.

[5] T. Leighton D. Karger, D. Lewin, and A. Sherman. Web caching and consistent hashing. In *Proc. of the 8th Intl. World Wide Web Conference*, May 1999.

[6] R. Doyle, J. Chase, S. Gadde, and A. Vahdat. The trickle-down effect: Web caching and server request distribution. In *Proc. 6th Intl. Workshop on Web Caching and Content Distribution*, June 2001.

[7] B. M. Duska, D. Marwood, and M. J. Feeley. The measured access characteristics of World-Wide-Web client proxy caches. In *Proc. of the 1997 USENIX Symp. on Internet Technology and Systems (USITS97)*, Dec. 1997.

[8] S. G. Dykes, C. L. Jeffery, and S. Das. Taxonomy and design analysis for distributed web caching. In *Proc. of the IEEE Hawaii Int'l. Conf. on System Sciences (HICSS'99)*, Maui, HI, Jan. 1999.

[9] S. G. Dykes, C. L. Jeffery, and K. A. Robbins. An empirical evaluation of client-side server selection algorithms. In *Proc. of IEEE Infocom 2000*, pages 1361–1370, Mar. 2000.

[10] S. G. Dykes and K. A. Robbins. A viability analysis of cooperative proxy caching. In *Proc. of IEEE Infocom 2001*, pages 1205–1214, Apr. 2001.

[11] S. G. Dykes, K. A. Robbins, and C. L. Jeffery. Uncacheable documents and cold starts in Web proxy cache simulations: How two wrongs appear right. Technical Report CS-2001-01, University of Texas at San Antonio, Division of Computer Science, San Antonio, TX 78249, Jan. 2001.

[12] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. In *ACM SIGCOMM '98*, pages 254–265, Vancover, Canada, 1998.

[13] S. Gadde, J. Chase, and M. Rabinovich. Web caching and content distribution: A view from the interior. In *Proc. 5th Intl. Web Caching and Content Delivery Workshop*, May 2000.

[14] S. D. Gribble and E. A. Brewer. System design issues for Internet middleware services: Deductions from a large client trace. In *Proc. of the First USENIX Symp. on Internet Technologies and Systems*, pages 207–218, Dec. 1997.

[15] M. Korupolu and M. Dahlin. Coordinated placement and replacement for large-scale distributed caches. In *Proc. IEEE Workshop on Internet Applications*, pages 62–71, July 1999.

[16] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[17] P. Krishnan and B. Sugla. Utility of co-operating Web proxy caches. *Computer Networks and ISDN Systems*, 30(1-7):105–203, Apr. 1998.

[18] J. Lee, H. Hwang, Y. Chin, H. Kim, and K. Chon. Report on the costs and benefits of cache hierarchy in Korea. In *Proc. of the 3rd International WWW Caching Workshop*, University of Manchester, June 1998.

[19] C. Maltzahn, Kathy J. Richardson, Dirk Grunwald, and James H. Martin. On bandwidth smoothing. In *Proc. of the 4th International Web Caching Workshop*, San Diego, CA, Mar. 1999.

[20] J. Mogul. Network behavior of a busy web server and its clients. Technical Report Research Report 95/5, Digital Equipment Corporation, Oct. 1995.

[21] A. Myers, P. Dinda, and H. Zhang. Performance characteristics of mirror servers on the Internet. In *Proc. of IEEE Infocom'99*, Mar. 1999.

[22] National Laboratory for Applied Network Research (NLANR). http://www.nlanr.net/.

[23] National Laboratory for Applied Network Research (NLANR). Ircache project. http://ircache.nlanr.net/.

[24] J. E. Pitkow. Summary of WWW characterizations. *Computer Networks and ISDN Systems*, 30(5):551–558, 1998.

[25] J. E. Pitkow. Summary of WWW characterizations. In *Proc. of the 7th WWW Conf.*, Brisband, Australia, Apr. 1998.

[26] M. Rabinovich, J. Chase, and S. Gadde. Not all hits are created equal: Cooperative proxy caching over a wide area network. In *Proc. of the 3rd Intl. WWW Caching Workshop*, June 1998.

[27] P. Rodriguez, C. Spanner, and E. W. Biersack. Analysis of Web caching architectures: Hierarchical and distributed caching. *IEEE/ACM Trans. on Networking*, 9(4):404–418, Aug. 2001.

[28] A. Rousskov and V. Soloviev. A performance study of the Squid proxy on HTTP/1.0. *World Wide Web*, 2(1-2):47–67, 1999.

[29] A. Rousskov and D. Wessels. Cache digests. *Computer Networks and ISDN Systems*, 30(22-23):2155–2168, Nov. 1998.

[30] R. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay. Design considerations for distributed caching on the Internet. In *Proc. of the Int'l. Conf. on Distributed Computing Systems (ICDS'99)*, 1999.

[31] K. Thompson, G. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, Nov.–Dec. 1997.

[32] Uppsala University. Uppsala University web-cache statistics. http://netstat.uu.se/Stat/Squid/.

[33] Squid Web Proxy Cache. http://www.squid-cache.org/.

[34] D. Wessels and K. Claffy. RFC 2186: Internet cache protocol (ICP), version 2. The Internet Engineering Taskforce, Sep. 1997.

[35] D. Wessels and K. Claffy. RFC 2187: Application of internet cache protocol (ICP), version 2. The Internet Engineering Taskforce, Sep. 1997.

[36] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy. Organization-based analysis of Web-object sharing and caching. In *Proc. of the 2nd USENIX Symp. on Internet Technologies and Systems (USITS'99)*, Oct. 1999.

[37] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. On the scale and performance of cooperative Web proxy caching. In *Proc. of the 17th ACM Symp. on Operating Systems Principles*, Dec. 1999.

[38] L. Zhang, S. Floyd, and V. Jacobson. Adaptive Web caching. In *Proc. of the 2nd Web Cache Workshop*, Boulder, CO, June 1997.

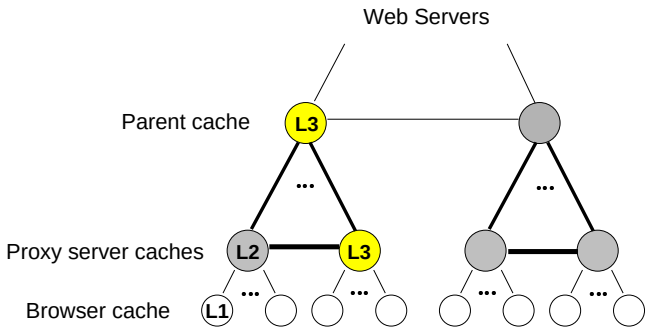

Figure 1: Proxy hierarchy. L3 caches are cooperating sibling or parent proxies.

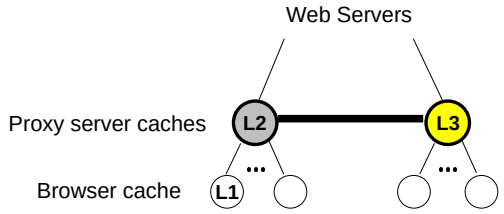

Figure 2: Proxy mesh. L3 caches are cooperating peer proxies.

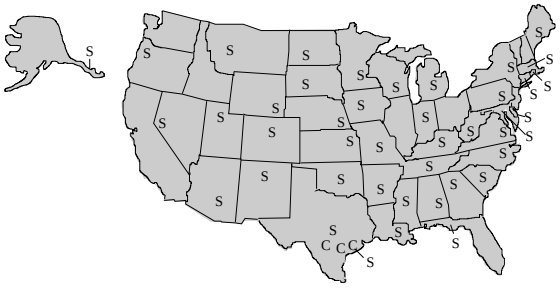

Figure 3: Location of servers (S) and clients (C).

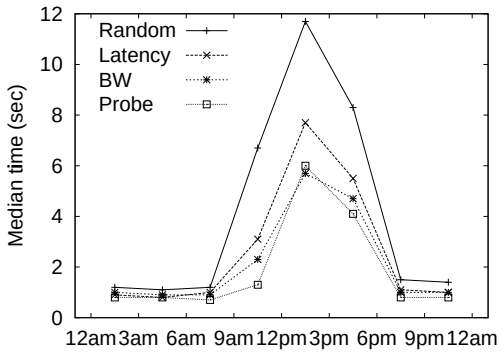

Figure 4: Time-of-day effect on the median response times at the UH client. Times are normalized to 50 KB.

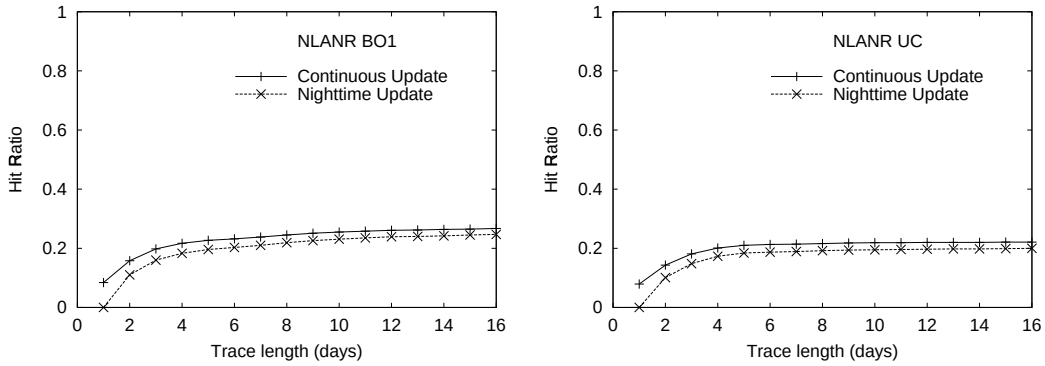

Figure 5: Efficiency of nighttime updates. Continuous Update assumes perfect directory information. Nighttime Update assumes directories updates occur only once per day (at midnight). Data are for NLANR caches BO1 (top) and UC (bottom), Jan 27 – Feb 11, 2000.

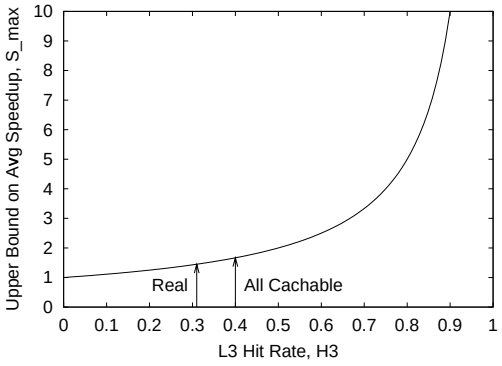

Figure 6: Upper bound on speedup in user response time as a function of the cooperation hit ratio. Real hit ratio reflects current cacheability constraints. All Cachable assumes all documents can be cached.

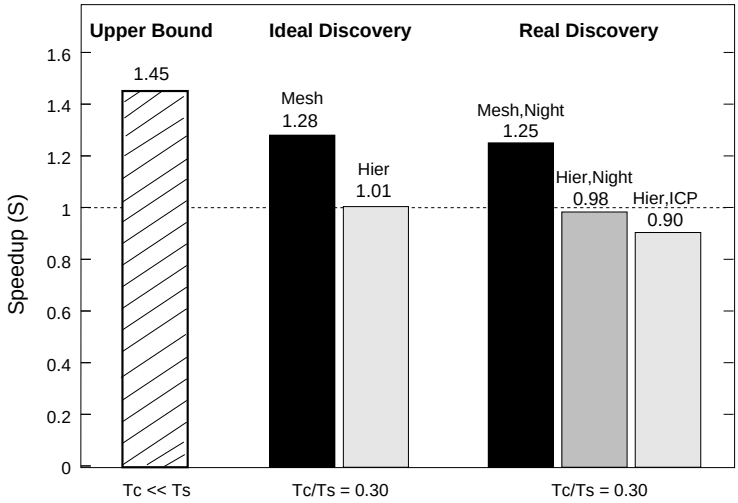

Figure 7: Speedups for proxy cooperation. The leftmost bar represents the upper bound for proxy cooperation. The dashed line at $S = 1$ indicates the boundary for a viable cooperation design.

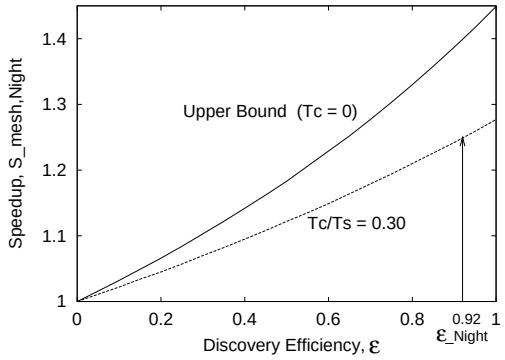

Figure 8: Speedup for a mesh using directories with only late night updates ($S_{mesh,\mathcal{E}}$). The arrow points to the $\mathcal{E}$ estimated from measurements described in Section 4.4.

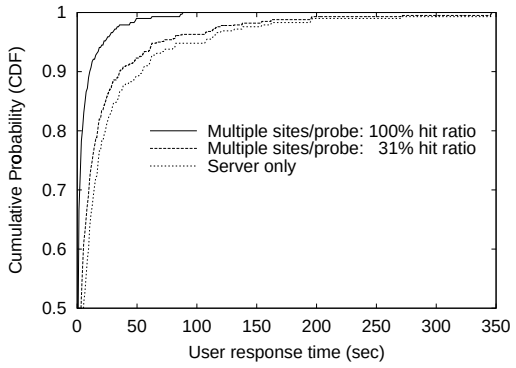

Figure 9: Cumulative distribution function (CDF) of user response times when clients use only the server site and when clients select from multiple sites using a network probe.