A Distributed Internet Cache

Dean Povey, John Harrison
School of Information Technology
University of Queensland
Brisbane, QLD, 4072
{*povey,harrison*}*@cs.uq.edu.au*

Abstract

This paper outlines the need for replication of resources on the Internet to combat the huge growth in user population and bandwidth requirements. It describes deficiencies in an existing hierarchical caching scheme for document replication and presents an alternative approach that more widely distributes the load across multiple servers, and provides additional resource discovery features. A simulation experiment which compares the distributed and hierarchical approaches is described and its results presented. These results indicate the applicability of the distributed caching scheme in the majority of situations considered.

Keywords Internet, World Wide Web, Caching, Harvest, Resource Discovery.

1 Introduction

The Internet is already large (approximately 12.8 million hosts at the last count [15]) and is growing at an exponential rate. The number of packets on the National Science Federation Network (NSFNET) alone grew by over 400% between 1988 and 1993[6]. A dramatic increase in the number of users, coupled with multimedia based information services, real-time audio/video transmissions and the emergence of network commerce are contributing to an incredible demand for bandwidth. Unfortunately, the resources available to service this demand are not expanding at a comparative rate. The amount of traffic on the network doubles every year, while the communications and router costs decrease at only 30% annually [6]. Subsequently, there is growing concern in the Internet community about the ability of the network to sustain future volumes of traffic. All these factors motivate the need for strategies to reduce network usage.

One approach which addresses these problems is the replication of popular information sources within an organisation or geographical region. This can greatly reduce the number of redundant fetches over heavily utilised links and can also help to alleviate the load on the servers at popular sites due to a reduced number of accesses. A study which examined traces of FTP transfers on the NSFNET, concluded that by placing multiple file caches at strategic points the volume of NSFNET traffic could be reduced by as much as 21% [4].

In section 2, existing schemes for document replication will be outlined. In section 3, a hierarchical caching strategy will be discussed. An alternative approach will be introduced in section 4, and compared with the hierarchical scheme. Finally, a simulation experiment will be outlined and its results presented.

2 Replication and Caching on the Internet

Replication is often used in distributed file systems [12] and databases [9] to improve the reliability and performance of data access. However, although there are similarities between these systems and information retrieval protocols such as the Hypertext transfer protocol (HTTP) which are used on the Internet, there are several differences which require us to develop different strategies.

1. *Information retrieved is almost invariably read only.* Because of this fact there is no need for the replication mechanism to manage consistency. No locking mechanisms are needed as only the maintainer of the primary copy may modify the document. This greatly simplifies the problem of replication.

2. *Support for legacy servers means there is a requirement for replication to be driven by intermediaries/middleware.* Some distributed systems strategies require the cooperation of the primary copy server to operate, however the proliferation of legacy applications makes this infeasible. HTTP is designed as a stateless protocol, therefore replication systems must be constructed which use special client software or a specialised intermediary.

Proceedings of the 20th Australasian Computer Science Conference, Sydney, Australia, February 5–7 1997.

3. *The Internet is very large.* Replication in distributed systems is typically designed for the smaller scale, resulting in inefficiencies and bottlenecks when applied to a very large and unstructured system such as the Internet.

For these reasons, different approaches are used to implement replication than are used in more structured distributed systems.

2.1 Mirroring

The periodic replication of a collection of files belonging to a particular Internet site or archive is often called *mirroring*. Mirroring is well suited to the replication of FTP archives, as they usually contain large and relatively static files which are frequently accessed by many users. Another advantage mirroring has, is that it allows a guaranteed level of service to be created and maintained. Consider for example, a multinational company which wishes to improve the response time of its information services in a certain country/region. It can do so by placing a server there which contains a replicated or 'mirrored' copy of its service.

The use of mirroring on the Internet is fairly widespread; yet despite this, users often ignore mirror sites in favour of the primary site. There are several possible reasons for this behaviour:

1. *Users don't know where the mirrors are.* This is a problem typical of many resources on the Internet. Locating information in such a large information system is often a frustrating and time consuming task.

2. *The mirrors are frequently incomplete.* Due to the large amount of data which may be stored at any one site, it is often infeasible for a mirror site administrator to maintain a full copy of an archive. This may discourage users from using the mirror site, as there may be a chance the file they are looking for is not there.

3. *The mirrors are sometimes out of date.* Frequently a user hears of a information resource such as a software program when it has only been recently released. At this time the mirror site of the archive which stores the file will not have had a chance to update its files.

Additionally, there is little information available to administrators as to which archives would be most profitably mirrored. Hence, the choice of what to replicate is largely based on personal bias, experience and response to requests. Many archives for which replication would provide significant benefits are either not mirrored or their mirrors are incomplete.

These problems highlight the lack of transparency in the mirroring strategy (both from the user's and administrator's perspective).

2.2 Caching

One alternative to mirroring files is caching. In this scheme, when a file is downloaded by a user it is stored locally so that subsequent requests may retrieve the cached copy. This form of replication is referred to as *Internet caching*. Internet caching is particularly suited to the World Wide Web where many of the objects are small and are modified frequently [14]. Caching does not have the problems associated with mirroring because:

1. *It is transparent to the user.* Because most schemes require no user intervention there is no need for the user to know the physical location of the replicated document.

2. *It is demand based.* The cache changes dynamically ensuring that files which are popular are cached. This relieves the responsibility of administrators to decide what to replicate and provides more flexibility for documents whose popularity is transient.

2.3 Existing caching schemes

World Wide Web Clients Most World Wide Web (WWW) clients provide a local cache. This is useful, as due to the hypertext nature of WWW browsers, users often skip backward and forward between documents. Additionally many authors of HTML documents will use the same inline image or background on multiple pages. If this image is cached then the redundant retrieval of these images will be reduced. However, client caches are not as efficient as they could be, as the cached documents cannot be shared with other users.

Shared caches using proxy servers Proxy servers were originally developed for use in Internet firewalls. A firewall is a security system which restricts access to the internal network of an organisation by only allowing incoming and outgoing packets to originate from a specific machine When a user wishes to download a document, they must direct their request to a proxy server which will download the file on their behalf and pass it through the firewall. One of the other advantages of proxy servers is that they can cache information retrieved to improve the performance of subsequent retrievals. A study of document caching indicated that the performance gain which is achievable by caching at the LAN level using a proxy server, is not that much higher than that obtained by using a client cache [2]. However, as a number of users can share the cache, this performance can be achieved using less resources.

3 The Hierarchical Cache

In their 1993 study of file transfer traffic on the NSFNET, Danzig, Hall and Schwartz showed that by providing caches in a hierarchical arrangement, the network bandwidth consumed for file transfer could be significantly reduced[4]. Following these findings, researchers at the University of Colorado and the University of Southern California developed a hierarchical caching strategy [3] as part of the Harvest resource discovery project[11]. The Harvest cache (and a later version named "Squid") is currently being used to provide caching for the World Wide Web in New Zealand [8], and as the infrastructure for a distributed information provision testbed developed by the National Laboratory for Applied Network Research (NLANR) [7].

The goals of the hierarchical cache are to distribute load away from server hot spots and to reduce access latency. It aims to do this by reducing redundant document transfers using a mechanism which enables the caches of multiple organisations to be shared in a hierarchical arrangement of servers. The server which is at the upper most level of the hierarchy is termed the *root* cache, while those nodes at the lowest levels of the hierarchy are termed *leaf* caches. In addition, a node which is at the immediate level above a given cache is known as the *parent* of that cache. Caches which share the same parent are known as *siblings* and the node which holds the original copy of the document is called the *primary site*. A diagram illustrating the arrangement of a hierarchical cache is given in figure 1.

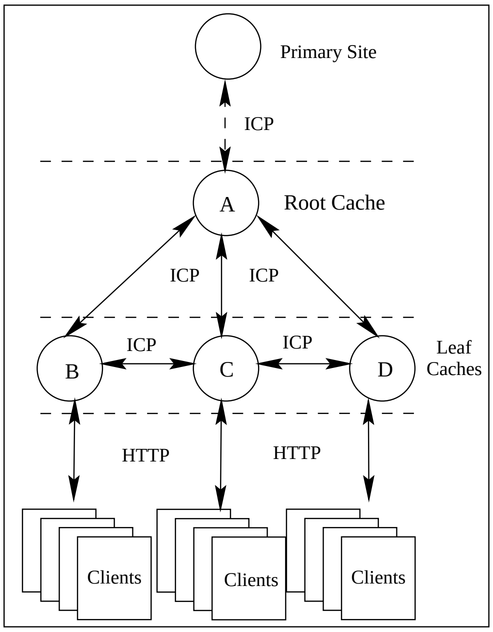

Figure 1: The Hierarchical Cache

In the hierarchical approach, WWW clients are configured to request a document from a leaf cache using a standard HTTP proxy protocol. If the request cannot be resolved from its local set of documents, the cache then queries its parents and siblings using a datagram oriented protocol known as the Internet Cache Protocol (ICP)[13]. The cache also sends a hit message to the UDP 'echo' port of the primary site. As the 'echo' port of a host will echo any message sent to it, this provides a mechanism of simulating a hit at the primary site. Thus, in the case that this site responds before replies are received from the parent or sibling caches, it may be retrieved directly from that site.

A cache retrieves the object from the first sibling, parent or primary site which responds with a 'Hit'. In the case that all the caches queried have replied with a 'Miss', the parent cache is contacted (either using the HTTP proxy protocol, or a connection oriented component of ICP) and the process is repeated. This procedure continues recursively with each higher level of cache querying its parent and siblings until the request is satisfied, or the root node is reached. In this last case the document is retrieved from the primary site.

3.1 Advantages

The arrangement of caches into a hierarchical system can reduce wide-area network bandwidth demand by reducing the redundant transfer of information at successive levels in the network. This is an improvement over a single caching proxy as it reduces the level of traffic at the organisation's gateway, as well as all gateways up to the root node of the hierarchy. The implementation can also help to further reduce the load on server 'hot spots' in the network by decreasing the number of accesses to such nodes.

The root node is responsible for processing all requests which are not resolved by the other caches in the network, therefore it can experience high load when the number of requests from these other caches is large. In these situations sibling caches are used to provide load balancing.

Consider the example in figure 2. In this example it is assumed that the root node A is under heavy load. Suppose node B wishes to obtain an object n, and neither node A or C has a copy. Node B will simultaneously send a query request to node A and C, as well as a resolution packet to the echo port of the objects primary site (step 1). In the usual scenario, when replies had returned from nodes A and C, node B would connect to A and resolve the request from the primary site via proxy. If node A is under heavy load, as in our example, the resolution packet may return from the primary site of n before A is able to process and respond to the original request from B (step 2). B would then connect to n's primary site and retrieve the object (step 3). Should node C now request the same object, it will be able to resolve the request from cache B, thus bypassing the root node (step 4).

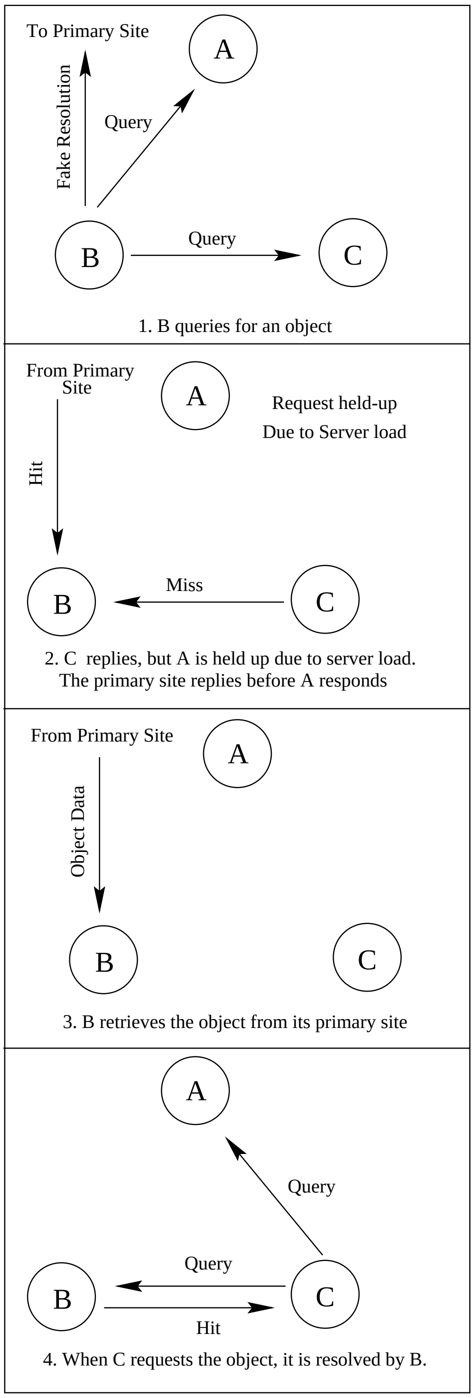

Figure 2: Example: Bypassing the root node under heavy server load

When A is heavily loaded, the other nodes are able to continue functioning without performance loss by bypassing the root node. The cache at the root node A will be a subset of the caches at B and C, so the hit rate will be similar to what it would be if the request had been resolved by the root node. It must be noted that there will then be an extra hop involved in transferring the object over the network.

This mechanism does not completely solve the problem of congestion at upper level nodes. Requests which the root node is able to process will still incur significant delays. In addition many primary sites may not provide a UDP echo service, or may disallow it as they are behind a firewall. Finally, this feature may be disabled in later versions of the Harvest and Squid caches and is turned off in the default configuration. Regardless of these problems however, the load balancing mechanism described above does improve the scalability of the hierarchical approach.

3.2 Disadvantages

Despite the benefits outlined above, there are several problems associated with hierarchical caching.

Disk space required at upper level nodes In a study of application-level document caching for the Internet at Boston University, the increase in cache size as a function of the number of users was examined for both local and remote documents in a cache[2]. A *Cache Expansion Index* (CEI) was computed which indicates the ratio of the cache size required to maintain a given byte hit rate as the number of users increases. The byte hit rate is the ratio of the size of documents served from the cache to the size of documents which were not in the cache. It was found for both local and remote documents that the CEI increased linearly[1] as a function of the number of users.

For a node to maintain an optimal hit rate, and provide the maximum reduction in redundant transfers, its local cache should contain the intersection of the contents of all its child nodes. For the root node and other upper level caches, the storage space required to hold this cache is thus likely to be quite large. Given the relationship identified above, as the number of users of the hierarchical cache increases, the upper level nodes will have to increase their storage capacity proportionally to maintain efficiency.

However, the number of users on the Internet is increasing exponentially. Therefore the size of the cache must also grow exponentially to maintain a constant byte hit rate. In addition, the size of files is also growing at a high rate. A study of the characteristics of wide area TCP/IP communications found that size of FTP files increased an order of magnitude from 1989 to 1991[5]. The increasing use of multimedia applications which have large storage requirements indicates that file sizes are likely to continue grow at a high rate, thus further decreasing the available storage capacity of nodes.

Although the cost/byte ratio of storage space is also decreasing exponentially, it is unlikely that it will improve at the same rate by which the demand

[1]The constant for local documents was quite small at around 0.03 (Meaning a 3% increase in cache size is needed for one additional user) The constant for remote documents was much higher at 0.12[2].

for it is growing. As the requirements for storage space increases, the proportional size of caches at nodes will be forced to decrease, resulting in lower hit rates. Thus, as the number of users and file sizes increase, the efficiency of the hierarchical approach will degrade.

Management of information about siblings The hierarchical cache attempts to cater for the high load on upper level servers by allowing caches to bypass these nodes using sibling caches. This mechanism is unwieldy and creates management problems because each node must maintain location information about all its siblings to enable queries to be directed to them. While it is trivial to maintain such information when the hierarchy is controlled by a single organisation, the task becomes increasingly complex as the responsibility for maintenance is disseminated throughout various organisations within a caching network. The dynamic nature of the Internet would make it difficult for cache maintainers to keep up with the cache server additions and deletions. This problem can be compared to the one that the Domain Naming System (DNS) was created to solve in the early 1980s. Originally name to address mappings of Internet hosts were stored in a file on each machine which required systematic updating as hosts were added or addresses changed. As the Internet grew these host files quickly became inconsistent due to the frequency of updates required. Thus, the DNS was created to manage the naming of hosts by using a hierarchical structure [1]. Like the host file, sibling information is stored statically in the hierarchical cache, resulting in consistency problems as the frequency of updates grows. Although a system could be conceived to propagate this information between nodes, this problem was not addressed by the designers of hierarchical cache.

Additionally, as the number of siblings in a cache topology grows, so do the number of query messages being sent. This is particularly problematic at the level immediately below the root where there are likely to be a large number of caches (potentially one for each second level domain in the region). This both increases the load on the network and the complexity of the system, as caches must cope with managing reliability for multiple messages sent over an unreliable transport system.

Given these problems, it would seem that the use of sibling caches at upper levels in the hierarchy may be inappropriate. However, this is where this strategy is most needed, as the main candidates for bottlenecks due to congestion will be the root and upper level nodes. This particularly true in the case of root caches which tend to have low hit rates compared to other caches and thus spend more time retrieving objects from the primary site.

Tariff Based Volume Charging One of the methods that can be used by providers of Internet services to encourage more responsible use of bandwidth, is volume charging on connections and imposing higher tariffs on heavily used links (e.g. the international links between Australia and the U.S.). A tariff-based scheme provides an incentive for users to make better use of domestic resources (such as caches and FTP mirrors).

Unfortunately, the use of a hierarchical cache limits the effectiveness of applying such a scheme. This is because caches closer to the root node act as proxy agents for lower level users. It is thus the maintainers of upper level caches that are charged the higher tariffs while lower level users get the benefits of international access at domestic rates. This situation is good for consumers (and would certainly encourage them to use the cache) but represents a significant deterrent to providers of the service. The extra costs incurred must either be absorbed by the provider or recouped by maintaining accounting procedures to recover them from end users. A level of security must also be maintained to prevent malicious users from using the cache to circumvent the charging scheme.

Differences in levels of the Hierarchy Experience with the hierarchical cache has shown that the hit rates in root caches are less than those at lower levels [7]. This might be expected due to the reduced commonality in the sharing of documents by a broader range of users. Researchers on the NLANR project suggest that there may be a need to provide different functionality at different levels of the cache. In addition, this lower hit rate will tend to increase the load on the root cache, as they it have to service more requests from external sources.

3.3 Summary

Despite its benefits, the hierarchical cache has disadvantages related both to the management of information about siblings and to scability problems related to the amounts of disk space required at upper level caches. In addition, the design of the hierarchical cache does not lend itself to applying traffic based tariff schemes, thus encouraging a more responsible use of bandwidth. Finally, the lower hit rates at root caches suggest that these may require differing cache policies and strategies to improve performance. In the next section some improvements to the hierarchical cache are suggested which address these issues.

4 A Distributed Cache

4.1 Improving upon the hierarchical cache

One approach which addresses the above issues, is to avoid the use of servers at the root and upper levels of the hierarchy to resolve requests and store cached documents. Instead, only the leaf caches would be responsible for retrieving objects, while the upper level caches would be used to maintain information about the contents of these caches. This scheme removes the requirement for upper level nodes to maintain large storage spaces, yet exploits the hierarchy for quickly locating replicated documents. While the hierarchy is still present in this scheme, it becomes more appropriate to call this a *distributed* cache because it dispenses with successive levels of document caching.

Figure 3 illustrates how such an approach might work for a hierarchy of three levels. In this case, neither nodes A, B or C are caching objects. They are instead used only to propagate cache information. For example, if node F requires an object, it queries node B (step 1). Here node B does not know where a copy of the object can be found. However, instead of returning a 'miss' immediately to F, it propagates the query to A (step 2). Node A is also unable to locate the object returns a miss to B which then propagates this information back to F. This recursive querying approach allows the entire cache contents to be searched very quickly so that node F can correctly determine that there is no cached copy of the requested object and thus retrieve it from its primary site (step 3).

When a miss is encountered by a leaf cache and it resolves the request from its primary site, a mechanism is required to indicate to the upper level nodes in the hierarchy that the document has been cached. This procedure is known as an 'advertisement' and is illustrated in figure 4 (part 4). Node F sends an advertisement message to its parent (B). The advertisement is then propagated recursively to the root node. If a child of node B then requests the object then B will have information about where to find it. If a child of node C requests an object then by recursively querying, the root node A will indicate that a copy may be found at F (step 5).

Rather than returning the actual object from its cache, the upper level server will return a 'pointer' to the cached object consisting of a Uniform Resource Locator (URL) and information about the size and modification time of the object. In the example, when E retrieves the object it may also advertise a pointer to C and A (step 6). A similar advertisement scheme can be used when the object at nodes at E or F is removed from the cache. In this case, a message would be advertised indicating

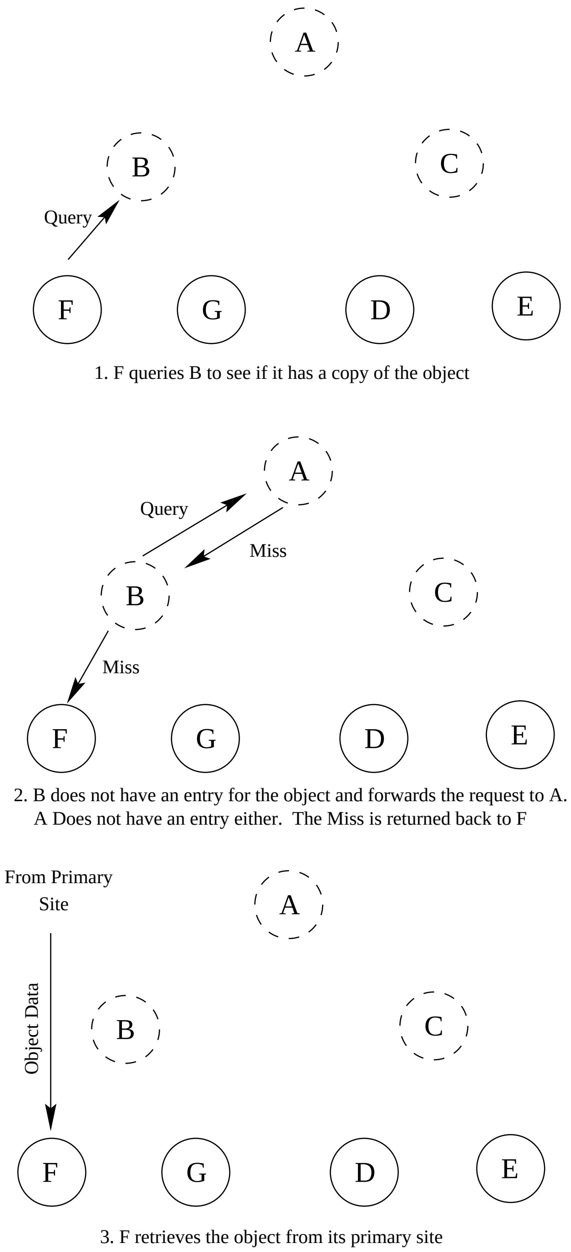

Figure 3: Example: Retrieving an object from a Distributed Cache

to the upper level caches that the pointer should be deleted.

When advertising an object, a time-to-live parameter is included for each object enabling upper level caches to expire the object after a given period without receiving a delete advertisement. This is important as delete advertisements will be delivered using an unreliable transport mechanism. When an object expires, an upper level cache can choose to either remove it from its pointer table, or query the lower-level node which advertised the pointer to see if the object is still cached and obtain a new expiry time.

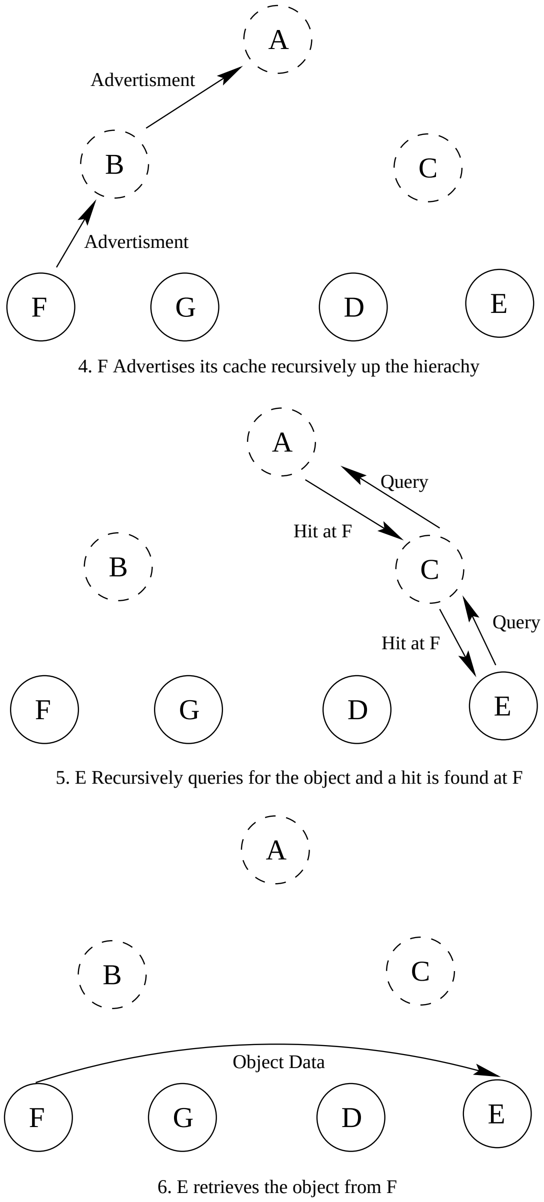

Figure 4: Example: Using advertisements in a Distributed Cache

4.2 Advantages

The goal of the distributed cache is to allow the rapid location of replicated documents at each of the lower-level caches while avoiding the requirement that caches in the upper levels of the hierarchy store them. As upper-level nodes maintain pointers to the documents rather than copies of them, the storage space required at these nodes will not be proportional to the size of the documents cached and the space required for each document will be dramatically smaller.

The distributed cache does not need to query its siblings to determine whether they cache objects, thus avoiding the management problems related to the maintenance of sibling information. Each parent 'learns' about the contents of its child caches by receiving advertisements. This allows a cache to query the contents of its siblings by sending a single query to its parent.

The latency for misses should also be reduced. The hierarchical scheme required each cache to query its parent and siblings and then connect to its parent to resolve the request. If a miss was found then the document would need to then be passed back down through each successive level of cache. In the distributed cache, a single query is propagated up the hierarchy before the document is retrieved by the leaf cache from the primary site.

Furthermore, providers of upper level caches in the distributed scheme are not disadvantaged because of volume based tariff charging schemes, as it is only the leaf nodes in the hierarchy that are responsible for resolving requests from the primary site.

An additional advantage is that the advertisement scheme could be extended to allow the system to integrate the propagation of information about mirror sites. As indicated in section 2.1, mirroring allows an organisation to replicate resources to maintain a guaranteed level of service, a benefit which is not realised using the hierarchical or other caching mechanisms. By extending the advertisment scheme in this way, users of the cache would benefit from the large number of mirror sites already operating for FTP and WWW. It also facilitates the replication of services such as index searches and databases (e.g. the Lycos Search engine and the Internet Movie Database). This was previously difficult, as the hierarchical scheme only allowed *documents* to be replicated. In the Squid cache, mirror sites may be statically incorporated into the system, however this approach lacks transparency and requires that cache administers maintain a database of mirror sites which may need to be updated on a regular basis. The advertisement system provides a way for the administrators of the mirror sites to add and remove themselves without the intervention of the cache administrators.

It is also feasible that the advertisement scheme could be further extended to propagate more general resource discovery information (e.g. to provide redirection for pages when WWW sites move).

4.3 Disadvantages

One of the main disadvantages of the distributed caching method (when compared to hierarchical caching) is that it increases the load on leaf servers. As the number of requests in the network increases, the leaf caches become responsible for servicing many requests from other caches in the network as well as their own clients. If the number of requests from both the clients and other caches is very large then this may become a performance bottleneck. In this case there are a number of different options which could be pursued to reduce this bottleneck. These include:

- restricting the propagation of advertisements so they only travel up the hierarchy one or two levels rather than to the root node. This would reduce the number of clients the leaf cache would have to service.
- allowing a leaf cache to reject requests for retrieval from other caches when the load on the cache is above a certain threshold.
- allowing upper level nodes to maintain small caches, but retaining the pointer system to allow the cache to still remain distributed.

These options may be statically configured into each cache or alternatively a system could be conceived where the cache is able to dynamically alter its configuration in response to changing system events.

Retrieving cached documents from the leaf servers also causes redundant information to be retrieved over some links. Arranging the cache in a hierarchy aims to ensure that a document isn't repeatedly retrieved *over the same network link*. In the distributed cache this is reduced to ensuring that a document isn't repeatedly retrieved *from the primary site by nodes within a region*. This difference means that bandwidth will be less efficiently utilised within a region as more traffic occurs within the regional network to retrieve cached objects.

Despite its disadvantages, if the distributed strategy is able to maintain similar or better throughput than the hierarchical cache, then its improved features make it a potential candidate to replace the hierarchical caching strategy. In the next section the results of a simulation study are outlined, which compare the performance of the hierarchical and distributed caching strategies to determine if the distributed cache meets this criteria.

Topology	Root	Lvl 2	Lvl 3	Lvl 4
A	1	16	8	4
B	1	8	8	8
C	1	4	8	16

Table 1: Number of siblings at each level in each topology

5 Simulation Experiment

To compare the hierarchical and distributed caches, a study was undertaken which evaluated how different request loads and cache topologies affect the performance of each strategy. The purpose of this study was to determine if the same or better level of performance can be obtained using the distributed cache as opposed to the hierarchical cache under the same conditions (performance being measured by the average transfer rate of documents in Mb/s).

The simulation assumed that the majority of bottlenecks were likely to be in the I/O overhead required to process and fulfil requests, so load was simulated by modelling this behaviour. In addition, the simulations looked at how the number of requests affected performance by testing a number of mean interarrival times (the average spacing between requests arriving at the leaf nodes). Three different network topologies were used to determine if different arrangements of nodes would affect the behaviour of the caching strategies. Each of the three topologies simulated consisted of four levels including a single root node and a total of 512 clients at the leaves. The number of siblings at each level in each topology is listed in table 1.

Other possible parameters were held at constant values. The values used were chosen to represent reasonable values in caches and networks. The settings chosen were: bandwidth of storage devices – 1.5Mb/s, bandwidth of regional network links – 2.0Mb/s, Bandwidth of inter-region/primary link – 15Mb/s and cache hit rate – 30%. Trace data used in simulations of both strategies was gathered over several months from a proxy cache existing at the Department of Psychology at the University of Queensland.

5.1 Results

A three way analysis of variance (ANOVA) was performed between 2 levels caching strategy, 3 levels of topology and 10 levels of mean interarrival time as variables in a $2 \times 3 \times 10$ between subjects design. Tests for significance were performed using an α level of .01 because of failure to meet sampling assumptions.

Analyses indicated that the three way interaction between mean interarrival time, topology and caching strategy was significant ($\mathcal{F}_{18,30660} = 1.941$,

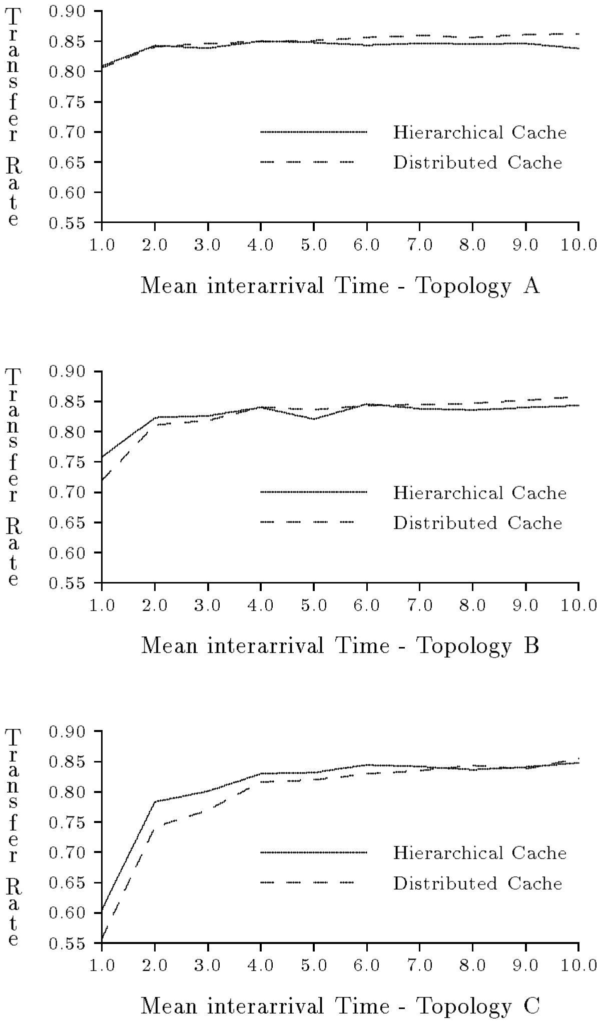

Figure 5: Mean interarrival time of requests vs Average transfer rate for each of the topologies

$p < .01$). This interaction is plotted for each topology in figure 5.

The presence of a significant three way interaction indicates that conclusions about the performance of the hierarchical vs distributed caching strategies cannot be made without considering both the topology used and the value of the mean interarrival time.

Further analysis for topology A, showed that the effect of the number of requests was not significant, and the distributed cache performed better overall than the hierarchical cache. However, for topologies B and C, the effect of mean interarrival time was significant when it had decreased below 2.0 for topology B and 3.0 for topology C. Both these results show that for topologies B and C, the hierarchical cache was able to maintain a higher average transfer rate when the number of requests was increased.

These results may be explained by considering that as the number of requests increased for topologies B and C, the leaf caches in the distributed simulation were under higher load than those in the hierarchical cache in the same situation. This is because in the distributed cache, the leaves had to not only serve a large number of requests from their clients, but an increasing number of requests from other caches. This load would have been higher than that experienced by topology A as there are many more leaf caches in topology A[2], but in all topologies there are the same number of clients.

5.2 Conclusions

In most of the simulation runs performed there is no significant difference with regard to the caching topology used. The Distributed caching strategy is thus favoured in these situations, as it addresses other deficiencies in the hierarchical scheme.

However, the results of the simulations indicate that as the number of requests in the network increase, a distributed cache node has to respond to too many requests from other caches and its ability to service requests for its clients is compromised. This is much in line with what was predicted in section 4.3 and future work may determine whether the measures outlined there will improve performance.

It should also be noted that the simulations underestimate the load placed on the root server as experience suggests that this machine will maintain a lower hit rate than other caches. It may be that a lower hit rate for the root cache will favour the distributed cache.

6 Summary

There has been dramatic increase in the installed user base of the Internet due in part to the popularity of the World Wide Web. Therefore, mechanisms need to be put in place to reduce the increasing amount of redundant information which flows through the Internet. One strategy which attempts to do this places caches at strategic points in the network using a hierarchical structure. By analysing the behaviour of the hierarchical cache, several key weak spots were identified in the scheme.

1. The large caches needed to provide appropriate hit rates at the root nodes require an exponentially increasing amount of storage space to grow with the volume of traffic on the Internet.

2. The use of sibling caches to relieve server load introduces significant management problems in keeping dynamic information about siblings up to date.

3. A hierarchical cache does not promote the use of volume based tariff schemes to promote more

[2]Topology A has 128 leaf caches, while B and C have 64 and 32 respectively.

responsible use of network bandwidth on heavily used links.

The distributed approach to caching introduced in this paper addresses these problems and incorporates additional resource discovery features. A simulation study indicates that this strategy performs well for most network topologies. However, for topologies where the number of servers at upper levels are low, the performance of the distributed cache is less than the hierarchical one when the number of requests is high. The increased load on leaf caches appears to reduce their ability to serve requests from both their own clients and other caches. Several strategies have been outlined which could be incorporated into the distributed cache to overcome this problem. In the majority of situations examined there is no significant performance difference between the caching strategies used.

A prototype implementation of the distributed cache has been completed as well as a draft protocol document outlining the caching protocol used [10].

6.1 Future Work

There are still some outstanding issues which need to be resolved before the implementation of the distributed cache can be completed. These include:

1. Further simulations to determine whether the problems identified with the distributed strategy can be overcome using some of the methods outlined in section 4.3 must be undertaken. The effect of I/O transfer rate, network, bandwidth and hit rate on the strategies should also be investigated.

2. Studies need to be done to determine the most efficient way of propagating advertisements, including expiration and maintenance of consistency.

3. The integration of mirror sites should be investigated.

References

[1] Paul Albitz and Cricket Liu. *DNS and BIND in a nutshell.* O'Reilly and Associates, 1992.

[2] Azer Bestavros, Robert L. Carter, Carlos R. Cunha Mark E. Crovella, Abdelsalam Heddaya and Sulaiman A. Mirdad. Application-level document caching in the Internet. In *Proceedings of the Second IEEE International Workshop on Services in Distributed and Networked Environments*, pages 166–173, June 1995.

[3] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz and Kurt J. Worrell. A Hierachical Internet Object Cache. Technical report, Department of Computer Science, University of Southern California and Department of Computer Science, Universityi of Colorado, Boulder, 1995.

[4] Peter B. Danzig, Richard S. Hall and Michael F. Schwartz. A case for caching file objects inside Internetworks. Technical report, Department of Computer Science, University of Colorado, Boulder, 1993.

[5] Peter B. Danzig, Sugih Jamin, Ramon Caceres and Danny Mitzel. Characteristics of wide-area TCP/IP conversations, September 1991. ACM SIGCOMM 91 Conference.

[6] Jeffrey K. MacKie-Mason and Hal R. Varian. Some economics of the Internet. In *10th Michigan Public Utility Conference at Western Michigan University*, page 37, November 1992. revised version: February 17, 1994.

[7] National Laboratory for Applied Network Research. A distributed testbed for national information provisioning. Available from http://www.nlanr.net/Cache/, 1996.

[8] Donald Neal. The Harvest object cache in New Zealand. Available from http://-www.waikato.ac.nz/harvest/www5/-Overview.html, 1996.

[9] M. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice-Hall, 1991.

[10] Dean Povey. Internet Cache Protocol-NG (proposed specification). Available from http://www.psy.uq.edu.au/˜dean/-project/icp.html, 1996.

[11] Michael F. Schwartz. Internet resource discovery at the University of Colorado. *IEEE Computer*, Volume 26, Number 9, pages 25–35, September 1993.

[12] Andrew S. Tanenbaum. *Modern Operating Systems.* Prentice-Hall, 1992.

[13] D. Wessels and K. Kaffy. Internet Cache Protocol (ICP), version 2, November 1996. IETF Internet-Draft.

[14] Duane Wessels. Intelligent caching for World-Wide Web objects. Master's thesis, Washington State University, 1995.

[15] Network Wizards. Internet Domain Survey. Available from http://www.nw.com/, 1996.