Caching Neighborhood Protocol: a Foundation for Building Dynamic Caching Hierarchies with WWW Proxy Servers

Cho-Yu Chiang   Ming T. Liu   Mervin E. Muller

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210-1277

E-mail: {chiang, liu, muller-m}@cis.ohio-state.edu

February 5th, 1999

Abstract

In this paper, we propose a *Caching Neighborhood Protocol* (*CNP*) that describes an infrastructure upon which proxy servers can build dynamic caching hierarchies. Compared to the conventional schemes based on static caching hierarchies where proxy servers can only communicate with each other following predefined path configurations, the proposed dynamic scheme permits flexible communications among proxy servers. Such a scheme can decrease the response times of requests by increasing the availability of documents without compromising the currency of rendered documents. *CNP* is based on the concept of *caching neighborhood*, in which originating servers extend their spans and availability through their *caching representatives*, incarnated as proxy servers. The coherence of documents provided by the originating server and the cached copies on its caching representatives is maintained via employing the *X tolerance coherence requirement*. Some qualitative reasonings and studies on the traces of request accessing logs collected by a proxy server at NLANR [16] are presented to justify the proposed *CNP*. One of the most important observations is that caching the most popular documents provided by the most popular Web sites would be the most effective way for Web caching. A five day trace analysis shows that the top 20% Web sites (in terms of frequency of accesses) provide 93% of the most frequently accessed top 20% documents and such top 20% documents account for 94% documents that have been accessed more than once. We also describe how *CNP* exploits these access characteristics to fare against the static caching hierarchy schemes based on *Internet Cache Protocol* (*ICP*) [7], such as *Harvest* [2] and *Squid* [21].

Keywords: WWW, proxy server, originating server, Caching Neighborhood Protocol (CNP), caching representative, Web caching, X tolerance coherence requirement.

1. Introduction

Web caching [14] has been generally recognized as an effective approach to deal with problems such as slow response times and network congestions resulting from the overwhelming popularity of the Web [5] in recent years. The basic idea is to have some intermediate servers, acting as proxies [4,12,20] for the Web site servers, process requests in between the end users and the Web site servers that provide the requested documents. These proxies, denoted by *proxy servers* in this paper, maintain individual cache storages for documents that have been downloaded in the past and provide the cached documents if they are requested afterwards. Essentially after a proxy server intercepts requests made by the end users, if it cannot find copies of the requested documents in its cache, it will forward the requests to the Web site servers providing the documents. The Web site servers, denoted by *originating servers* to reflect their identities in providing original copies of documents, will process the requests and return appropriate responses. The responses then will be cached at the proxy servers and returned to the requesting users. Although the Web supports many Internet protocols such as FTP [18] and Gopher [19], only caching of HTTP [17] responses is our concern in this paper.

The conventional caching hierarchy schemes, such as Harvest, Squid and their variants [2,21,23,24], are the forerunners in the effort of decreasing loads on the originating servers and reducing the request response time. With the introduction of Internet Cache Protocol (ICP) [7], communication paths among a fixed set of proxy servers are defined. Such kind of caching hierarchy schemes are considered "static" in the sense that they lack flexibility in the communication paths among proxy servers. All unresolved requests will follow the same path in the caching hierarchy before being forwarded to the originating servers. According to ICP, proxy servers are configured in reference to their physical locations in the network and form a family-tree like relationship. The relationships among the proxy servers are classfied as parent-child and siblings. The relationship of parent-child is derived from the positions of the poxy servers in the actual network hierarchy, while the relationship of siblings come from the reality that the proxy servers share the same parent. A child will forward an unresolved request to its siblings and parent simultaneously.

If the child does not receive the needed document within a period of time, the unresolved request will be forwarded to its parent on a higher layer of the hierarchy. The parent will process the requests in the same fashion, maybe again forwards the request to its parent and so on until a request is serviced or has reached the top of the hierarchy. In such circumstances, the request has to be processed by the originating server and the returned response is passed down the hierarchy and saved in the caches of proxy servers that had forwarded the request.

In this paper, we propose a *dynamic* caching hierarchy scheme based on the proposed Caching Neighborhood Protocol (CNP), which incorporates some ideas from [8,10,11,15,22] along with observations from proxy traces to create a promising new approach. Such an approach is considered *dynamic* because the set of proxy servers that collaboratively handle the requests may change for every single request and thus the request handling paths will vary. The Caching Neighborhood Protocol introduces and employs the idea of *caching representatives*, which essentially are partially replicate servers [13] and mainly devoted to distributing loads for frequently accessed originating servers. The goals of proposing CNP are to deal with the cache coherence problem [3,6,9] existing in the static caching hierarchy schemes, increase the document availability while not compromising the currency, and reduce the response times of document requests. Some insights were obtained from examining the traces of request accessing logs collected by a proxy server at NLANR [16], which points out a few statistical characteristics of collective user accessing patterns that are transformed into the foundation of CNP.

The rest of the paper is organized as follows. Section 2 describes the fundamentals of the caching neighborhood approach. Section 3 describes the Caching Neighborhood Protocol in detail. Section 4 provides analysis based on traces of request accessing logs to argue why the static caching hierarchy schemes are inferior to the proposed dynamic caching scheme. Section 5 is dedicated to the discussion on the potential about using the proposed dynamic caching hierarchies approach and possible improvements. Section 6 overviews the directions of future research works before we conclude the paper in section 7.

2. Fundamentals

Our approach aims at building dynamic caching hierarchies in which proxy servers and originating servers cooperate on a distributed basis. The major advantage of employing dynamic caching hierarchy infrastructure is to retain proven merits of using hierarchies of proxy servers while alleviating or eliminating problems and drawbacks that are manifest in the static caching hierarchy schemes. Generally speaking, the common problems of using static caching hierarchy that are susceptible to criticism are as follows.

1. Static caching hierarchy schemes are inflexible in dealing with changes of network traffic volume and network hardware configurations. If a request causes a series of cache misses as being forwarded up in the static caching hierarchy, the request may have to travel through a traffic-saturated area to reach the originating server and thus the response time is prolonged. Besides, since all the proxy servers have to be software configured so that they can be aware of their positions in the hierarchy, a proxy server can only exchange messages with other proxy servers following the pre-defined configuration setup. The consequence is, a newly inserted proxy server or a proxy server out-of-service indefinitely requires adjustment of proxy configuration files on many proxy servers in the same hierarchy.

2. It is not cost-effective to maintain coherence of documents in the static caching hierarchy schemes. There would be considerable duplicated documents stored throughout the static caching hierarchies of proxy servers, especially popular documents. Not only do the duplicated copies waste storage space provided that cache storage space of all proxy servers is regarded collectively as a whole, but also it is a challenging problem in maintaining coherence among duplicated documents stored in the caches of proxy servers and original copies on the originating servers. In a static caching hierarchy scheme, the cache coherence problem can be compromised by fetching possibly outdated documents. It could also be handled at the expense of frequent communications among proxy servers and originating servers in verifying the currency of cached documents on the proxy servers.

3. In the static caching hierarchy schemes, excess interogation delay could be introduced due to cache misses as the processing delays accumulated on a chain of proxy servers in the hierarchy. That is because requests may have to travel up to the top and corresponding responses may have to travel all the way down the hierarchy. Consequently, static caching hierarchy approaches will affect the scalability because a hierarchy should not have too many layers. The queueing and processing delays could add up and affect the response time of a request. Such an issue cannot be neglected because a good percentage of documents are requested for only once in a period of several days. If such documents are cached, quite possibly they will be purged before their next requests are due because the amount of cache storage space is not unlimited. As shown later in Figure 3 and Figure 4, 75%-82% of documents receive only single request and collectively account for 33%-46% of total accesses, depending on the lengths of the observation periods. Altogether, the problems that stem from the static caching hierarchies justify exploring the possibility of deploying dynamic caching hierarchies to overcome these shortcomings.

2.1 Caching Neighborhood

The notion of caching neighborhood unveils the fundamentals of our scheme upon which the dynamic caching hierarchies can be built. Basically, a *caching neighborhood* is a logical relationship that (i) consists of an originating server and a variable number of proxy servers acting as caching representatives for the originating server; (ii) defines an information disseminating partnership such that the data objects provided by the originating server can be distributed by any proxy server in the neighborhood with authority. By authority, we mean that any document fetched by a representative in a particular caching neighborhood can be regarded as being retrieved directly from the originating server of that neighborhood. Note that the concept of caching neighborhood does not impose any constraint on physical proximity among an originating server and its caching representatives. Instead, it is a cooperating partnership which increases the availability of documents as well as a scheme flexible to the changes of network configurations and network traffic volume. Each originating server can build its own neighborhood with exclusively proxy servers functioning as caching representatives. A caching representative would assume the responsibility of resolving a document request if a request it received is for an originating server it represents.

Figure 1 is used to introduce the basic concept of caching neighborhood. There are two originating servers denoted by *F* and *G* in the figure. Each originating server holds its own caching neighborhood marked by solid connecting lines or dotted connecting lines. The lines can be regarded as specifying HTTP persistent connections [17] among the originating servers and their caching representatives. In addition, a line connecting a user to a proxy server only indicates that the proxy server will handle all the requests for that user. The connection between the user and the proxy server does not have to be HTTP persistent.

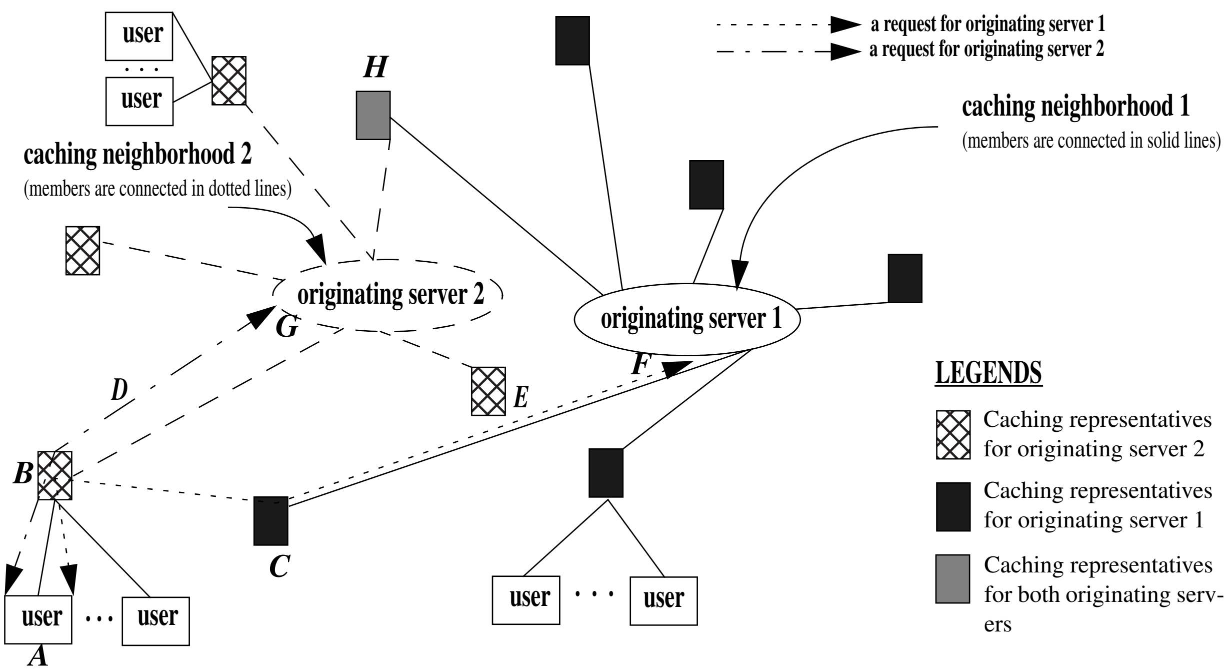

Figure 1. A generic scenario consisting of two caching neighborhoods

Figure 1 is presented to display the following properties of caching neighborhoods:

1. The members of each caching neighborhood, in addition to the sole originating server being represented, are all proxy servers.
2. A proxy server may represent multiple originating servers; for example, the proxy server *H* represents both originating servers in the figure. Therefore, caching neighborhoods can overlap each other.

3. The sizes of caching neighborhoods may vary -- in terms of both number of representatives and geographical coverage.
4. Requests destined for different originating servers will travel through different caching neighborhoods. Since the data inquiry paths could be established dynamically with different proxy servers, it indicates that the caching hierarchies thus built are dynamic.

To simplify the explanation of using such a dynamic caching neighborhood infrastructure, the idea is described here without the actual complexity. The key to fulfilling a request in the caching neighborhood scheme is that every single request has to be processed by at least one member of the caching neighborhood, either a caching representative or the originating server. Therefore, a request from the user *A* to the originating server *F* providing the needed document may follow the path *user -> client-end proxy server -> proxy server of the caching neighborhood -> originating server*, which corresponds to *A->B->C->F* in the figure. Of course, that would happen only if there is never a cache hit along the path. In fact, the caching neighborhood scheme demands the client-end proxy forward requests either directly to the originating server or a caching representative. Another example is accessing a document on the other originating server *G* from the same user *A*. The request may take the route *A->B->G* in the case of a cache miss at proxy server *B,* otherwise *A->B* might be the actual request processing path. Generally speaking, if a proxy server represents an originating server, it is entitled to respond to any request for the originating server it represents based on some criteria set for that caching neighborhood. The unfulfilled request at a caching representative should not be forwarded any further and would be redirected to the originating server.

2.2 Proxy servers as caching representatives

A proxy server can choose to be either client-end or non client-end, both of which can function as caching representatives for some originating servers. If a proxy server opts to be non client-end, it will accept only requests from other proxy servers and dedicate itself to the role of caching representatives for multiple originating servers. On the other hand, a client-end proxy server has to handle requests for the registering users. For both categories of proxy servers, the operations and behaviors can be viewed from several perspectives -- from the users, from the external, and from the internal to the proxy servers. In the following, we will discuss and elaborate the operations and behaviors of proxy servers from each of the viewpoints.

2.2.1 View from the users

The internal operations of proxy servers as well as the request fulfilling detail are invisible from a user's perspective. The only user involvement needed is to register with an active proxy server, which becomes a client-end proxy server as the user's entry point to the Web. All the requests from this user will be delivered to and processed by the proxy server.

The selection of a client-end proxy server might be based on a predetermined precedence of a group of proxy servers set up by network administration staff or decided at the discretion of users. A local proxy in the vicinity of an end user's network environment may be a natural choice. However, a user may opt to assign any proxy server in the Internet as its client-end proxy server provided that the access to that proxy server is permitted. The idea is that the users are inclined to find a best Web community to join in the hope that they can receive the best quality of service.

2.2.2 External behaviors

Note that requests for dynamically generated Web documents have to be forwarded directly to the originating servers by the client-end proxy servers. For the rest of the requests, first the client-end proxy server needs to check if it is a caching representative of the originating server which owns the requested document. Assuming that the client-end proxy server happens to be a caching representative. In case it has the document, the document is returned immediately to the client. Otherwise, due to being a caching representative, the proxy server should consult the originating server to retrieve the document, return it to the client and cache it locally. When a proxy server is not able to fulfill a request for not being a caching representative, based on its prior knowledge, the request is forwarded to another proxy server which it assumes to be a caching representative that can service the request. In case the caching representative has the needed document which still conforms to the cache coherence requirement currently set for the neighborhood, it would return the document back to the proxy server that forwarded the request. In turn, the response is delivered to the user and the request is fulfilled. The whole scenario is depicted in Figure 2.

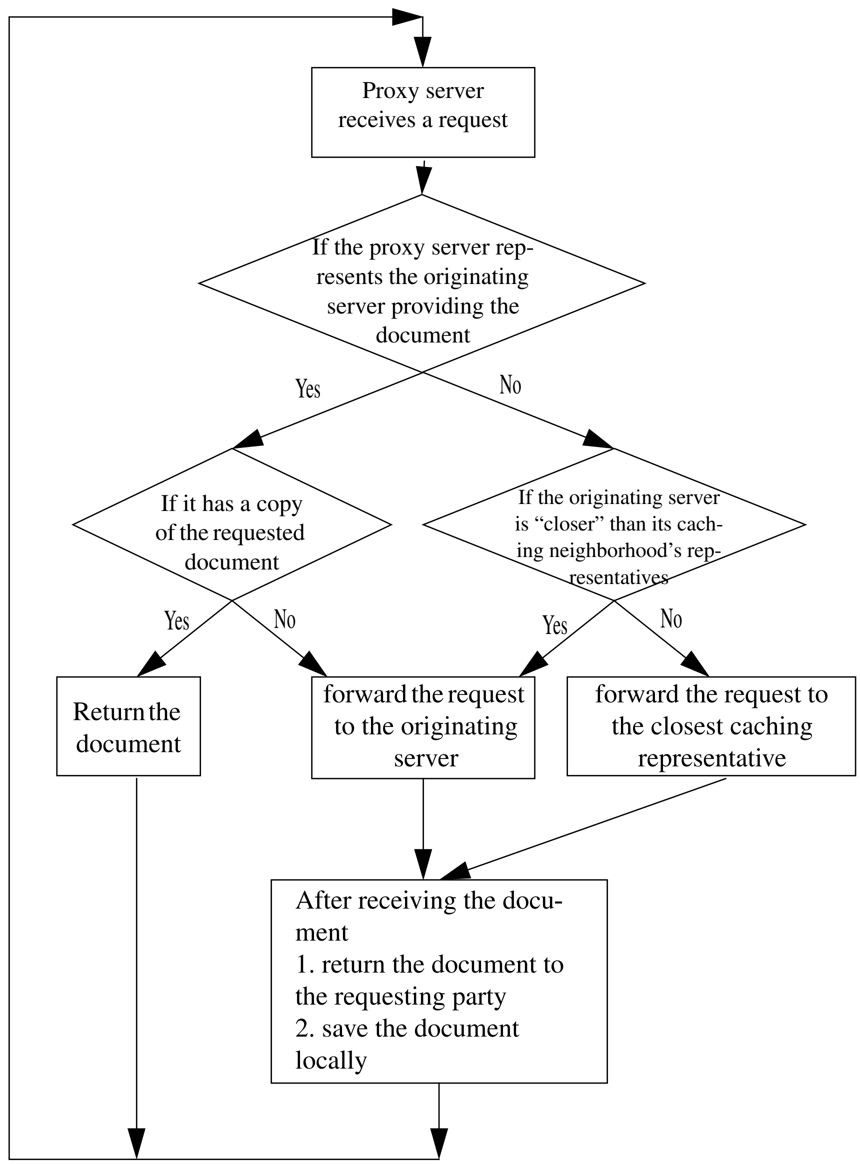

Figure 2 A simplified flowchart demonstrating how a proxy server handles an incoming request

Some exceptions and special cases are skipped here for the clarity of the presentation. For instance, since the caching hierarchies are built dynamically and proxy servers may change the membership status of caching neighborhoods, a caching representative receiving a forwarded request might be no longer entitled to respond and has to inform the forwarding proxy server. Such situations need to be handled with care.

In addition to the inter-proxy communications that are associated directly with servicing a request, one of the keys to the effectiveness of the caching neighborhood scheme is making the information of existing caching neighborhoods widespread. Proxy servers can cooperate as Internet routers do by exchanging information they have for the other proxy servers regarding the representing status and the estimated turn-around time among proxy servers. Such information can be distributed and gathered by each proxy server to derive its own knowledge about all the existing caching neighborhoods.

Finally, proxy servers within a caching neighborhood would also exchange information with the originating server. The purpose of such communication is to understand the status quo of the caching neighborhood as well as to regularly update the data objects already disseminated by the originating server. Again, detail of these message exchanges will be elaborated in Section 3.

2.2.3 Internal operations

Each proxy server internally has a routing table to decide the forwarding destination provided that a request cannot be satisfied. In such circumstances, the proxy server needs to "figure out" in addition to the originating server, where the request can be forwarded to. The decision would be made based on recognizing what proxy servers could also be the candidates of fulfilling the request and the estimated response times between the proxy server and each of the candidates. Apparently, the selection of candidates hinges on the concept of caching neighborhood where caching representatives take care of the requests on behalf of the originating servers. To estimate the response times, in addition to relying on its prior experience in communicating with these candidates, the proxy servers need to learn the existence of other proxy servers nearby so that virtual communication links can be established. The proxy servers can regularly exchange routing information with peers via the virtual communication links so that the collective knowledge base is shared by and distributed to proxy servers in the vicinity. As a result, information items such as representing status, how and when such information is acquired, and average response times between a pair of proxy servers can assist in the decision of forwarding unfulfilled requests. Such configuration is somewhat similar to the Internet routing so that other strategies used for Internet routing could be tailored and then applied to the proxy routing problem.

2.3 Originating servers

An originating server is the heart and soul of its caching neighborhood. The following task list gives an overview on the role an originating server has to play to maintain its own caching neighborhood.

1. An originating server needs to maintain the number of caching representatives and the load distribution of the caching neighborhood by analyzing the requests to make appropriate decisions. An originating server may invite proxy servers to join its neighborhood to distribute its load and it may also drop a caching representative off its list if that proxy server is not of much help in handling requests for this caching neighborhood.
2. An originating server may assist a newly-recruited caching representative by providing a certain percentage of the most frequently accessed documents to set up the cache for the caching representative. The motivation is to reduce the likely high loads on the caching representative when its cache does not contain sufficient documents of the originating server.
3. An originating server has to regularly send update information to its caching representatives and notify them of the validity of the documents the representatives have stored in their local caches. To do so, the originating server has to log the requests it receives from every representative and piggyback the update information along with the responses to the incoming requests to each individual representative. The major benefit is to reduce the response times in cases where cache hits occur at its caching representatives because the representatives can reply to the requests immediately without confirming the validity of their cached documents with the originating server.

4. Broadcasting information about the current status of the neighborhood is also an important task for the originating server. Caching neighborhood information such as the identities of the caching representatives and adjustment of neighborhood members can be delivered to the caching representatives.

3. Caching Neighborhood Protocol (CNP)

The detailed description of CNP is presented in this section. Due to the complexity of the caching neighborhood protocol, some modules are only briefly introduced since the purpose of this paper is to present a overall picture of the protocol. Basically CNP can be divided into two major components: HTTP request fulfillment and caching neighborhood maintenance. The former dictates how proxy servers and originating servers behave as they handle incoming requests, while the latter stresses how a caching neighborhood is maintained and why the caching representatives can oftentimes confidently respond to the requests with locally cached information without verifying the coherence with the originating server in the neighborhood. These components are specified separately in the following sections.

3.1 HTTP request fulfillment

This section focuses on the behaviors of the request handling duties that originating servers and caching representatives assume. A request made by the end users may be processed by the cooperation of client-end proxy servers, caching representatives and/or originating servers. To begin with, a user has to register its Web browsers with a proxy server as its client-end proxy server. The selection and assignment of the client-end proxy server can either be permanent or session by session. As the client-end proxy server is appointed, the requests will be handled by the guideline of CNP. In addition, the following assumptions are used in the ensuing discussion.

- Only GET and GET-IF-MODIFIED-SINCE methods are used in the HTTP requests made by the end users.
- The *timestamp* of a cached document refers to the “last modified time” of the document.

• The *last-known unchanged time* of a cached document on a caching representative refers to the latest time the caching representative receives a confirmation message from the response to a GET-IF-MODIFIED-SINCE request for the originating server.

To simplify the task of presenting the protocol without loss of clarity, some short-hand abbreviations are used. Orig-Svr stands for the originating server which provides the requested document and C-Rep stands for caching representative for the Orig-Svr.

For the proxy servers:

```
-------------------------------------------------------------------------------------------------------------------------
// after receiving a request for registration from an end user
if the proxy server still has capacity to support more users {
   return the message "request for registration is accepted";
}
else {
   return the message "request for registration is denied";
} // end of this part

-------------------------------------------------------------------------------------------------------------------------
// after receiving a request for a document (part 1: caching representative's behavior)
if the proxy server is a C-Rep {
  if the proxy server has the requested document in its cache { // ** a **
     if the timestamp of the cached entry still satisfies the X tolerance conherence requirement currently adopted by
     this caching neighborhood {
        return the document to the requesting party;
     }
     else { // the cached entry violates the X tolerance coherence requirement
        send GET-IF-MODIFIED-SINCE request to the Orig-Svr and wait for reply;
        if timeout and no reply yet {
           return the message "cannot connect to the originating server" to the requesting party;
        }
        else { // get the reply from the Orig-Svr
           return the document to the requesting party;
           update the last-known unchange time of the cached entry;
        }
     }
  } // ** a **
  else { // not having the requested document ** 1 **
     send a GET request to the Orig-Svr and wait for reply;
     if timeout and no reply yet {
        return the message "cannot connect to the originating server" to the requesting party;
     }
     else { // receive the reply from the Orig-Server ** 2 **
        if the original request is in the GET-IF-MODIFIED-SINCE format and there is no change to the document {
           return an appropriate HTTP message to indicate no change on the requested document;
        }
```

```
        else { // the original request is in the GET format
            return the document to the requesting party;
        }
        save a copy of the document in the local cache and perform some logging;
    } // else ** 2 **
  } // else ** 1 **
} // the end of part 1
```

The component for dealing with client registration is straightforward and does not require further explanation. Since handling requests is a complicated procedure for proxy servers, we divide it into two sections for easier referral and explanation. The protocol described above specifies the behavior of a proxy server when it takes responsibility of a caching representative. The notable key points are elaborated as follows.

1. The *X tolerance coherence requirement* is a coherence-efficiency tradeoff adjustment utility which defines the coherence level among the documents stored on the originating server and its caching representatives. The initial, *X*, is a variable which can be set by an originating server to adjust the relative currency of the documents rendered by the members of the caching neighborhood. The value of *X* can be viewed as a degree of distributing the load of an originating server to its caching representatives, which could be any time duration ranging from zero second, ten minutes, two hours, five days to *MAX*, the maximum time duration an originating server consider appropriate. One of the extreme case is *zero second tolerance*, or *zero tolerance*, which indicates that every single request has to get through the originating server, either for the actual document rendering or the verification of a document's last modified time. On the other extreme, if the value of *X* is set to *MAX* and the value of *X* is very large, it implies that the caching representatives can always reply to the requests on behalf of the originating server without ever confirming the currency of the cached documents. Intuitively, the larger the value of *X* is, the less the number of requests an originating server would receive. Therefore, if a server's load is heavy or the documents provided by the server are updated relatively infrequently, the originating server might want to enhance the value of *X* to decrease the amount of traffic coming from its caching representatives. In case the originating server's load is light or it would like to stress the importance of

the coherence of documents, the value of *X* can be decreased accordingly. These are only some simple and direct observations from the piece of protocol presented above. The actual effect of changing the value of *X* depends on many factors such as average and variance of document update frequency as well as the fluctuation of a server's load. Therefore, fine-tuning the value of *X* demands more studies.

2. A caching representative, in addition to storing document content in its cache, also needs to save related information such as *last-known unchanged time* and the *last modified time* of the document for book-keeping purpose. In addition, it also needs to log some other information which would accelerate the regular caching neighborhood status update and shorten the length of the messages exchanged. The issues are not discussed here and we focus only on the fundamentals of the protocol.

The piece of protocol specified below focuses on the behaviors of the client-end proxy servers. In section 2.3, we have discussed how CNP can efficiently and effectively solve the proxy routing problem. We also briefly explained why the combined solution from IP, ICMP, and RIP protocols in the TCP/IP protocol suite [1] cannot be directly applied to the proxy routing problem. The impact of different routing approaches and possible adjustments is currently under study.

```
------------------------------------------------------------------------------------------------------------------------
// after receiving a request for a document (part 2, continue from the end of part 1)
else { //  the proxy server is not a C-Rep ** 1 **
  if the requesting party is a proxy server {
    return the message "no longer a C-Rep" with the original request to the requesting party;
  }
  else { // requesting party is an end-user ** 2 **
    check up the routing table to decide the next forwarding destination among the Orig-Svr and C-Reps;
    if the proxy server has a local copy of the requested document { // ** a **
      if the request is in the GET-IF-MODIFIED-SINCE format {
        set the time to be the newer one between timestamp of the request and timestamp of the local copy;
      }
      else { // the method is GET
        set the time to be the timestamp of the locally cached entry;
      }
      use time to make a GET-IF-MODIFIED-SINCE request;
      send the request to the forwarding destination and wait for reply; -------- (A)
      if timeout and no reply yet { // ** b **
        if the forwarding destination is Orig-Svr {
          return the locally cached entry along with a possibly stale indication to the requesting party;
        }
        else { // forwarding destination is not Orig-Svr
```

```
            set forwarding destination to Orig-Svr;
            update the local routing information of the C-Rep to reflect the timeout situation.
            go back to step (A) in the above;
         }
      } // if ** b **
      else { // receive the reply
         return the document to the requesting party;
         save it in local cache storage if the returned document is newer and storage space permitted;
         update the routing information of the forwarding destination;
      }
   } // if ** a **
   else { // if the proxy server does not have a cached copy of the requested document ** 3 **
      forward the request to the forwarding destination and wait for reply; -------- (B)
      if timeout and no reply yet {
         if the forwarding destination is Orig-Svr {
            return the message "cannot connect to the originating server" to the requesting party;
         }
         else { // forwarding destination is not Orig-Svr
            set forwarding destination to Orig-Svr;
            update the local routing information of the C-Rep.
            go back to step (B) in the above;
         }
      }
      else {
         return the document to the requesting party;
         update the local routing information of the forwarding destination if necessary;
         save it in the local cache storage if space permitted;
      }
   } // else ** 3 **
 } // else ** 2 **
} // else ** 1 **
```

The following two pieces of protocol are self-explanatory. They are presented here for the completeness of CNP.

```
// after receiving a "no longer a C-Rep" message
forward the request to the Orig-Svr, ask for current caching neighborhood information and wait for reply;
if timeout and no reply yet {
   return the message "cannot connect to the originating server" to the requesting user;
   remove the ID of the sender of the message from the C-Rep list of the Orig-Svrs;
}
else {
   return the response to the requesting user;
   update the caching neighborhood information of the Orig-Server;
}
```

```
// after receiving  a "cannot connect to the originating server" message
update the local routing information;
if the requesting party is an end-user {
```

```
   return HTTP message "no response" to the requesting end-user;
}
else {
   return the message "cannot connect to the originating server" to the requesting proxy server;
}
```

<u>For the originating servers:</u>

```
// after receiving a request for a document
retrieve the requested document to generate a response;
perform logging if the document had been downloaded by a C-Rep and was updated since it was last seen;
if the request comes from one of its C-Reps {
   log the necessary information for caching neighborhood maintainence purpose;
   append caching instruction to the response;
}
else {
   if the request also asks for information of C-Reps {
      append the information of C-Reps along to the response;
   }
}
return the response to the requesting party;
```

Therefore, to fulfill a request under CNP, in the component pertaining to HTTP request fulfillment, the originating server does not need to do much except some loggings, which are mainly used to identify what documents have been retrieved by its caching representatives and whether the documents have been updated. Such information will be delivered to the caching representatives to keep the status of each cached document current. On the other hand, the proxy server plays an important role in this protocol. The caching representatives partially replicate the content of the originating server but works on an autonomous and cooperative basis. The potential and advantages of CNP will be presented in Section 5.

3.2 Caching neighborhood maintenance

The originating server plays a prominent role in the effort of maintaining caching neighborhood information. The primary task is to keep the content of the cached documents on the caching representatives coherent with the corresponding documents on the originating server. Besides, the application for becoming a caching representative and the request for the removal of representing responsibility are also administrative jobs for an originating server. On a regular basis the originating server also needs to broadcast the information of the caching neighborhood such as each member's essential data to the caching representatives.

<u>For the originating servers:</u>

```
// receive the request asking for permission to serve as a caching representative
if the Orig-Svr is willing and capable of supporting the proxy server's request {
   compose the returned message which consists of the following:
      1. "request accepted";
      2. identification proof and request for more profile information;
      3. instructions for setting up a persistent HTTP connection;
   return the message to the requesting proxy server and wait for reply;
   if receiving a reply before timeout from a persistent HTTP connection {
      update its caching neighborhood member list;
      assist the new member filling its local cache with the most frequently accessed documents on the Orig-Svr;
      compose and broadcast the "new caching representative" message to the current caching representatives;
   }
   else {
      return the message "timeout and therefore request is rejected";
   }
}
else {
   return the message "request rejected" to the requesting proxy server;
}
-----------------------------------------------------------------------------------------------------------------------------

// receive a request to quit the job of caching representative
if the proxy server of the request is still one of the caching representatives {
   drop the proxy server from the list of caching representatives;
   update the caching neighborhood related information of documents;
   broadcast the message "caching neighborhood updated" to the rest of the caching representatives;
}
return the message "the request of leaving this caching neighborhood is granted";
-----------------------------------------------------------------------------------------------------------------------------

// updating caching entries of its caching representatives
for each C-Rep, if the accumulated statistics collected over a period of time does not match the standard of the caching neighborhood {
   drop the proxy server from the list of C-Reps;
   update the caching neighborhood related information of documents;
   send the message "C-Rep privilege is revoked" to the C-Rep;
   broadcast the message "caching neighborhood updated" to the rest of the caching representatives;
}
else {
   To the best knowledge of the Orig-Svr, notify each C-Rep individually about the modification and/or non-existence
   of the documents ever being downloaded. Popular documents are also prefetched if known to have been updated;
}
-----------------------------------------------------------------------------------------------------------------------------
```

Among the modules described above, the one delineating the procedure for updating caching entries on the caching representatives deserves more treatment. To begin with, this module will be executed by originating servers on a regular basis. The server does not deliver all the updated documents it knows of to the caching representatives. Only the transmission of the most popular documents are initiated by the originating server. Such an approach can ensure that for documents in high demand, even if they are cached, can oftentimes be coherent with the original copies and delivered as the current version by the caching representatives regardless of what *X tolerance coherence requirement* is used. To reduce the processing and transmission time of those update information, the originating servers only notifies the caching representatives of those documents that are still current and those that are outdated. The messages could be coded in a compact format and thus reducing the transmission time if necessary. As a result, we can see that the *X tolerance coherence requirement* only specifies the baseline of the required coherence level and the actual coherence level of rendered documents could be much higher, depending upon the frequency of transmitting the update information of the caching neighborhood.

The correlation of adjusting such update frequency and the *X tolerance coherence requirement* can be observed as follows. Suppose an originating server adopts the *zero tolerance coherence requirement*. Because the caching represention have to resort to the orignating server regardless of having the document or not, the request is guaranteed to be fulfilled with the current document but at the expense of generating more network traffic and prolonging the request response time. Therefore, the frequency of updating the currency information of the caching entries tends to be low. On the other hand, if the originating server enforces the *MAX tolerance coherence requirement*, the updating frequency tends to be high to ensure high probability of rendering current documents by the caching representatives.

Also note that dropping a caching representative from the neighborhood of an originating server should be handled discreetly. By recording the number of bytes of maintainence information the originating server delivers to the caching representatives and the number of bytes of data the representatives handle without the help of the originating server, an originating server can base its decision on the ratio of the two kinds of transmitted bytes to make a decision. Of course, if the amount of neighborhood maintenance information far exceeds the amount of information in terms of number of bytes transffered, the originating server needs to drop this representative since keeping this caching representative defeats the purpose of distributing the load of the originating server. Some criteria also needs to be identified to aid such decision.

If the originating server can be perceptive to all changes of documents, its caching representatives will receive accurate and complete update information regularly and thus the coherence level can be maintained close to, or exactly as if *zero tolerance* coherence requirement is adopted. On the contrary, if an originating server is incapable of detecting what documents have been updated since the previous time it sends the caching neighborhood update information, the originating server can only deliver its knowledge about the document update information based on the recent requests it has handled since the previous time it sends out the document update information. Whenever the load of an originating server is low, a partial or complete walkthrough can be performed on its file system to acquire the update information of all the documents it has provided for its caching representatives.

<u>For proxy servers:</u>

// ask for permission to become a C-Rep for some Orig-Svr
send a request to the Orig-Svr and wait for reply; -------- (A)
if the response is "*request accepted*" {
  establish a persistent HTTP conncetion with the Orig-Svr and wait for reply;
  if timeout and no reply yet {
    retry for a certain times before giving up; go back to (A);
  }
  else {
    coordinate with the Orig-Svr to set up a warm cache;
    update its representing status to reflect the change;
  }
}

// notify an Orig-Svr to drop its committment of serveing as a C-Rep
repeat sending the Orig-Svr "*quitting C-Rep status*" until the message "*the request of leaving this caching neighborhood is granted*" is received;
update its representing status to reflect the change;

// update the local caching entries
if the message "*C-Rep privilege is revoked*" is received {
  update its representing status to reflect the change;

```
}
else {
   update the status of the caching entries as per the notification of the Orig-Svr;
}
```

Establishing a HTTP persistent connection between a caching representative and an originating server can contribute to the reduction of response time because of the removal of conncetion setup overhead. Since the number of caching representatives for an originating server can be controlled, there would be sufficient system resources to support these permanent connections to accelerate the response. It is anticipated that once the caching representatives become established and known to the outside world, considerable amount of requests received by an originating server would come from its caching representatives, which could further reflect the benefit attributed to employing persistent connections.

4. Analysis

In this section, our findings on the traces of user requests observed at a proxy server are presented. The collected statistics hint that CNP could be promising if adopted. These access traces of a proxy server were downloaded from NLANR [16] and examined during the period 01/12/99 - 01/16/99. Totally there are 5,427,907 accesses made to 1,797,264 distinct documents spreaded over 112,955 Web sites. The foundation of our argument relies upon the assumption that in the caching neighborhood approach, proxy servers are inclined to represent a small percentage of Web sites which attract considerable amount of accesses. Proxy servers may derive accessing statistics by observing the document requesting frequencies at their own Web community. Nevertheless, in the following we would like to argue that establishing caching neighborhoods for a low percentage of Web sites should be sufficient to benefit the Internet users. The following claims, derived from the trace study, display that the caching neighborhood concept gains support directly from the accessing charcteristics of Web documents.

1. A low percentage of Web sites account for a large percentage of total accesses.

2. The most popular x% documents covers y% total accesses and the rest (100-y)% accesses are mostly for the rarely accessed documents.
3. The most popular documents are mostly located at the popular web sites.

The first claim implies that caching only popular Web sites is a good idea. The availability of most of the requested documents can be greatly increased if caching neighborhoods can be established and stretched out for some Web sites. As will be seen in Figure 7, around 90% total accesses accounted for by only 20% Web sites being accessed. In the second claim, if the value of y is large while the value of x is still small, it indicates that caching mostly popular documents will help make the Web caching succeed. The myth is clarified in Figure 5 that 20% documents account for about 60% total accesses while the rest 40% accesses is covered by 80% documents. The encouraging news is, as observed in Figure 4, around 32% accesses are made to documents that have been accessed only once! To summarize, the first claim indicates that caching should be focused on popular sites and the second claim indicates that caching would be most effective if only the most popular documents are cached. In practice, finding out popular sites can be easily achieved by collecting statistics, but how to identify if a document is a popular one and better be cached in the first place?

Fortunately, the third claim answers the above question. Basically it says that caching documents from popular Web sites is the perfect choice. It is quite amazing to find that the most popular 20% Web sites provide over 90% of the most popular 20% documents (see Figure 9). By simple math, we also find that these most popular 20% sites responsible for 90% * 60% = 54% total accesses with requests for the most popular documents. Therefore, the remaining 90% - 54% = 26% total accesses directed to the most popular 20% sites will not benefit caching much. It suggests that not every single document provided by the popular Web sites deserves to be cached. This fact is taken into account by CNP since a Web server can easily identify the popularity of the documents it provides by keeping their access counts. Therefore, the potential usefulness of CNP is justified.

For each figure shown in this section, we supply statistics on accesses collected in single days as well as accumulated accesses over multiple days. The reasons for doing that are two-folds. (i) We want to examine if the statistics will vary dramatically for a fixed proxy server on different days. (ii) We have not done extensive study about the update frequency of documents and it is hard to arbitrarily decide the length of trace evaluation period. As a result, both sets of statistics are provided for comparison purpose.

Figure 3 displays the document access distribution. We can see that a high percentage (around 75% in multiple days' trace and 80% in a single day's trace) of documents are only accessed once. The drop of such accesses from 80% to 75% is not surprising because the rarely accessed documents have a better chance of getting hit again in a longer period of time. As we look into accesses accumulated over five and three consecutive day periods we find both are very similar. Hence, we speculate that the situation will not change a lot as we extend the examinaion period on traces, which is currently being collected. The fact strongly suggests that a good percentage of documents do not deserve being cached and caching such documents is just wasting time and storage space.

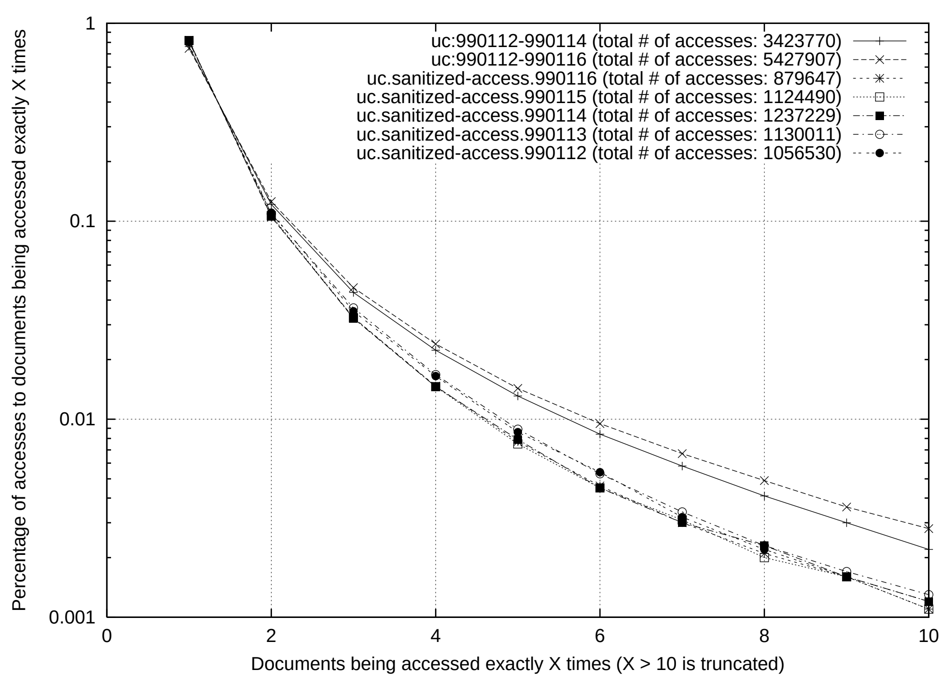

Figure 3 Document accessing distribution

The curves in Figure 4 indicates that the accumulated percentage of accesses to documents which have been accessed at least twice is about 54% in a single day's trace, and it is climbing to around 66% in a five day trace. It also indicates that at least 34% of accesses in the five day trace are made to documents receiving only one hit.

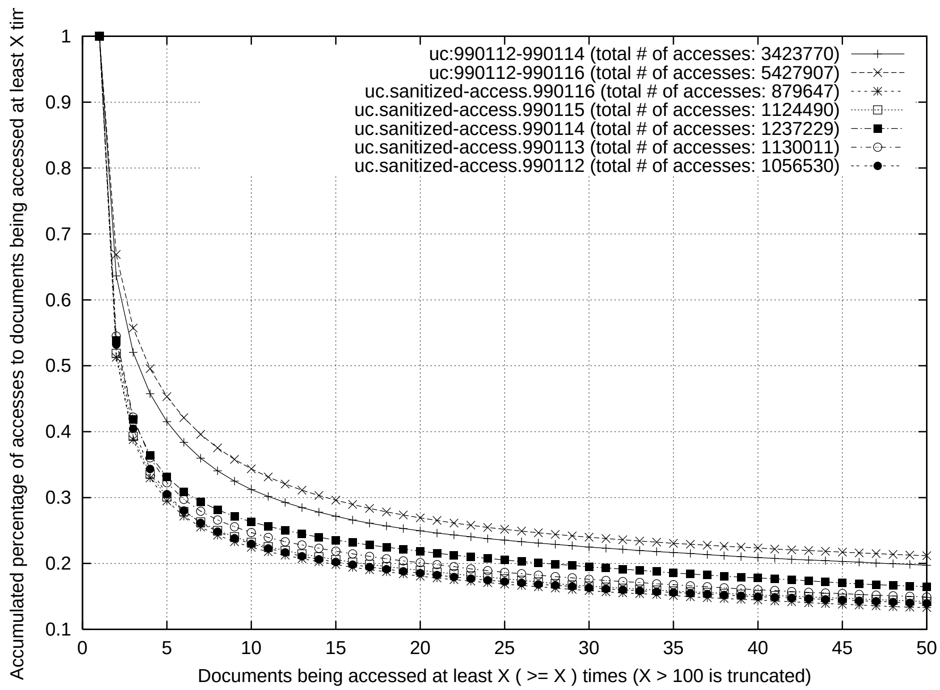

Figure 4 Accumulated document accessing distribution

The curves appear in Figure 5 directly justify the truth of the aforementioned claim 2. It can be observed that about 20% documents account for up to more than 60% total document accesses. The curve grows exponentially as the value of X in those figures are small, which indicates that for the Web caching to be effective, those top documents are "can't misses". Another important information revealed by this figure is that only the most popular 25% documents were hit at least twice and the most popular 20% documents account for about 63% total accesses. Therefore, the top 20% documents cover 94% documents that have been accessed more than once. This result turns out to have a profound effect in designing Web caching strategies. In the static caching hierarchy schemes, even a request counts as a 'cache hit' does not necessarily mean the returned document is current. Therefore, many hits are “false” and should be considered misses otherwise.

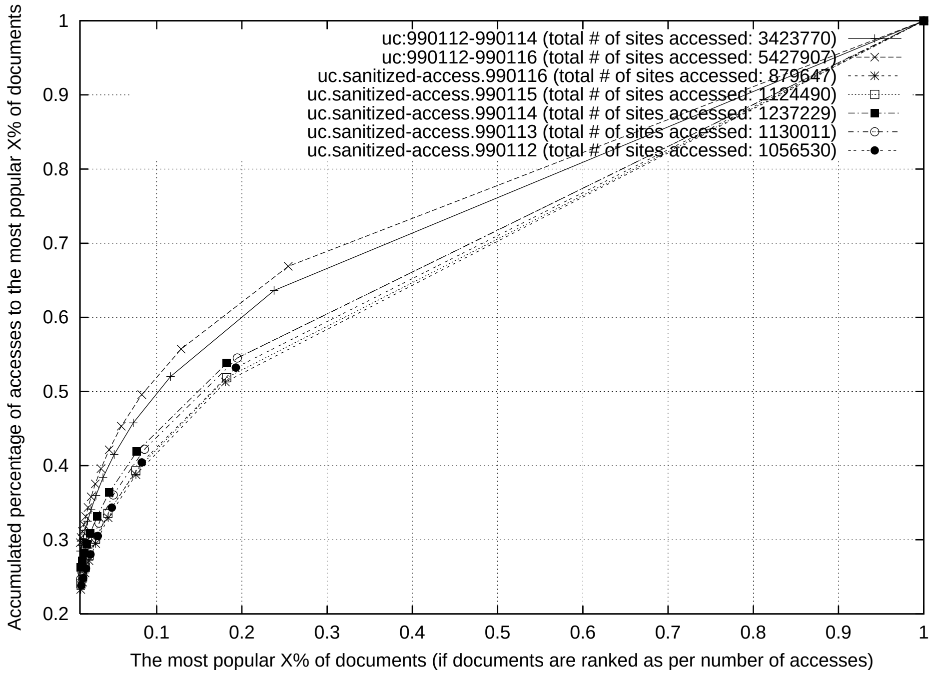

Figure 5 The most popular X% of documents and the accumulated accessing percentage

Some studies are performed in order to clarify the effect of the popularity of Web sites (originating servers) on caching. Figure 6 points out that there is a considerable percentage of Web sites receiving very few requests, even as we prolong the length of observation period. For a five day observation span, the accumulated percentage of Web sites that receive no more than ten requests is about 65% and 25% Web sites receive only one hit in that period. This further indicates that it is not beneficial to cache documents from those not so popular Web sites.

Figure 7 identifies a very useful insight for every caching solution. It suggests that documents provided by popular Web sites really deserve to be cached. The statistics show that the most popular 20% Web sites account for almost 90% total accesses and the most popular 30% Web sites receive about 93% total accesses. That’s a huge factor in designing caching strategies.

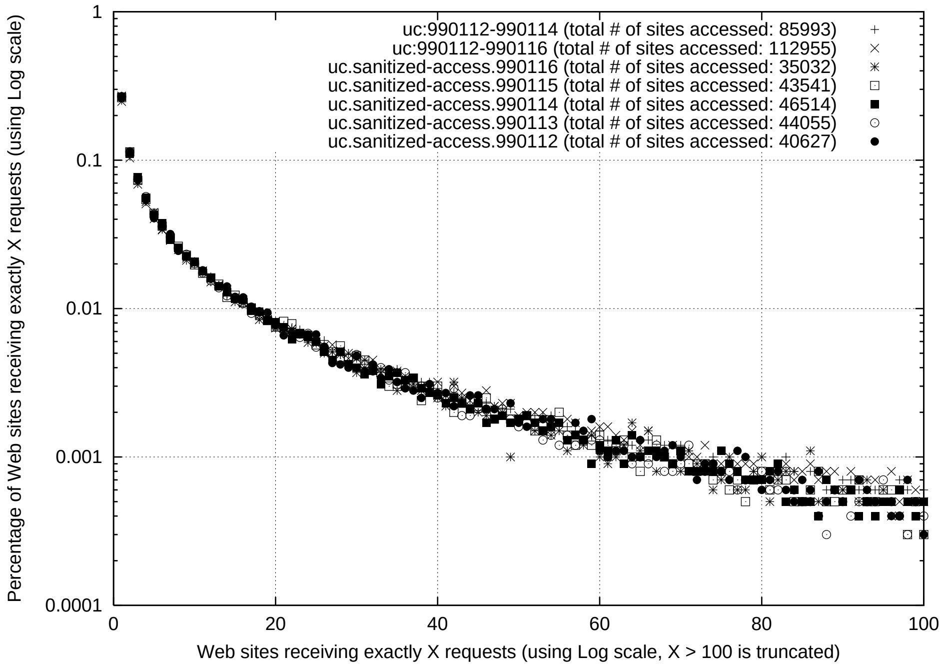

Figure 6 Accessing percentage of Web sites

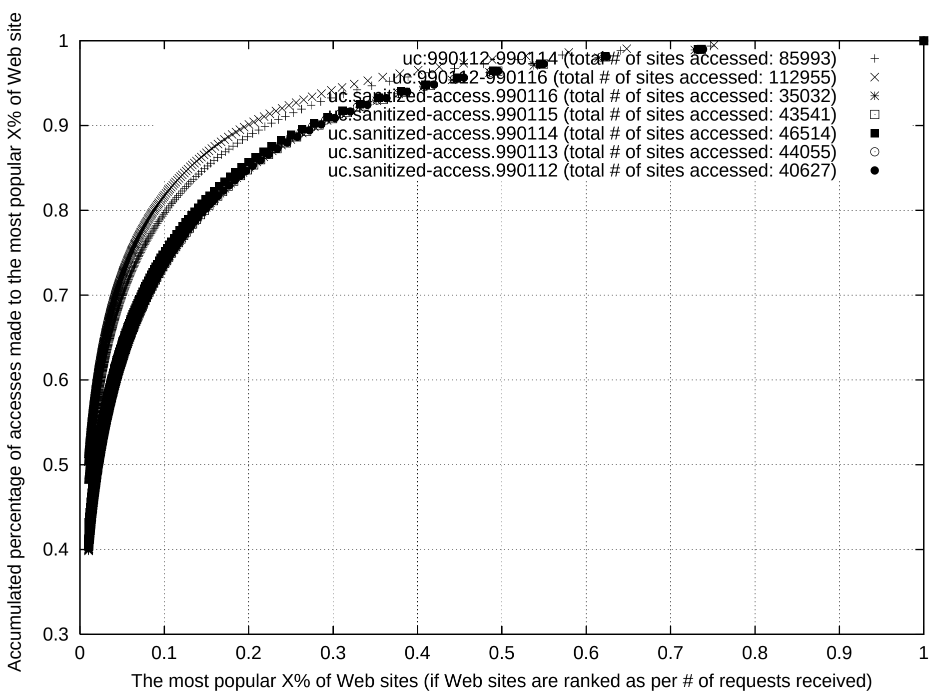

Figure 7 The most popular X% of Web sites and their accumulated accessing percentage

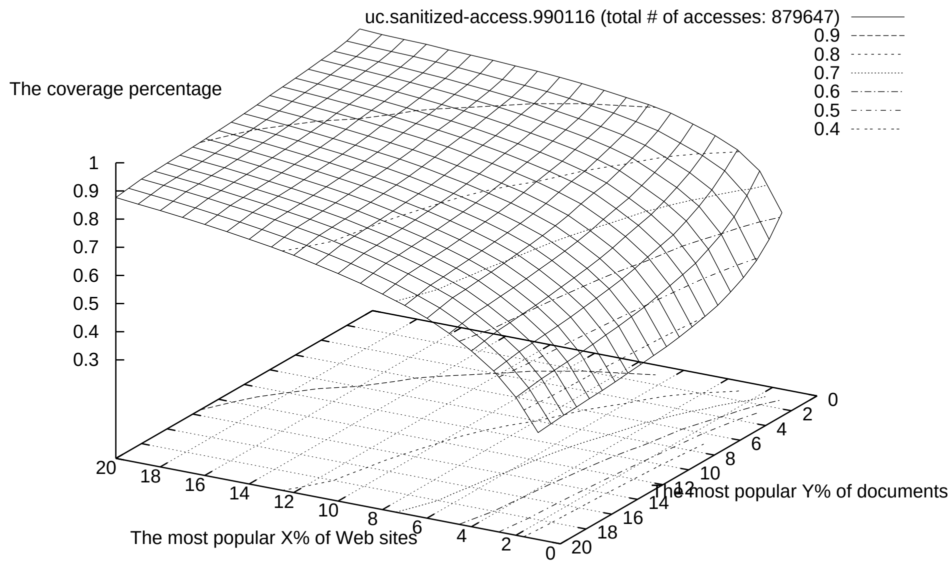

Figure 8 Coverage percentage of the most popular 20% of documents by the most popular 20% of Web sites for accesses accumulated in a one day period

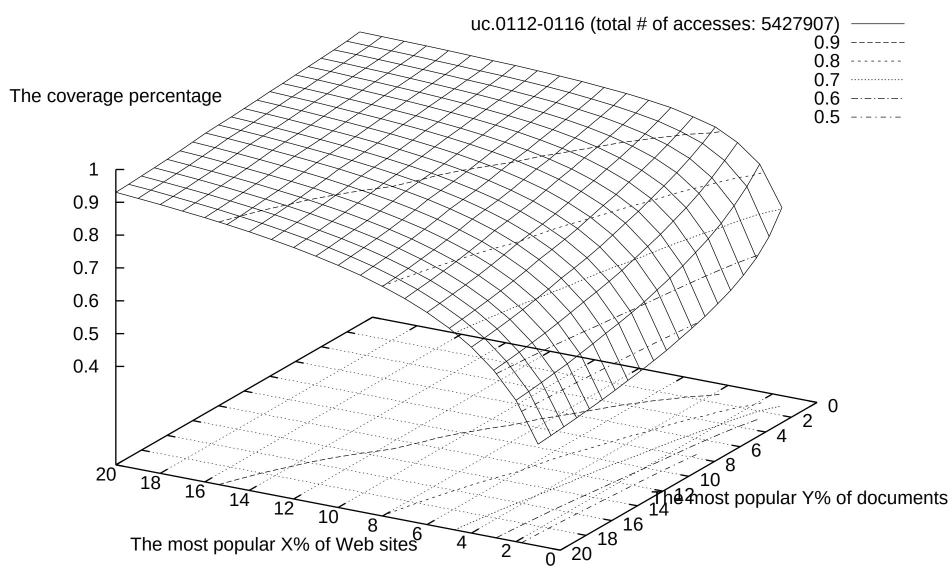

Figure 9 Coverage percentage of the most popular 20% of documents by the most popular 20% of Web sites for accesses accumulated in a five day period

The claim 3 is vindicated by the curves in Figure 8 and Figure 9. These two figures display a strong correlation among popular Web sites and popular documents where popularity is determined by the number of accesses received. Figure 8 is based on the access log of a single day and Figure 9 is based on the access log collected over five consecutive days. By observation, they are similar in shape except that the coverage percentage for the five day access log is a little bit higher at a fixed (X,Y) pair compared to the one in one day log. In the five day trace, the most popular 20% Web sites account for almost 90% total accesses. More than 40% of the most popular 20% documents are supplied by the most popular 1% Web sites, 90% of the most popular 20% documents are supplied by the most poular 15% Web sites, and about 93% of the most popular 20% documents are supplied by the most popular 20% Web sites. The concentration of document accesses on a small percentage of sites is so overwhelming that it again indicates the direction of caching mainly the most popular Web sites is rather promising.

5. Discussion

This section is devoted to the discussion of various issues relating to CNP. Like every other approach, ours has its advantages, shortcomings, and some side-effects that are hard to evaluate their impacts. Note that two of the major motivations of proposing CNP are reducing the response times perceived by the end users while in the meantime minimizing the amount of network traffic. We will present the potential advantages and attempt to analyze the arguable advantages.

Potential advantage 1: CNP provides an effective solution for the cache coherence problem. With the introduction of the *X tolerance coherence requirement*, the coherence problem is no longer compromised for shorter response times or over-stressed at the price of prolonging the response times and generating more network traffic. The originating server has absolute control over the coherence level to be used by its caching representatives. The idea is to strike a balance between the load and document coherence an originating server can afford to offer.

Potential advantage 2: The dynamic caching hierarchies embodied with the use of CNP can be incorporated into today's static caching hierarchy schemes. Of course, the proxy servers currently in use have to be upgraded to be aware of the existence of the caching representatives for originating servers. The easiest way to adapt proxy servers running ICP to comform partially to CNP is when a cache miss occurs in a cache, in addition to forwarding the request to the siblings and parent, it also forwards the request to one of the caching representatives which can respond to the request. Such an approach can be regarded as an intermediate step to the adoption of dynamic caching hierarchy scheme.

Potential advantage 3: CNP also provides promising hopes to reach a higher cache hit rate for the proxy servers, which stands for better utilization of the proxy resources. As demonstrated in the previous section, CNP can enhance the cache hit rate by making partially replicate servers, incarnated as caching representatives, bring the most frequently accessed documents to the Web communities. Because the caching representatives will also avoid caching the extremely unpopular documents with instructions from the originating server, the potential cache hit rate could be even higher than expected.

Arguable advantage 1: If we compare running CNP to running ICP on proxy servers, CNP likely will reduce the response times. The comparison is based on the assumption that CNP is widely adopted and many proxy servers have participated in various caching neighborhood. To begin with, there are two major components contributing to the response time -- turnaround time and proxy processing time. The intuition is that CNP is capable of reducing the turnaround time by spreading out the caching representatives all over the places. Chances are the needed documents are moved much "closer" to the end users and thus the turnaround time should decrease. The only questionable part in the above statement is that it does not take into account how "close" the caching representative would be and how widespread the caching neighborhoods have been extended. Moreover, CNP is compared to ICP at a disadvantage because CNP provides coherence level guarantee on cached documents while ICP does not.

The argument that claims CNP can also reduce proxy processing time is as follows.

a. Redundant hierarchies in the static caching hierarchy schemes are removed, so are the inquiry timeouts. CNP's hierarchy structure is simple and only has two layers -- client-end proxy servers and caching representatives.

b. The potential hit rate for requests received by proxy servers, if using CNP, is higher than using ICP.

c. CNP enforces a persistent HTTP connection for each caching representative connecting to its originating server. There is no HTTP connection setup time overhead for a large group of requests.

d. The originating servers have distributed the loads to their caching representatives. Therefore, it would have more capacity to handle the requests that mostly come from its caching representatives. Furthermore, a possible enhancement technique could be applied to CNP where the originating servers assigns higher processing priority to requests coming from the caching representatives.

Arguable advantage 2: If we compare running CNP to running ICP on proxy servers, potentially CNP will decrease the amount of network traffic relating to Web browsing. The basis for such a claim has credibility mainly because the potential hit rate for requests received by proxy servers, if using CNP, is higher than using ICP. Therefore, less requests need to get to the originating server for service. However, the limitation is that the originating server will need to deliver update information regularly to the caching representatives. The validity of the above claim hinges on the degree of load distribution that is achieved by the caching representatives. Of course, if such a degree is high, i.e., the caching representatives handle a considerable amount of requests on behalf of the originating server and are much "closer" to the client-end proxy server in terms of hop distance, the amount of traffic caused by the delivery of caching neighborhood update information can be easily offset. The actual tradeoff relationship requires more studies.

6. Future works

There are some aspects of CNP requiring more work to improve the performance. The following is a list of tasks we expect to pursue.

1. The fundamental protocol described above does not address the detail of issues on how the proxy servers adjust themselves to the shift of popularity from one group of Web sites to another group because proxy servers may opt to change their representing status. In the prototype implementation, we will just assume that a proxy server picks up originating servers to represent based upon the popularity of the servers observed locally along with the capacity of the proxy server itself to simplify the matter. A better criteria may be more sophisticated in order to make caching representatives really autonomous.

2. A key to the success of CNP is how the routing issues are solved. Clearly the routing table lookup will be a factor in determining the performance in terms of both effectivity and efficiency of the approach. Certainly there are some simple and straightforward solutions to the problem. The strategy of maintaining routing table entries and accelerating routing table lookups are our main concerns. We are looking into ideas that could possibly improve the performance.

3. As mentioned previously, the interval of sending update information, coding of such messages, and the scheme adopted in propagating caching neighborhood update information are also of concern. It is a complicated factor that could influence the performance in multiple ways. For example, high update frequency can enhance the coherence level but increase the load on the originating servers.

4. Only qualitative reasoning on why CNP can improve the performance of using proxy servers over ICP is given. Therefore, one of the ongoing research effort is to understand the differences in performace measurement when comparing CNP against ICP quantitatively. Another relevant research is to collect related statistics and build a mathematical model to analyze how the performance of deploying CNP would be improved by considering metrics such as response time, hit rate, and amount of network traffic generated.

7. Summary

In this paper, the possibility of employing dynamic caching hierarchies for the Web documents using a distributing infrastructure is discussed. First, we introduce the notion of caching neighborhood in which originating server and proxy servers form a cooperative partnership to distribute information. Then we use concrete examples to show how the caching neighborhood concept can be applied to establishing dynamic caching hierarchies. After that, we present and explain the kernel of the Caching Neighborhood Protocol (CNP) and supply relevant explanation information. Some applicable statitistics of accesses to documents and Web sites derived from traces of a Web proxy server are plotted. The purpose of presenting the statistics is to justify the proposed approach. We also devote a section to discussing various issues of CNP and compare CNP favorably against Internet Cache Protocol (ICP) using qualitative reasoning.

8. Acknowledgement

We would like to express our gratitude to NLANR for collecting and providing access to the Web proxy accessing logs. This study would not be possible without the log of access traces.

References

[1] W. R. Stevens. "TCP/IP illustrated, volume 1" Addison Wesley, 1994

[2] A. Chankhunthod, P. B. Danzig, C. Neerdals, M. F. Schwartz and K. J. Worrell. "A Hierarchical Internet Object Cache" in Proceedings of the USENIX Technical Conference, San Diego, CA, January 1996 (http://catarina.usc.edu/danzig/cache/cache.html or http://excalibur.usc.edu/cache-html/cache.html)

[3] A. Dingle and T. Partl "Web Cache Coherence" Computer Network and ISDN Systems, 28(1996), pp. 907-920

[4] A. Luotonen, K. Altis "World-Wide Web Proxies" Computer Networks and ISDN Systems 27(1994) pp. 147-154

[5] T. Berners-Lee, R. Calliau, A. Luotonen, H. Nielsen, and A. Secret. "The World-Wide Web". Communications of the ACM, 37(8), pp.76-82, 1994

[6] C. Liu and P. Cao. "Maintaining Strong Cache Consistency in the World-Wide Web" in Proceedings International Conference on Distributed Computing Systems, 1997

[7] D. Wessels and K. Claffy. Internet cache protocol (ICP), version 2. (http://ds.internic.net/rfc/rfc2186.txt, 1998.

[8] J. Gwertzman and M. Seltzer. “The case for geographical push caching” in Proceedings of the 1995 Workshop on Hot Operating Systems, 1995 (http://www.eecs.harvard.edu/~vino/web/hotos.ps)

[9] J. Gwertzman and M. Seltzer. “World Wide Web Cache Consistency” in Proceedings of the 1996 USENIX Technical Conference, San Diego, January 1996.

[10] K. J. Worrell. “Invalidation in Large Scale Network object Caches” Master Thesis, Dept. of Computer Science, 1994.

[11] L. Zhang, S. Floyd, and V. Jacobson. “Adaptive Web caching” in the 2nd Web Caching Workshop, Boulder, Colorado, June 1997. (http://ircache.nlanr.net/Cache/Workshop97/Papers/Floyd/floyd.ps)

[12] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams and E.A. Fox. “Caching proxies: Limitations and potentials”, Proc. Fourth Intl. World Wide Web Conference., Dec. 1995, Boston, MA

[13] M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm. “Enhancing the Web’s infrastructure: from Caching to Replication” IEEE Computing, pp. 18-27, March-April, 1997

[14] M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm. “World Wide Web Caching: The application-level view of the Internet”, IEEE Communications Magazine, June 1997

[15] M. Nabeshima. “The Japan Cache Project: An Experiment on Domain Cache” in Proceedings of the 6th International World Wide Web Conference, 1997.

[16] National Lab of Applied Network Research, Sanitized access log, Jan. 1999. (ftp://ircache.nlanr.net/Traces/)

[17] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk and T. Berners-Lee. “Hypertext Transfer Protocol - HTTP/1.1 “

[18] RFC 959: File Transfer Protocol. (http://www.cis.ohio-state.edu/htbin/rfc/rfc959.html)

[19] RFC 1436: The Internet Gopher Protocol. (http://www.cis.ohio-state.edu/htbin/rfc/rfc1436.html)

[20] Steven Glassman. “A caching Relay for the World Wide Web” in Proceedings of the 1st International Conference on the WWW, 1994.

[21] Squid Internet Object Cache (http://squid.nlanr.net/Squid/)

[22] V. N. Padmanabhan and J. C. Mogul. “Using Predictive prefetching to improve World-Wide Web Latency” ACM Computer Communication Review, pp.22-36, v.27, n3, July 1996.

[23] L. Fan, p. Cao, J. Almeida and A. Z. Broder. “Summary Cache: A scalable wide-area cache sharing protocol” in Proceedings of the ACM SIGCOMM’98, October, 1998.

[24] A. Rousskov. “Cache digest” April, 1998 (http://squid.nlanr.net/Squid/CacheDigest/)