Web caching and document distribution
Course team
Developer: Nancy J Yeager
Designer: Chris Baker, OUHK
Coordinator: Dr Reggie C Kwan, OUHK
Member: Dr Andrew Lui, OUHK

External Course Assessor
Professor Mingshu LI, Institute of Software at the Chinese Academy of Sciences

Production
ETPU Publishing Team

Acknowledgement
The OUHK gratefully acknowledges a grant of $100,000,000 from the Hong Kong Government designated for course development.

MT834 Web server Technology was developed in part using resources provided by this generous grant.

Copyright © The Open University of Hong Kong, 2001.

All rights reserved.
No part of this material may be reproduced in any form by any means without permission in writing from the President, The Open University of Hong Kong.

The Open University of Hong Kong
30 Good Shepherd Street
Ho Man Tin, Kowloon
Hong Kong
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Objectives</td>
<td>2</td>
</tr>
<tr>
<td>Organization of the document tree</td>
<td>3</td>
</tr>
<tr>
<td>Mirroring</td>
<td>8</td>
</tr>
<tr>
<td>Caching Web information</td>
<td>12</td>
</tr>
<tr>
<td>The effectiveness of caching</td>
<td>26</td>
</tr>
<tr>
<td>Hierarchical caching</td>
<td>28</td>
</tr>
<tr>
<td>Unresolved problems for Web caching</td>
<td>30</td>
</tr>
<tr>
<td>Summary</td>
<td>32</td>
</tr>
<tr>
<td>Feedback to activities</td>
<td>33</td>
</tr>
<tr>
<td>Feedback to self-tests</td>
<td>36</td>
</tr>
</tbody>
</table>
Introduction

Unit 6 examined ways of boosting the performance of a Web service by improving the design and operation of the Web server software, computing platform, and network. Another way of improving the performance of a Web service is to increase the number of places on the network from which users can fetch the same set of Web documents. The more Web servers there are serving the document, the more users that will be able to access the document with satisfactory performance.

This unit will explore several techniques for replicating and distributing documents on the Web. The following techniques will be discussed:

- Mirroring which replicates and distributes Web documents on several Web servers.
- Using a browser cache which replicates and distributes Web documents on the user's local computer disk.
- Using a LAN proxy cache which replicates and distributes Web documents on the local area network.

Each of these approaches offers its own advantages, disadvantages, and management and configuration considerations.

The first half of this unit describes several ways of organizing a Web server document collection. The second half of the unit explores caching technologies for Web documents. These topics are required knowledge for a Web administrator who is setting up and managing Web services. As part of Unit 7, you will set up a Web browser cache and a LAN proxy cache and conduct experiments to see how these caches work. You will also learn about the effectiveness of caching technologies in a Web server environment.

You need to read Chapter 5 from your Web Server Technology textbook as you work through Unit 7. The organization of the unit closely follows the textbook. The unit should take you about three or four weeks (or about 25 to 30 hours) to complete. Please plan your time carefully.
Objectives

By the end of Unit 7, you should be able to:

1. Explain why Web directories are logical names that map to physical directory names.

2. Evaluate the benefits and limitations of mirroring as a Web replication technique.

3. Install and configure a Web caching proxy and a Web browser cache.

4. Trace through a sample Web request to a Web proxy and understand how the Web caching proxy works.

5. Evaluate the benefits and limitations of using a Web cache: cache consistency issues, effectiveness, and costs.

6. Describe how both a load balancing network appliance and a Web cache network appliance work.

7. Use the HTTP header fields and the HTML tags that affect caching.
Organization of the document tree

A Web server’s document collection can have many shapes and sizes. The document set may be very large — too large to fit on a single hard disk drive or on a single computer’s file system. Or the Web documents may be spread across several computers for several reasons. The documents in the collection may have been contributed by a group of people, each having access to only one computer system. Or, a corporate department’s documents may be on several computer systems due to a company reorganization. Whatever the reason, it is possible that a lot of effort may be required, or it may not be possible at all, for the Web server administrator to reorganize the document collection into the shape that Web users expect.

As you learned through past experiments, the Web server presents its document set according to a logical view as a single tree structure with a Web server document root at the top of the hierarchy. Users of the Web service will see the document collection according to the logical view presented by the Web server, regardless of how the documents are physically organized as files in a file system on one or more computers. One of the Web server’s jobs is to translate from the logical view of the single tree hierarchy to the physical location of the documents. The next reading will explore the organization of the Web document tree and reasons why Web documents may be distributed among several Web servers.

Reading

*Web Server Technology*, Chapter 5, Section 5.1, page 177 to the middle of page 184.

The above reading explained how mappings, expressed as the MAP command, translate between the logical view of a single tree of documents (what the users on the Web see) and the true organization of the files (the way the files are actually stored).

This mapping between the logical view and the physical files can be maintained by the operating system’s file system or by the Web server. An operating system maintains a logical view of files with distributed system tools such as file systems (Network File System (NFS), Andrew File System (AFS), Distributed File System (DFS), Novell, Appleshare) or Unix symbolic links. A distributed file system and Unix symbolic links allow the administrator to ‘mount’ or ‘paste’ a directory that exists in one physical location, in file system A on Computer A on top of a directory in file system B of Computer B. The end result is one large file system that appears to be in only one place, similar to the ‘Network Neighborhood’ on Windows systems. The distributed file system or Unix symbolic links creates the logical view of one big file system and maintains the links where the individual directories map into the larger file system.
Currently, Unix-based operating systems provide more choices than the Windows operating system in the available tools to create a single logical file system. The Windows shortcut facility is similar in some ways to Unix symbolic links but it is not a true symbolic link facility that is transparent to applications that use the file system. The Apache Web server for Windows does not currently recognize a directory shortcut as a mapping to another directory; so shortcuts cannot be used to create a logical view of the Web documents. Therefore in a Windows operating system environment the Web server is likely to be the sole maintainer of the logical document view. In this case the Web server can create and maintain mappings from the logical view, the URL requested by the Web user, to the true storage location of the requested file. The next activity will demonstrate the configuration tools your Apache Web server uses to impose a logical view of the Web document collection.

**Activity 7.1**

In this activity you will experiment with your Apache Web server’s ability to present a single tree logical view of Web directories from several disjoint directories. You will use the Apache Alias directive to create the logical view. The Alias directive is similar to the ScriptAlias directive you used in *Unit 3* when setting up CGI scripts. The Alias directive has the same function as the MAP command shown in the *Web Server Technology* text. For more information about how to use the Alias directive read the Apache Web server manual at <http://localhost/manual/mod/mod_alias#alias>.

Imagine that you are a Web administrator for a group who wants to set-up an online encyclopaedia of animals. The directories containing the encyclopaedia’s HTML pages already exist and are organized in the following way in the Web server’s file system:

Physical Directory Structure
(the position of the directories in the filesystem)

```
$(Web_Document_Root)
```

```
   flying_mammals
       bats

   general_mammals
       horses
       whales

   insect_eating_mammals
       moles
       shrews
```

The new Web service wants to present to the Web users a logical directory structure that is organized in the following way.
Your assignment is to use the Apache Web server Alias directive to create the logical view shown above. Contributors to the online encyclopaedia expect the current physical directory organization to remain the same; it is not an option to physically reorganize the directories in the Web server file system. Please disable the Java Spy program (from Unit 3) in this experiment.

Activity 7.1 demonstrated that the Apache Web server Alias directive may be used to relocate a Web document on the same Web server computing platform without disrupting the user’s logical view of the website’s document tree. You could also use the Apache server URL rewriting facility to achieve the same result as the Alias directive. Most Web servers will provide one or more features to maintain these logical-to-physical mappings.

Sometimes, though, as a Web administrator you may find that you need to relocate a Web document or a section of the whole Web document tree to a file system on another Web server computing platform. For example, when two companies merge and change their domain name the company usually wants to create a new homepage at the new domain name and redirect the user’s requests from the two old homepages to the new homepage. Another situation that could benefit from relocating portions of the Web document tree is an overloaded Web server. To boost performance the Web document tree may be split between two computing platforms:

\(<\text{http://www.company.com}\>\>
\(<\text{http://www.newserver.com}\>\>

Each half of the document tree is now served by its own CPU and Web server software, in theory, doubling the serving capacity of the website.

The problem that both these websites now face is the need to redirect user’s requests to the computer with the new domain name. One could simply change all the hypertext links on the Web server HTML pages to point to the Web server with the new domain name. This could be very time-consuming for a large website. Plus, there is still the issue of user’s ‘bookmarks’, or ‘favourites’ or other hypertext links in other user’s webpages. As you reconfigure a website some URLs become invalid. You could maintain HTML pages on the original website instructing
Web users to go to new URL. But this solution is not very user-friendly; it forces the user to manually traverse another hypertext link and be aware of the domain name change. The user-friendly solution is to automatically redirect the user’s request to the new Web server. To see a website that redirects users’ requests, point your Web browser to <http://abc.go.com/astrologyzone/>. What happens? The Web browser automatically displays the URL for <http://www.astrologyzone.com>; the browser request has automatically been redirected without any user assistance. How is this done?

The Apache Web server, as well as much other Web server software, has the capability to automatically redirect outdated URLs to another new URL. The Redirect directive configures the Apache Web server to map the old URL to the new URL. When the Web server encounters a user request for the old URL it sends the new URL to the Web browser so it can automatically fetch the requested document at the new URL. The user is not aware that their request has been redirected, other than perhaps a somewhat longer delay in fetching the document. In the next activity you will see how this process works.

Activity 7.2

In this activity you will experiment with your Apache Web server’s ability to map to redirect a user request from one URL to a new URL with a different domain name. You will use the Apache Redirect directive (<http://localhost/manual/mod/mod_alias.html#redirect>) to create the mapping.

1 Make sure you do not use the Java Spy program from Unit 3 in your localhost experiments.

2 Set up two Web servers on your localhost, each with their own copy of the Apache Web server documentation as a document tree. One Web server will be listening on port 80; another Web server should be listening on port 8080.

3 Modify the Web server on port 8080 to redirect homepage requests to the Web server on port 80 using the Apache Redirect directive.

4 Request <http://localhost:8080> with your Web browser.

5 Examine the log files for both Web servers. What did you observe? What conclusions can you make?

Another way to selectively redirect a Web browser to a new URL on a different Web server is to put the redirect instruction in the HTML page. HTML META tags are used to communicate meta-information, descriptive information about a HTML page. In this case the meta-information is the updated URL where the browser should be redirected.
Here is how the HTML META tags work. The Web server reads the META HTML tag in the HTML file and creates a HTTP Entity Header Field according to the format in the META tag. For this “META refresh” tag:

```html
<META HTTP-EQUIV="refresh" content="http://www.any.com" />
```

the Web server reads the META HTTP-EQUIV tag and writes the title of the Entity Header as Refresh and the contents of the Entity Header Field as URL=http://www.any.com/. The Entity Header Field sent by the Web server to the Web browser is Refresh: URL=http://www.any.com. The browser reads the Entity Header Field and automatically redirects the user’s request to the new URL, <http://www.any.com>. Most browsers and Web servers support the HTML META tag mechanism. But, some browsers and Web servers do not support META tags and this can be a big disadvantage to using this redirection mechanism. In the next activity you will explore how HTML META tags work for redirecting traffic to a new URL.

**Activity 7.3**

Your assignment is to create two HTML pages that constitute an infinite redirection loop and install them on your localhost Web server. This means that Page1.html will redirect the browser to Page2.html and Page2.html will redirect the browser back to Page1.html, thus creating an infinite loop of redirections.

1. Read about the syntax of HTML META tags at: <http://htmlhelp.org/reference/wilbur/head/meta.html>. You will be using the `<META HTTP-EQUIV="refresh" >` tag to create page1.html. Page1.html will redirect the browser to page2.html after three seconds.

2. Use the `<META HTTP-EQUIV="refresh" >` tag to create page2.html. page2.html redirects the browser to page1.html after three seconds.

3. Install page1.html and page2.html on your localhost Web server.

4. Start the Java Spy program if you would like to see the Web browser and Web server communications.

5. Point your Web browser at <http://127.0.0.1/page1.html> to begin the infinite redirection loop.
   (Do not refer to the page as <http://localhost/page1.html> if you are using the Java Spy program.)

Answer the following question to test what you have learned about URL redirection and dividing Web documents among several Web servers.
1 What are some of the problems and disadvantages of using URL redirection and spreading documents among several Web servers to boost performance of a Web service?

**Mirroring**

In the section from Chapter 5 that you read earlier in this unit, you learned that you can boost the performance of the Web service by spreading the load over several Web servers. This increases the number of Web servers from which users can fetch the same set of documents; the document is more available on the Web. There are two ways to distribute the documents among several CPUs:

- dividing up the documents and spreading them among several Web servers (you have just explored this option)
- replicating or mirroring the entire document tree over several Web servers.

*Mirroring* is another tool to provide better availability for popular webpages and software downloads. The Apache HTTP Server Download website provides a set of mirrored copies of the Apache HTTP server for users to choose from. Using mirrored copies spreads the user request load over many Web servers, if the mirrored copies are distributed on networks across the globe, as they are with the Apache download site. Users can gain an added performance advantage by choosing a download site that is close to them. The downloaded software travels the shortest distance over the network from a geographically close mirror site and, therefore, arrives faster. You will learn more about some of the benefits and problems associated with mirroring in the next reading.

**Reading**


In this reading, you should have learned that some management issues arise when mirroring a document collection:

1 The Web server document tree must be replicated for each mirrored copy. The multiple copies of the Web server documents must be kept consistent as the documents change over time.

2 There must be a mechanism for users to find and use the mirrored copies so that the load of user requests is distributed among the mirror sites. This process is called *load-balancing*. 
Let’s examine how a few popular websites handle these issues. The Apache HTTP server download website replicates the releases of the HTTP server code and documentation. When the HTTPd software releases are created a new sub-tree is added to Web document collection. Managing rapidly changing documents, such as those documents found on a news site, is not a problem for the Apache HTTPd server download website because software releases are only generated every few months. The Apache site uses a primitive load-balancing method. The domain names of the mirror sites are displayed on a single page and users are requested to select and use a mirror site. Since compliance to this request is entirely voluntary by the user, this load-balancing method’s effectiveness is highly questionable.

The NCSA Web service, as seen in Figure 5.6, page 188, in the Web Server Technology text, uses sophisticated distributed system tools and does a better job of addressing the replication and load-balancing requirements. The document set on the NCSA Web server changed daily. The challenge was to make managing the multiple copies easy for the administrator and transparent to the user. A distributed file system was used to ease the administrative burden of daily document updates.

The NCSA load balancing technique worked by spreading the load of user’s requests over several Web servers, while providing a clean solution to the management of the individual Web server names. Users accessed the set of Web servers by the original website’s domain name, <www.ncsa.uiuc.edu> and used round-robin DNS to map one domain name to many IP addresses. How can one domain name apply to several Web servers? If you use nslookup to find the IP address for a large website you will find that it is fairly common for one domain name to be mapped to several IP addresses. For example an nslookup of cnn.com reveals that the domain name maps to eight IP addresses:

64.236.16.20, 64.236.16.52, 64.236.16.84, 64.236.16.116, 64.236.24.4, 64.236.24.12, 64.236.24.20, 64.236.24.28

cnn.com could be using round-robin DNS, just like the NCSA server, for load-balancing. Given a single domain name round-robin DNS returns IP addresses in a simple rotation. For example:

When user #1 requests cnn.com she would receive 64.236.16.20 from DNS.
When user #2 requests cnn.com she would receive 64.236.16.52 from DNS.
When user #3 requests cnn.com she would receive 64.236.16.84 from DNS, and so on, until user #9 would begin the rotation again by receiving 64.236.16.20 from DNS.

Using round-robin DNS to map a domain name to several IP addresses solves two naming problems that arise when a Web server is scaled (expanded):
1. It distributes the load across mirrored websites; and

2. URLs in ‘bookmarks’ or in hypertext links that still point to the original website do not need to be modified.

cnn.com could be using a network appliance for load-balancing. A network appliance is a dedicated one-function unit situated on the network. Just as toasters or radios or televisions are appliances performing one specialized function, network appliances are appliances performing one specialized function: in this case, load balancing. A load balancing network appliance performs the same domain name-to-address translation function as round-robin DNS: mapping one domain name to many IP addresses. The difference between DNS and a load balancing network appliance is the level where the load balancing logic does its work. DNS is placed at the application level. Applications can call DNS functions and DNS applications like nslookup to translate names to addresses. Network appliance load balancing functions are buried deep in the network layer, the low-level component responsible for routing or directing packets of data in the TCP/IP data stream. An application cannot ‘see’ or call functions to direct TCP/IP packets across networks, and they cannot call functions to direct the load balancing among sets of network addresses. The function a network appliance performs is transparent to applications using them.

Figure 5.6 (p. 188) illustrates the most common Web server architecture in which companies use load balancing: all the Web servers in the scalable Web service exist on the same local area network. For international companies with networks that span continents, a Web server/load balancing architecture as seen below is also possible.

![Network Architecture Diagram](image)

**Figure 7.1**

In this architecture the DNS/load balancing support and Web servers are on a LAN in Hong Kong. The load balancer can also direct incoming HTTP traffic to a Web server on a different network in London. This configuration creates a virtual Web service that spans all three Web
servers. If a Web client with a company business unit in France wanted
to download large images from the virtual Web service, the load
balancer may be able to look at the Web client’s domain name,
determine that she is in Europe, and redirect her request to the London
Web server.

You should now understand the concepts of mirroring and load
balancing. Answer the following question to test your knowledge.

**Self-test 7.2**

1. Explain some of the disadvantages and problems associated with
   mirroring a website.

In this section you have reviewed document distribution, mirroring and
load balancing as a way of boosting a website’s performance by making
available several copies of a Web document.
Caching Web information

Mirroring is one of several mechanisms where multiple copies of Web documents exist on the Web: caches are everywhere on the Internet. You are probably using a cache at this very moment and are not even aware of it. This section of the unit explores the fundamentals of caching on the Web.

The following reading will introduce you to caching. As you read, try to guess what type of cache you are using.

Reading

*Web Server Technology*, Chapter 5, Section 5.3, 5.3.1, pages 190–95.

(Note, you are not required to read Section 5.2, p. 189.)

In the reading you learned that the major motivation for caching is to increase performance and efficiency of the Web in the following ways:

1 **Reducing latency.** Caching reduces the time it takes for clients to retrieve Web documents that reside on a Web server. Cached documents are closer to the client on the network and are therefore faster to retrieve. Caching a document locally is beneficial both to the user in terms of reducing the time to download a document and to the network in reducing bandwidth consumption. Performance increases due to caching are especially evident when documents are frequently downloaded (very popular) from a distant server or the document is very large and consumes a lot of network bandwidth.

2 **Reducing the server load.** Caching reduces the number of requests made to an originating Web server. The cache acts on behalf of the Web server in delivering documents to the user.

3 **Reducing the load on the network (bandwidth consumption)** With caching there is the potential for fewer requests and responses that travel over the network.

The basic idea of a cache is to store documents locally and then when Web document requests are made, short-circuit the request and check to see if the document is available locally instead. From the reading, have you guessed what cache you are likely to be using at this time? Try this quick experiment: fetch a series of static webpages from one of your favourite Internet sites. Disconnect your computer from the Internet. (If you dial-up to an Internet Service Provider, hang-up the modem connection.) Select the ‘Back’ button on your browser, to revisit a URL you just visited and then select the ‘Forward’ button. You should be able to see the webpages you just visited when you were connected to the Internet. How is it possible to view a webpage if your computer is not connected to the Internet? This is your browser’s cache at work.
The Web browser cache is an area on the computer’s hard disk (and in the memory) where the browser manages and stores files it retrieves from Web servers on the Internet. If the user requests the same document more than once, the browser may return to the user the document in the cache rather than fetching the document again from the Internet. Retrieving documents from a browser’s cache is much faster than fetching documents from the Internet because the cached documents are on the local computer’s hard disk. In the next activity you will learn how the browser’s cache works.

**Activity 7.4**

In this experiment you will learn how to adjust your Web browser cache settings, view the contents of the browser cache, turn off or ‘zero-out’ the browser cache, activate your browser cache, and witness how your browser cache stores documents as you make Web requests.

1. **How do I adjust the cache?**

   Settings for your Netscape browser’s cache may be adjusted through the following graphical user interface:
   - **Edit -> Preferences -> Advanced -> Cache**
   - You should see a menu similar to this:

   ![Figure 7.2](image)

   **Figure 7.2**
Settings for Internet Explorer’s browser cache may be adjusted through the following graphical user interface:
Tools -> Internet Options -> General -> Temporary Internet Files -> Settings
You should see a menu similar to this:

![Browser Cache Settings](image)

**Figure 7.3**

You can specify how the browser will use its cache through the Check for newer version of stored pages and Compare the page in the cache to the page on the network settings. You will learn how this works later on in the unit. At this time select one of the following settings:
Once per session, Automatically, or Every time you start Internet Explorer.

You can specify the size of the browser’s disk cache with the Disk space and Amount of disk space to use settings. A large cache will store and manage a lot of files for a long period of Web browsing; a small cache can only store a few files over a short period of Web browsing.
You should see that the size of your browser cache is greater than 0. If it is not, set the value to a number greater than 0, and browse around a bit on the Web before returning to this exercise.

The Netscape browser allows you to specify the amount of memory the browser cache will use; IExplorer does not permit you to adjust this quantity. In general, the Netscape browser offers greater control over browser cache settings so you may want to choose to use Netscape for experiments in this unit.

2  **Where is the browser cache?**

The location of the browser cache depends on the installation directory of the browser root, `${BROWSER_ROOTDIR}`
You can use the Explorer ‘Tools->Find->Files or Folders’ and search for ‘Cache’ or ‘Temporary Internet Files’ to find the browser cache.

In the example above, IExplorer’s cache is at:

C:\WINNT\Profiles\Administartor\Temporary Internet Files\
The Netscape browser cache is located at:

C:\WINNT\Profiles\{username}\Application\Data\Mozilla\Users50\default\{username}\Cache (for Netscape 6)
or
$\{BROWSER_ROOTDIR\}\Users\{username}\Cache (for Netscape 4)

3 View the contents of the browser cache

The browser’s cache is a protected directory that is managed by the browser application. The browser determines the way the files in the cache are organized and named. The cache may be organized as a single directory or as a tree directory structure. You should not attempt to directly view files in the cache since the browser is actively managing and protecting this directory. However, it is safe to copy cache files into another directory and then view the copy with a Web browser.

To see the contents of the Internet Explorer browser cache use:
Tools -> Internet Options -> General -> Temporary Internet Files ->
Settings -> View Files

The files in the cache are organized in a single directory and named according to their URLs. The cache should look similar to this:

![Temporary Internet Files](image)

Select and copy a few of the cache files to a new directory and view the copies with a Web browser. You should recognize the images and documents as files you have already downloaded from the Web.

The Netscape (version 6) browser cache is organized as a tree directory structure and named according to the browser’s own internal naming convention. View the cache with Explorer by opening the directory:

C:\WINNT\Profiles\{username}\Application\Data\Mozilla\Users50\default\{username}\Cache

Figure 7.4
The contents of the cache should look similar to this:

![Cache Content](image1)

**Figure 7.5**

An easy way to view the entire contents of a tree-structured cache is to use the Explorer Find Tool. With Explorer select ‘Tools->Find->Files or Folders’ and search for “” (any filename). The cache should look similar to this:

![Explorer Find Tool](image2)

**Figure 7.6**

Individual cache files are named ‘01000000’, ‘01010020’, etc. You can also use the Explorer Find Tool to view changes in the cache’s contents over time. Simply select ‘Find Now’ to see an updated view of the cache contents.

4 Turning off the browser cache

For some experiments in this unit it is necessary to turn off the browser caching feature or clean out the cache of old cache entries. Netscape offers you better control in disabling the browser cache, so it is recommended that you use Netscape in these experiments. The Netscape browser cache can be turned-off by setting the size of disk cache to 0. Setting the disk cache to size 0 and then resetting the disk cache to a size greater than 0 will ‘clean out’ all the current browser cache entries.
Follow these instructions to turn-off the Netscape cache:

1. Edit -> Preferences -> Advanced -> Cache
2. Set Memory Cache to size 0
3. Set Disk Cache to size 0
4. Select ‘Clear Memory Cache’
5. Select ‘Clear Disk Cache’
6. Fetch a static webpage from the Internet
7. Disconnect your network connection to the Internet. If you dial-up to an Internet Service Provider, hang-up the modem connection.
8. Try to refetch the same webpage. You should get an error. If you can see the webpage then your Web browser cache is still turned on and you have not disabled your browser correctly.

5. **See the browser cache in action.** To see the browser cache in action:

1. Clear out the browser cache by setting the disk cache size to 0 and then resetting the value to a value greater than 0.
2. Use the Explorer Find Tool to view the contents of the cache. It should show that the cache is empty.
3. Fetch a static webpage from the Internet
4. Select ‘Find Now’ with the Explorer Find Tool see an updated view of the cache contents. You should see a few new files that your cache has stored from your Web requests.
5. Select and copy a few of the cache files to a new directory and view the copies with a Web browser. You should recognize the images and documents as files you have just downloaded from the Web.

You have just witnessed your browser cache at work!

Caches that reside on the Web browser’s local disk are just one type of Web cache. Many Web caches reside on the local area networks (LANs) of Internet service providers and major corporations. Web servers use LAN caches and they can also maintain their own auxiliary caches. For example, set your browser cache to 0 and then navigate to <http://abcnews.com>, watching the bottom line of your browser that displays the URL that is loading. You may be able to see that some images or text on the HTML page is downloading from <http://a8.g.akamaitech.net>, a commercial cache. To witness another Web server cache, complete a search for a popular homepage with the search engine, <http://www.google.com>. You will see under the result an
entry entitled ‘Cached’. If you select ‘Cached’ you will receive a snapshot of the homepage from the Google Web server’s cache. The copy you receive is a snapshot saved to the Google website’s cache the last time the Google robot traversed that Web server.

Most caches act as proxies — an agent representing the client to the server as well as representing the server to the client. Proxies act as a mediator, a go-between for the client and the server.

Most Internet caches are placed at sites where there is a serious network bandwidth load, like the link between the local area network and the Internet (see Figure 5.8, Figure 5.9, pp. 191 and 192 in Web Server Technology). These are LAN Web caching proxies and they are especially beneficial for users in large organizations requesting documents from many transcontinental Web servers. In large organizations many users frequently request the same documents. If users can download a local cached copy of the document from the cache their document retrieval will be much faster than fetching the document from the original Web server. This performance boost will be greatly increased if the originating Web server lies on a distant network where document download may take a minute or more.

Answer the following self-test questions to determine if you understand why caches are useful on the Web.

**Self-test 7.3**

1. Name three reasons for using Web caches.

2. Where do you think the OUHK LAN Web proxy cache is located on the network? Why? Why is a Web proxy cache useful for OUHK LAN users?

In the next reading you will learn how a cache operates as you examine a LAN Web caching proxy.

**Reading**

*Web Server Technology*, Chapter 5, Section 5.3.1, pages 195–203.

In the reading you learned that Web browsers must be redirected to use a LAN Web proxy server. The proxy, by definition, acts as a Web server to fulfil requests from the Web clients and as a Web client to make requests of distant Web servers. The proxy does four things:

1. Receives requests from clients.
2. Serves documents from its cache if possible.
3. Fetches documents if they are not in the cache. The cache may fetch documents from Web servers or other caching servers, as you will see in later sections.
4. Manages the cache documents. The proxy cache decides which document to remove from the cache if the cache is full and there is a new incoming document waiting to be stored.

You will now set up a Web caching proxy on your localhost server to witness this sequence of steps. Your test Web server operates on a very limited localhost network. To understand the benefits of a LAN Web caching proxy you should imagine during this next activity that your Web server exists on a busy LAN and is used by many Web browsers on the LAN.

**Activity 7.5**

**Experiments with the Web server as a caching proxy**

In this exercise you will configure your Web server to also act as LAN Web caching proxy, configure your Web browser to use the proxy, and then discover how the proxy works to cache Web files.

1. **Configure the Web server as a proxy**

   1. Some proxy caches do not cache domain name mappings to IP addresses. In this experiment you will use URLs containing IP addresses of Web services so that you can witness how the proxy cache works without any influences due to DNS.

   1. Find the IP of address of any Web server you choose that serves static HTML pages (not a CGI service). For example, you may choose `<http://hdf.ncsa.uiuc.edu>`.

   2. Open a MSCMD window and use nslookup to call DNS to resolve the domain name hdf.ncsa.uiuc.edu to an IP address:

      ```
      > nslookup hdf.ncsa.uiuc.edu
      Name: hdf.ncsa.uiuc.edu
      Address: 141.142.2.44
      ```

   3. You will use http://141.142.2.44/ as the URL in this experiment.
Edit the Web server configuration file, httpd.conf, and uncomment the following configuration directives. Instructions to you are noted inside parantheses "()")

(Uncomment these modules so they are dynamically loaded by the Apache Web server)
LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
LoadModule proxy_module modules/mod_proxy.so

(Uncomment to allow the proxy to cache documents)
CacheNegotiatedDocs

(Uncomment these to turn the Expire headers on)
ExpiresActive on
ExpiresDefault M60

(Insert this log file format so your proxy will log the relevant HTTP header values)
LogFormat "%h %l %u %t "%r" %>s %b "%{User-Agent}i" "%{Pragma}i" "%{Cache-Control}i" "%{Expires}i" "%{Via}i" "%{ETag}i" "%{Refresh}i" " debug

(Turn on the custom logging capability)
CustomLog logs/access.log debug

(Uncomment to enable the proxy server. IMPORTANT!: Change the "Order" and "Allow" parameters as below so that your local Web browser can use the proxy)
<IfModule mod_proxy.c>
ProxyRequests On
<Directory proxy:*> Order allow,deny Allow from all
</Directory>

ProxyVia On
(Modify here to indicate your own Apache proxy root directory, ${PROXYROOT_DIR})
CacheRoot "C:/Program Files/Apache Group/Apache/proxy"

(Uncomment these cache parameter defaults without modifying the values)
CacheSize 5
CacheGcInterval 4
CacheMaxExpire 1
CacheLastModifiedFactor 0.1
CacheDefaultExpire 2
3 Stop and then start your Web server to activate the Web server as a caching proxy. (It may not be sufficient to select ‘restart’ to activate your proxy on a running Web server.)

2 Configure the Web browser to Use the Proxy, Turn-off Web Browser Caching

For Netscape

1 Edit-> Preference->Advanced->Proxies

2 Select ‘Manual proxy configuration’. This directs your browser to first look for a document in the proxy cache before fetching the document from a Web server on the Internet.

3 Set the HTTP proxy value to 127.0.0.1 and the port value to 80 or whatever port your localhost Web server is listening on. Select ‘OK’.

4 Turn off Web browser caching.

3 See the proxy cache in action

1 Turn off Web browser caching.

2 Fetch a few webpages from the Internet, beginning with the URL containing an IP address. For example, <http://141.142.2.44>.

3 Disconnect your computer from the Internet. If you dial-up to an Internet Service Provider, hang-up the modem connection.

4 Select the ‘Back’ button on your browser, to revisit a URL you just visited and then select the ‘Forward’ button. You should see the webpages you just visited; they are in the proxy cache. This is your proxy cache at work!

5 Open the proxy root directory, ${PROXYROOT_DIR} and you will see that the proxy cache is organized as a tree structure just like the Web browser cache. The proxy cache manages that structure and internal naming of the cache files. Use the Explorer Find Tool to look for the newly cached files. For example, the cached HTML file from <http://141.142.2.44> was found in

$PROXYROOT/m/1/i/41fabrm2v5bi0ffrmxc4gzg

When viewed with a word processing application the cached file looks like the file ‘cachehdfhome.txt’ (please go to the course website for this file).
The GIF image on the homepage was located at:

$PROXYROOT/r/b/2/nhvurj35lt4oonrnnrh0pevo

and looks like the file ‘cachehdimage.txt’ (again on the course website).

Notice that the HTML and GIF file in the cache can no longer be displayed as normal HTML and GIF files with a Web browser because the proxy cache has written some additional data at the top of each file to help it manage the files in the cache. The proxy cache is caching the HTTP header files sent from the original Web server along with the HTML and GIF files.

6 Re-enable your Web browser cache and refetch the page from <http://141.142.2.44> (or whatever file you are using). Then disable your Internet connection and re-fetch the same page. Which cache is returning the page to you? Is it the browser cache or the proxy cache? Identifying the correct copy of the Web document and the correct source of the document is a common problem when using multiple cache systems.

Maintaining consistency

Through reading and experimentation you should have learned by now that caches make decisions in managing their store of documents. When a cache receives a user’s request it must determine if the requested document is in the cache, and then decide if it should serve the document from the cache or pass the request along to the originating Web server.

Another important management decision a cache makes is how it keeps its documents up-to-date. The cache must, by some means, detect that a document on the originating Web server has changed and if so, pass the document request along to the originating Web server and replace the ‘stale’ document in the cache with the more up-to-date document version.

The reading explained the following approaches that caching components (Web server, proxy cache, Web browser) could adopt to communicate that a document is stale.

1 The originating Web server could notify all caching servers that the document has changed. This approach is impractical because the Web server is unaware of the locations of all caches storing its documents. Also, the number of caches using a Web server’s resources is potentially so large that it would be impractical for a Web server to notify all caches even if it knew the cache locations.

2 The cache could consult the originating Web server to determine if a document is stale. However, there would be no overall performance gain in using a cache if the cache always consulted the originating Web server. In practice the cache could periodically check with the
originating Web server; but how often is best? This is an open question that the cache must resolve.

3 The cache could store and check the value of the HTTP Expires Entity Header in the HTML document sent by the Web server. This method requires that the creator of the HTML document include meta-information in the HTML document explicitly stating when the document will expire. Meta-information is included with the META tag, similar to the redirection Activity 7.2, the difference being that the META tag now contains expiration information. News websites with rapidly changing documents use this approach, as shown by the following META expires tag found on the New York Times website.

   <title> The New York Times on the Web
   <meta http-equiv="Refresh" content="900">
   <meta http-equiv="Expires" content="now">

4 The proxy cache could send a conditional GET to the Web server and store and check the value of the HTTP Last-Modified Entity Header in the HTML document sent by the Web server. The expiration date can be calculated by the proxy according to a heuristic rule that says if the document has not changed recently, it is not likely to change anytime soon.

5 The caching components could use the HTTP version 1.1 protocol Entity Headers such as Cache-Control and ETag to control document expiration.

While there are several methods for handling the cache consistency problem, no single method has emerged as the established standard for all caching components on the Web today. The non-standard nature of caching on the Web is complicated by the fact that several caches may be involved in a typical Web request/response. Consider, as shown in the following diagram, the caches that interacted to deliver a Web response in your simple test environment. On the Internet many caches may cooperate to deliver information to you.

Caches Interacting to Deliver a Web Response

![Diagram of caches interacting to deliver a Web response](image-url)
For the most efficient operation between caches, it would be best if all cache components involved in the transaction chain could speak the same ‘language’ to communicate when a document is stale. But, the language, the HTTP protocol, is constantly evolving. As the HTTP protocol evolves from HTTP version 1.0 to HTTP version 1.1 the HTTP caching headers and recommended caching methodology change also. It takes time for all the cache component software on the Web to be updated to the new version of HTTP, making it even more difficult for all components to speak the same language.

Currently the Expires Header Field and the If-Modified-Since or Last-Modified header, as shown in Figure 5.17 (p. 202 of Web Server Technology) are the most widely supported methods among cache components for communicating that a document is stale. The HTTP 1.1 Entity Header Fields are not yet widely implemented by all caching components; but, as the Web matures, the HTTP 1.1 headers may eventually replace the Expires Header Field as the major method for controlling cache consistency.

The next activity will demonstrate the way a proxy cache utilized the Expires Header Field and the HTML META tag to manage cache consistency.

---

**Activity 7.6**

1 Disable your browser cache so it is not a factor in this experiment.


3 Create the following HTML file containing the META HTTP-EQUIV tag

```html
<HEAD>
<TITLE> Expires Monday 5:30 PM </TITLE>
<META HTTP-EQUIV="expires" content="Mon, 29 Jul 2002 22:30:00 GMT">
</HEAD>
<BODY> Testing the expires tag
</BODY>
</HTML>
```

4 Modify the Content field to contain an expiration time which is a few minutes in the future. Note that the Web document expiration date must be expressed in Greenwich Mean Time.

5 Install the modified page on your localhost Web server.

6 Stop your Web server and remove log files and proxy files. Start the Web server that has the proxy enabled.

7 Download the HTML document containing the META expires tag with your Web browser. View the page in your proxy cache with the
Explorer Find Tool. The Web server has interpreted the META expire tag and created *Expires* HTTP Entity Headers. What other caching-related Entity Headers do you see?

8 Look at the log file entry and review the HTTP headers and Web server status codes.

9 Change the original document on the Web server in some way, by adding a word, etc.

10 Request the same document again via your Web browser. (Important note: to request the document again, simply select Enter (or Return) after the document name, do not use the Web browser RELOAD button. The RELOAD button issues a HTTP GET with a HTTP Pragma: no-cache Entity Header. This instructs the cache to always go to the originating Web service to fetch the page. This instruction overrules the expiration header and will ruin your experiment.)

11 What happened? Did you see the changed document or the original document?

---

**Online reading 7.1**

If you do not understand the cache consistency concepts covered so far read the following supplementary references. (This is NOT required reading)


2 RFC2616, HTTP/ 1.1, with emphasis on caching headers and status codes, http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

Answer the following self-test questions to see if you have understood the fundamental functions of a cache.

---

**Self-test 7.4**

1 What are the basic steps of a cache operation?

2 What are some of the advantages and disadvantages of a proxy cache using the *Expires* header versus the *If-Modified Since* header to maintain cache consistency?
The effectiveness of caching

So far, this unit has explained the potential performance benefits of caching. But, in practice, how well does caching really work? The following reading will explore this question.

Reading

*Web Server Technology*, Chapter 5, Sections 5.3.2, 5.3.3, 5.3.4, and 5.3.5, pages 203–10.

An efficient, well-managed cache can greatly improve performance as seen by Web users. The performance gain is especially great for accessing large files over great network distances. The use of caching, though, requires extra actions by the proxy that can be thought as ‘overheads’. For example, the proxy cache must locate the document in the cache, copy the Web document into the cache, as well as carrying out the initial document fetch from the originating Web server. If the caching proxy spends too much time completing overhead tasks, it is possible for the addition of a network cache to degrade performance. The cache overheads can be high if there is no document locality, that is, no single document is more popular than any other. In this case, the cache will spend a lot of time fetching documents from the originating Web server and the cache may be of little benefit performance-wise. The key to an efficient cache is maximizing the number of cache hits and minimizing the cache overheads.

Some aspects of a cache’s performance can be improved by tuning, or by making adjustments to the cache’s decision-making process. As you witnessed in Activity 7.1, the most important decision a cache makes is how to handle stale documents. Effective solutions for indicating stale cache entries — such as the use of the META expires tag and HTTP Expires Entity Header — rely on the author of the Web document specifying a document expiration date. This information is not often included with the HTML page, so the Web server is forced to guess when the document should expire. Web servers typically have configuration parameters that can be adjusted to optimize the guessing process. The Apache Web server has the following tunable cache configuration parameters:

1. CacheMaxExpire
2. CacheLastModifiedFactor
3. CacheDefaultExpire

Many Web servers guess based on the time the HTML file was last modified, and that is what CacheLastModifiedFactor parameter does. The CacheMaxExpire and CacheDefaultExpire specify default document expiration values. You can briefly review in the Apache Web Server Reference Manual at <http://localhost/manual/mod/mod_proxy.html> how these configuration parameters affect the cache’s ability to guess which documents are stale.
Compared to the Web caching proxy, the Web browser cache supplies very crude controls to the user for handling document consistency. Both Netscape and IExplorer browsers ask the user to specify how the browser will use the cache, through the Check for newer version of stored pages and Compare the page in the cache to the page on the network settings. The Never setting indicates the browser cache will always return a document to the user if it is in the browser cache. The Every time I view the page and Every visit to the page setting indicates the Web browser cache will always check with the Web server that owns the document to see if there is a newer version of the document before it returns the cached copy to the user. Other settings indicate periodically checking with the original Web server, such as once a session, before using the document in the Web cache.

Another major tunable cache parameter for the browser cache is the size of the cache, which determines how many files are kept in the cache and for how long. This parameter affects the cache replacement policy that decides which file should be removed when the cache is full and a new, incoming file needs to be added to the cache. In the next activity you will witness how cache replacement decisions are made in the Web browser cache.

**Activity 7.7**

The main parameter that a user can adjust with the browser cache is the size of the cache. In this experiment you will see how the browser cache implements policies, based on cache size, that decide what files are kept in the cache and for how long.

1. Clear your browser cache by setting the size to 0 and then to a small size such as 500 Kilobytes.

2. Use the Explorer Find Tool to view the browser cache entries and select ‘Find Now’ to update your view of cache entries as you download files from the Web.

3. Select a website that contains a few large jpeg images, such as an online art gallery. For example, download a full sized image, Image #1, from this collection of Vermeer Paintings at <http://www.cacr.caltech.edu/~roy/vermeer>. Can you find the painting in the browser cache by copying it to another directory and viewing it?

4. Continue downloading Image #2, Image #3, Image #4, and watch the contents of the cache change. At what point does one image file disappear from the cache? Which image file disappears? Did the first image loaded into the cache disappear? Or did the largest image, regardless of age, disappear from the cache?
The image that was selected to be deleted from the cache reflects the decision implemented by the cache’s replacement policy. Answer the following self-test questions to see what you have learned about actual cache performance.

**Self-test 7.5**

1. What factor(s) determine a cache’s effectiveness?

2. Name several types of overheads due to using a cache?

**Hierarchical caching**

The previous section described how multiple caches on the Web cooperate by sharing Web document expiration data so that the most current document can be delivered to the Web user. Some caches are organized in a hierarchy and these caches cooperate for every cache operation. The next reading will describe these cache architectures.

**Reading**


In the reading you learned that there is considerable variation in cache architectures; they can be stand-alone single caches, or caches can be organized in a tree-like hierarchy. Hierarchical-structured caches are typically used by big organizations with many divisions spread across a large geographical area. Each division maintains a local cache which reports to a regional cache. If the local cache does not have a document, it might ask the regional cache for it. This type of cooperation requires a common language, as was explained in the previous section on cache consistency. The HTTP protocol is not well suited to this type of inter-cache communication and several specialized languages have developed that are dedicated solely to this purpose. The Harvest cache uses Internet Caching Protocol, ICP, as an inter-cache communication protocol. The Harvest cache has evolved into a network appliance commercial product, NetCache. Caches can exist as network appliances, just as load balancers have become network appliances. A caching network appliance architecture is shown in the diagram below.
In this architecture, a router redirects the TCP/IP stream to the network appliance cache, which acts like any other proxy, returning Web documents to the user if they are in the cache, and fetching documents from Web servers on behalf of the client. Just as with hierarchical caches, it is possible to have more than one network appliance cache available on a local area network. In this case the network appliance caches use a specialized communication protocol to ‘talk’ to the other caches and the router. For commercial products, the inter-cache communication protocol Web Cache Control Protocol, WCCP, is popular and, indeed, this is one of the languages that the NetCache product supports. WCCP transparently redirects HTTP requests from the requested server to the network appliance cache.

**Self-test 7.6**

1. What are some of the advantages and disadvantages of using hierarchical cache architecture versus a stand-alone cache on a LAN?
Unresolved problems for Web caching

Up to this point, all discussions of caching have addressed static HTML, which are compatible with caching. The next reading will reveal some Web document types that cause problems for caches.

Reading


This unit has described how caches must extend considerable effort to managing rapidly changing documents. Dynamic documents which, for example, display the output of database queries, are potentially ever-changing Web documents and cause significant problems for caches. Some dynamic documents are out-of-date as soon as they are generated; some dynamic documents have a small measurable time-to-live. As a rule most proxies do not cache dynamic documents at all but there is a bit of variability. Most caches apply heuristics to detect dynamic documents and then avoid copying these files into the cache. The cache may search the URL for strings like ‘cgi-bin’ and ‘?’ to detect a dynamic document. Or it may reject any Web document using the POST HTTP method. Some dynamic documents with Last-Modified headers may be cachable by the Apache proxy cache; but this is the exception rather than the rule.

Secure Web documents are another class of document that pose problems for caches. As you will learn in Unit 9, secure documents require permission or authorization, to access. Authorization is typically performed by the Web server to protect the secure document. The user must type in a password, for example, to retrieve the document. If secure documents are stored on a cache without any authorization capability, other users may be able to download the cached document without typing in a password. If the cache does not have authorization capability and does not protect the document in the same way that the Web server does then, clearly, the security of the document is being compromised by storing it on the cache. Caches are not allowed to subvert the Web server’s protection mechanisms so they are not permitted to cache secure documents.

Self-test 7.7

1 What types of documents should not be cached? Why?
Online reading 7.2

If you do not understand caching network appliances read the following supplementary references. (This is not required reading.)

Summary

This unit covered the following tools and techniques for increasing the availability of documents on the Web and reducing the latency Web users experience when downloading Web documents.

1. restructuring a Web document tree and URL redirection
2. mirroring
3. scaling a Web service via load-balancing and replication
4. caching.

From these discussions you should have learned how the Web server imposes and maintains a logical view of the document tree, and some of the motivations for installing a cache. Once a cache has been installed there are important management issues to understand such as the cache placement on the network, tuning, and cache consistency. Dynamically generated Web documents are difficult to manage well with a cache and secure documents shouldn't be cached at all without additional security enhancements to the cache. There is much variety in caching systems. A cache can cooperate with a system of other caches in delivering Web documents or it can be a stand-alone network appliance on a LAN. All these techniques and network architectures have the potential to boost the performance of your website if they are understood and applied correctly.
Feedback to activities

Activity 7.1

1. Create the physical structure in your Web server file system. You may add any HTML files to the directories to use in your testing.

2. Modify your ‘httpd.conf’ configuration file in the following way. For example, if ‘D:/Apache/htdocs/’ is your Web server document root, then add the following lines:

   ```text
   Alias /animals/horses/ “D:/Apache/htdocs/
general_mammals/horses/”
   Alias /animals/moles/ "D:/Apache/htdocs/
insect_eating_mammals/moles/"
   Alias /animals/shrews/ "D:/Apache/htdocs/
insect_eating_mammals/shrews/"
   Alias /animals/whales/ "D:/Apache/htdocs/
general_mammals/whales/"
   ```

3. Restart your Web server.

4. Verify the correctness of the new logical view with your Web browser: <http://localhost/animals>.

Activity 7.2

1. The modified text in port 8080 Web server configuration file, ‘httpd.conf’, should look like this:

   ```text
   Redirect / http://127.0.0.1:80/
   ```

   This configuration entry tells the Apache Web server to redirect traffic for the port 8080 Web server’s document root to the document root of the port 80 Web server.

2. You should have observed that the Web browser initially displayed a URL of http://127.0.0.1:8080/ and then the URL automatically changed to display http://127.0.0.1:80 and the homepage was displayed.

3. You should have seen log entries for the port 8080 Web server similar to this:

   ```text
   127.0.0.1 - - [20/Jun/2001:14:58:36 -0400] “GET / HTTP/1.0” 302 268 “Mozilla/4.61 [en] (WinNT; I)” “-” “-” “-” “-” “-” “-” “-”
   ```

   and log entries for the port 80 Web server similar to this:

   ```text
   ```
The Web server logs indicate that the Web server at 8080 received the GET request and returned a HTTP status field of 302 to the browser, indicating a temporary redirect status. The Web server at port 80 then receives a redirected request for its homepage document, which it sends to the Web browser, along with the GIF image on the homepage. At this point you can assume that the Web server at port 8080 sent to the Web browser the URL of the relocated document and the Web browser then sent the GET request to the URL of the relocated document to the Web server at port 80. You will witness the Web browser’s actions in a later activity.

**Activity 7.3**

1. Here is the HTML source for page1.html. Note the syntax of the META tag.

2. Here is the HTML source for page2.html. Note the syntax of the META tag.

3. You should have seen the browser cycling through requests for page 1 and page 2. The relevant browser HTTP headers, as displayed by the java proxy, would look similar to this:

   GET http://127.0.0.1/page2.html HTTP/1.1
   Host: 127.0.0.1

   GET http://127.0.0.1/page1.html HTTP/1.1
   Host: 127.0.0.1

4. The relevant Web server HTTP headers, as displayed by the java proxy, would look similar to this:

   Date: Thu, 21 Jun 2001 14:11:00 GMT
   Server: Apache/1.3.19 (Win32)
   Last-Modified: Fri, 01 Jun 2001 20:40:26 GMT
   Content-Length: 238
   Content-Type: text/html
   <HTML>
   <HEAD>
   <META HTTP-EQUIV="refresh" content="3; URL=http://127.0.0.1/page2.html">
   <TITLE> PAGE 1
   </HEAD>
   <BODY>
   This is PAGE 1. Redirecting your request to http://127.0.0.1/page2.html in 3 seconds.
   < /BODY>
   < /HTML>

5. The Web server is creating a Refresh HTTP header of the form: Refresh: 3; URL=http://127.0.0.1/page2.html and the browser is receiving, interpreting, and responding to this Refresh header. It is not easy to print out the Refresh header with the available tools; but, you can see the result of the header’s use in this activity.
Activity 7.4
(no extra feedback, description in text is complete)

Activity 7.5
(no extra feedback, description in text is complete)

Activity 7.6

1 If you need help expressing the current time in Greenwich Mean
   Time use My Computer -> Control Panel -> Date/Time-> Time
   Zone. It will tell you that you have to subtract eight hours to express
   Hong Kong time as Greenwich Mean Time.

2 Notice how the proxy has written Expires and Last-Modified HTTP
   Entity Header Fields into the data at the top of the HTML pages in
   the proxy cache:
   Expires: Mon, 25 Jun 2001 01:45:56 GMT
   Last-Modified: Mon, 25 Jun 2001 00:30:56 GMT
   These cache consistency headers help the proxy cache manage the
   time-to-live for this document.

3 If the document retrieval time was after the expiration date, you
   should have seen the new copy of the document displayed in the
   Web browser. The proxy cache should also have fetched the
   modified document and updated its cache. Use the Explorer Find
   tool to locate the updated document in the proxy cache.

4 You may have observed in the Web server logs the Web server
   responding to the proxy server's conditional GET request. If the
   document had not yet expired or had not yet been modified the Web
   server responds with a status message 304 (Not Modified)

5 As you looked at documents in the proxy cache you should have
   seen some HTTP version 1.1 cache-related Entity Headers created
   by the Web server and stored by the caching Web proxy:
   Cache-Control: max-age=48
   ETag: “0-59-3b380574”

Activity 7.7

Observations of the Netscape 6 cache replacement policy indicated that
the full Web browser cache removed the oldest very large file that was
loaded into the cache. Very old, small files remained in the cache for a
long time. The cache's management policy appears to favour deleting
files based on their size over deleting files based on their age.
Feedback to self-tests

Self-test 7.1
Some of the problems and disadvantages of using URL redirection and spreading documents among several Web servers are:

1 When documents are moved from their original location there may be many documents around the Web that point to the original document location (URL).

2 When a URL is redirected, the original Web server must answer the request to redirect the Web browser to the new URL. This causes two problems. First, the original Web server must maintain the outdated URL indefinitely. The second problem is the Web client must pay a performance penalty for redirected requests since the browser must call two Web servers: firstly to be redirected and secondly to receive the requested document.

3 It is difficult to efficiently balance the load of user requests by moving sections of a Web document tree to another Web server, since the demand for popular Web documents is difficult to predict and adjust.

Self-test 7.2
Some of the problems and disadvantages of mirroring a website are:

1 There may be documents around the Web, including ‘bookmarks’ and ‘favourites listings’ that still point to the original copy. Creating a mirror copy is of little value if new user requests are not redirected to the mirrored copy, which distributes the load of user requests among the mirrored copies. How can users find the mirrored copies?

2 Creating a mirrored copy of an entire document tree can require a lot of effort if the document tree is huge. Mirroring can also consume and waste a lot of disk space if a large number of documents in the collection are rarely used. Sometimes due to the size of a document collection it may not be feasible to mirror the entire collection.

3 If mirrored documents change rapidly and there are many mirrored copies, keeping the mirrored copies up-to-date can require a lot of administrative effort.
Self-test 7.3

1 The major reason to use a Web cache is to reduce the time (latency), as seen by the Web user, to download documents from the Web. In terms of the entire Internet, here are three motivations for using caches on the Web:

   1 Reduce latency
   2 Reduce the load on Web servers
   3 Reduce bandwidth consumption

2 The Open University of Hong Kong LAN Web proxy cache is probably located near the network gateway. Positioning the cache at this location is likely to improve overall Web performance, as seen by the LAN users by reducing latency and reducing consumption of network bandwidth. Open University of Hong Kong LAN users probably download a significant number of Web resources from distant Web servers in Europe and the United States, so the performance benefits from caching are significant.

Self-test 7.4

1 The four basic functions of the proxy cache are:

   1 Receive Web requests from clients.
   2 Serve Web documents from its cache if possible.
   3 Fetches documents from Web or caching servers if they are not in the cache.
   4 Manages the cache documents. The proxy cache decides which document to remove from the cache if the cache is full and there is a new incoming document waiting to be stored in the cache.

2 The advantage of the Expires header is that it provides precise accurate information to the cache, as supplied by the document’s author, about the date when the webpage is no longer valid. Therefore, cache users will not get a stale document if the author updates the document accordingly. The disadvantage of the Expires header, though, is that the expiration date is based on the META expires tag, which is often not supplied by the author of the HTML page.

The advantage of the conditional GET and If-Modified Since header is that an expiration date can be applied to a document even if the author did not supply one. The expiration date is calculated by a heuristic rule that says if the document has not changed recently, it is not likely to change anytime soon. The disadvantage of the conditional GET and If-Modified Since headers is that the expiration date may not be accurate and proxy users can still get stale documents if the proxy caching server does not check frequently enough to determine if the document has been recently modified.
Self-test 7.5

1. The goal of an effective cache is to maximize the percentage of cache hits. The biggest performance benefit is gained if the document in the cache is returned to the user with ever consulting the originating Web server. An effective cache would maximize cache hits by selecting the most popular, most requested, Web documents to keep in the cache. A user request pattern for Web documents that is highly localized and a slow-changing document collection would contribute to a cache’s overall effectiveness. The performance gain from a cache that serves documents from a slow, long-distant network would be greater.

2. Each of these actions is considered an overhead due to the caching proxy. The caching proxy must make additional Web requests which incur network delays. The caching proxy must copy the Web document into the cache, locate the document in the cache, and detect and replace stale documents.

Self-test 7.6

1. It is possible for a well-configured hierarchical cache system to improve document retrieval performance over stand-alone caches. A document that is a ‘cache miss’ in a local cache and a ‘cache hit’ in the regional cache, will typically be faster to retrieve than a ‘cache miss’ in a stand-alone cache. However, a request that is both a cache miss in the local and regional cache will take a lot longer to retrieve the document from the originating Web server and return it to the user. If a hierarchical cache is poorly configured and both the regional and local cache experience a lot of ‘cache misses’, the performance for a hierarchical cache can be worse than the performance of a stand-alone cache. Another disadvantage of hierarchical cache systems is their complexity.

Self-test 7.7

1. Secure documents that are protected by access controls on the originating Web server should not be cached because caching compromises the security of the document. Dynamic documents should not be cached because most dynamic documents have a time-to-live of 0; the proxy should always go to the originating Web server to fetch the most current result of the dynamic document.