Crossroads
The ACM Student Magazine

Using Apache as a Caching Proxy Server

by ***Josiah Dykstra***

Fundamentals of a Proxy Server

In the standard web server model a client requests a resource directly from a server and that server sends the resource directly back to the client. An alternate model incorporates a proxy server, an intermediate machine sitting between the server and the client(s), as illustrated by Figure 1.

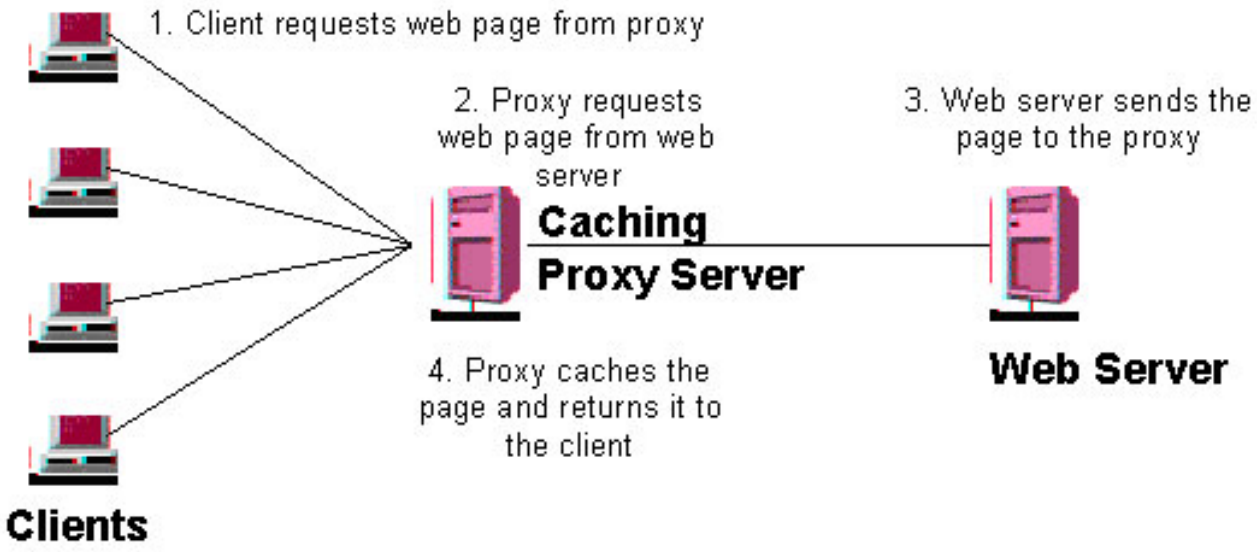


Figure 1: Proxy Setup

The proxy server model offers several several advantages. For instance, the proxy may be combined with a firewall to provide added security to a local area network (LAN). Exposure of the internal network to outside hackers is reduced because machines on the internal network are not directly accessible from machines on the other side of the firewall. However, the proxy assures that access to the Internet by its clients is not hampered.

Second, the proxy can act as a packet filter for incoming web traffic. Because all information first

passes through the proxy before reaching the client, it can be manipulated in useful ways. For example, the proxy can read incoming HTTP (Hyper Text Transfer Protocol, the protocol used on the world wide web) packet requests and discard packets originating from a particular set of domains, such as pornographic sites. The proxy can also reduce ads being displayed on web pages by recognizing the image tag from known ad servers, and filtering the image out of the page before it reaches the client. Additionally, some configurations allow for the removal of unwanted pop-up windows.

Finally, the proxy can cache documents in much the same way as end-user browsers. The proxy may greatly decrease the access time for all clients, by caching the documents of many users. Then allowing the accumulated cache to be available for all users to access. So, instead of only retrieving locally cached documents that a single user has accumulated, each user has access to all users documents.

Finding personally useful information has become one of the major obstacles on the Internet today. One new possibility that I have begun researching is to use pages users have previously viewed (via cache files) to build a search engine. By searching the local copies of pages you already have viewed and perhaps frequently view, there is a higher chance that this search technique may find information most useful to the user. A possible means to this end is to utilize caching proxy servers that end users install and configure to communicate with a central database. This strategy is radically different than the traditional use for a proxy server.

Apache as a Proxy Server

The Apache server is arguably the best web server currently available [**1**]. Its success can be partially attributed to its stability and ability to be easily configured. In addition, the code is open source and freely available. For our purposes, it is advantageous for its proxy module [**2**] that allows each installation of the program to include only those services needed. For example, the module to enable CGI or authentication may be included or excluded depending on each individual installation. Apache can be run on a variety of platforms, including Linux, Solaris, Macintosh and Windows.

Installing the Caching Proxy Server

The following instructions apply to Linux and UNIX type operating systems. The first step is to obtain the source code. This is necessary even if you already have Apache installed, since it is necessary to recompile with the proxy module enabled. The official site is at **http://httpd.apache.org**. Apache version 1.3.20 or newer is recommended. As root, uncompress the file and begin installation.

```
./configure --prefix=/usr/local/apache --enable-module=proxy
```

```
make
make install
```

Many more options are available for configuration; type `configure --help` or see the Linux Documentation Project [3] for more information. Assuming there are no compilation errors, Apache is now installed. Some initial configuration is needed to setup caching. Begin by editing the file /usr/local/apache/conf/httpd.conf. The section we are interested in initially looks like this:

```
#<IfModule mod_proxy.c>
#   ProxyRequests On

#   <Directory proxy:*>
#      Order deny,allow
#      Deny from all
#      Allow from .your_domain.com
#   </Directory>

#   ProxyVia On

#   CacheRoot "/usr/local/apache/proxy"
#   CacheSize 5
#   CacheGcInterval 4
#   CacheMaxExpire 24
#   CacheLastModifiedFactor 0.1
#   CacheDefaultExpire 1
#   NoCache a_domain.com another_domain.edu joes.garage_sale.com

#
```

You will need to uncomment lines you wish to implement, including the `<IfModule>` lines. First, enable the proxy server by uncommenting the `ProxyRequests` directive. In order to restrict the access to your proxy, uncomment the `<Directory proxy:*>` section, and insert your domain where shown. The proxy can add an HTTP Via header to each request with its own name and server version number, see RFC2616 [4] for more information. However, you will probably want to set the `ProxyVia` directive to off, which ensures privacy and security.

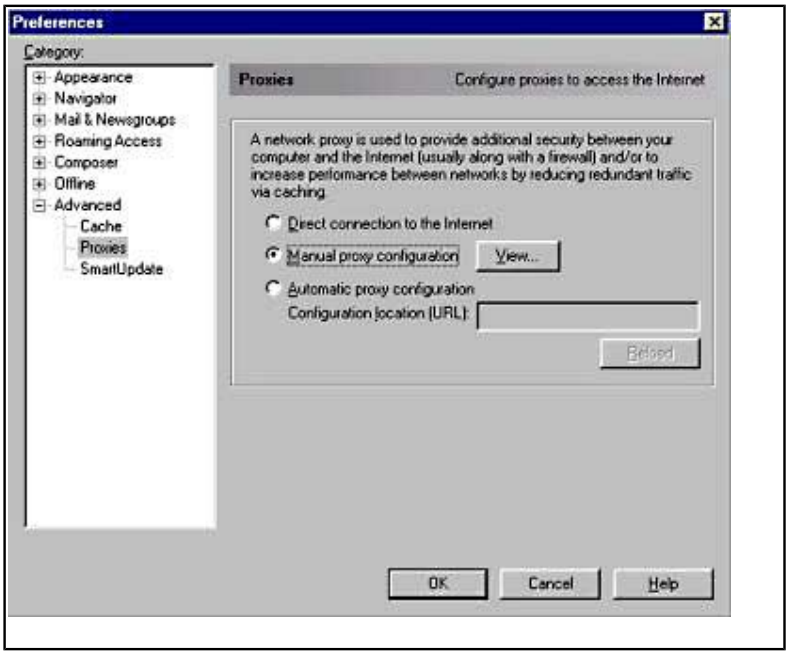
Each of the seven Cache directives, controls how the server handles caching. Setting the

CacheRoot enables caching on the server. This directory must be writable by the user running the server (usually "nobody"). The CacheSize sets the desired space usage in kilobytes. You will probably want to set this higher than the default of 5, based on your available disk space, to allow the greatest number of documents to be stored locally, thus allowing local cache access by the clients. Garbage collection, which enforces the cache size, is set in hours by the CacheGcInterval. If unspecified, the cache size will grow until disk space runs out. CacheMaxExpire specifies the maximum number of hours for which cached documents will be retained without checking the host server. If the origin server for a document did not send an expiry date, then the CacheLastModifiedFactor will be used to estimate one by multiplying the factor by the time the document was last modified. If the protocol used to retrieve a document does not support expiry times (FTP, for example), the CacheDefaultExpire directive specifies the number of hours until it expires. A sample edited configuration file looks like this:

```
<IfModule mod_proxy.c>
  ProxyRequests On

  <Directory proxy:*>
    Order deny,allow
    Deny from all
    Allow from .cs.hope.edu
  </Directory>

  ProxyVia On

  CacheRoot "/usr/local/apache/proxy"
  CacheSize 102400
  CacheGcInterval 8
  CacheMaxExpire 168
  CacheLastModifiedFactor 0.1
  CacheDefaultExpire 72
</IfModule>
```

Configuring Clients

Both Netscape and Internet Explorer can easily be configured to use a proxy server. In Netscape, select Preferences under the Edit menu. Click Advanced, then Proxies (Figure 3). Enable "Manual proxy configuration" and click View. Enter the server name and port in their respective fields (Figure 4).



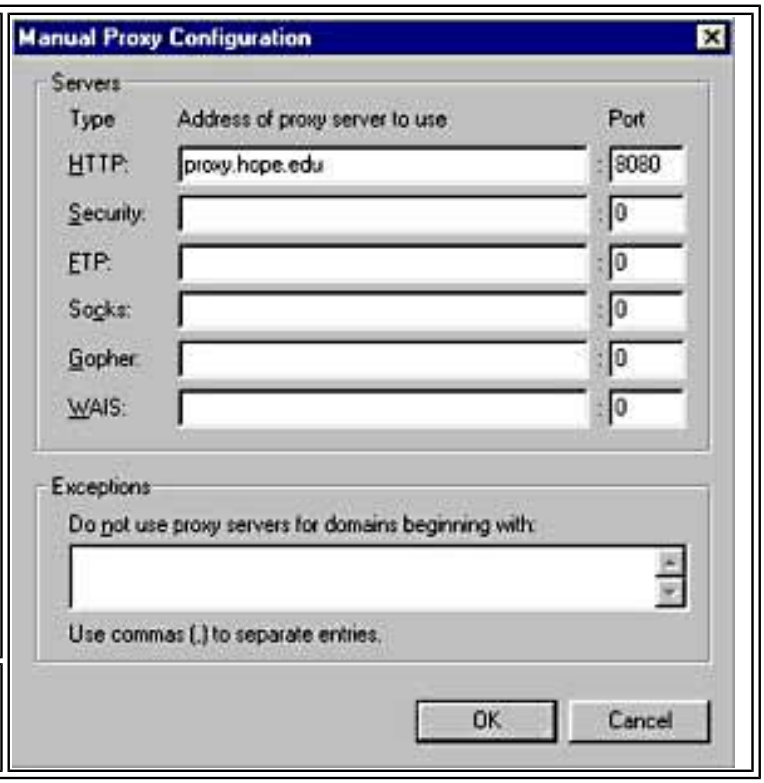
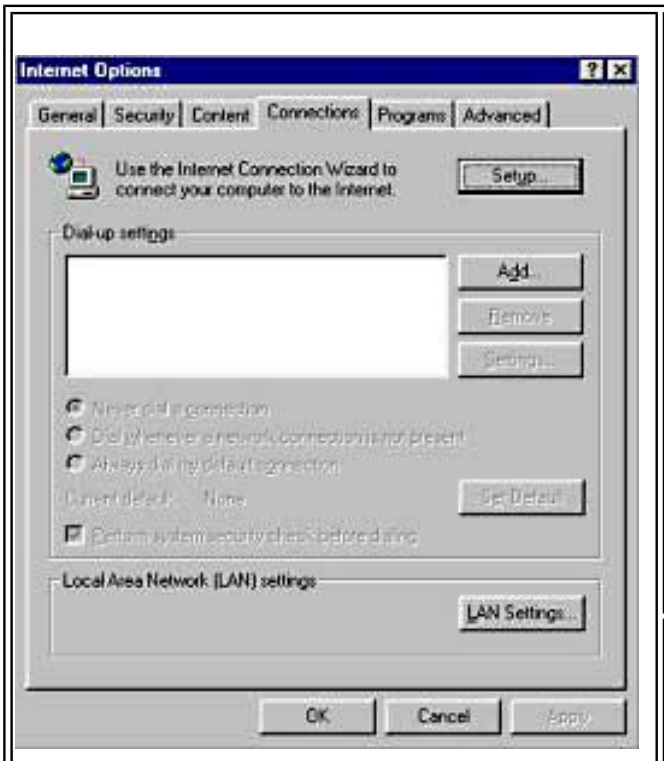


Figure 3: Configuring Netscape, step 2
Figure 4: Configuring Netscape, step 3

Alternatively, in Internet Explorer, select Internet Options from the Tools menu. Click the Connections tab and then LAN Settings (Figure 5). Check "Use a proxy server" and enter the server name and port number as specified (Figure 6).



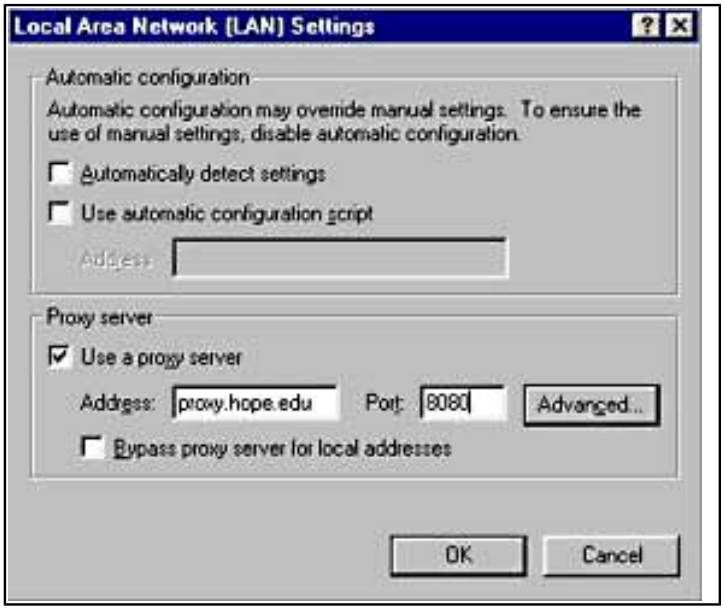


Figure 6: Configuring IE, step 3

Figure 5: Configuring IE, step 2

Conclusions

There are many reasons to implement a proxy server. Network security, content control, and

faster web browsing are just some of the possible benefits. In addition, our research may be able to use the cache of a proxy to make Internet searching personalized and relevant. The Apache server is widely available, easily installed, easily configured and universally accepted as a strong contender for a proxy server. Continued research will make use of these properties and build upon them in new and exciting ways.

References

1

Apache module mod_proxy, *Apache HTTP Server Project,* **http://httpd.apache.org/docs/mod/mod_proxy.html** (10 May 2001).

2

Hypertext Transfer Protocol -- HTTP/1.1, *The World Wide Web Consortium,* June 1999, **http://www.w3.org/Protocols/rfc2616/rfc2616.html** (10 May 2001).

3

Linux WWW HOW-TO: Apache, *Linux Documentation Project,* 8 May 2001, **http://www.linuxdoc.org/HOWTO/WWW-HOWTO-7.html** (10 May 2001).

4

Netcraft Web Server Survey, *Netcraft,* April 2001, **http://www.netcraft.com/survey/** (10 May 2001).

Biography

Josiah Dykstra **dykstra@cs.hope.edu** is a senior computer science and music double major at Hope College in Holland, Michigan. He is a member of IEEE and is currently serving his second term as President of the local ACM chapter. He has held summer internships at Gateway, Inc. and the NSA and conducted research with the NSF Research Experience for Undergraduates.