1. Introduction

1.1. Key to Icons and Notes

1.1.1. Reference Icon

REFERENCE:
This heading is used to highlight additional reference material.

1.1.2. Important Icon

IMPORTANT:
This heading is used to highlight information that is critical to a successful implementation.

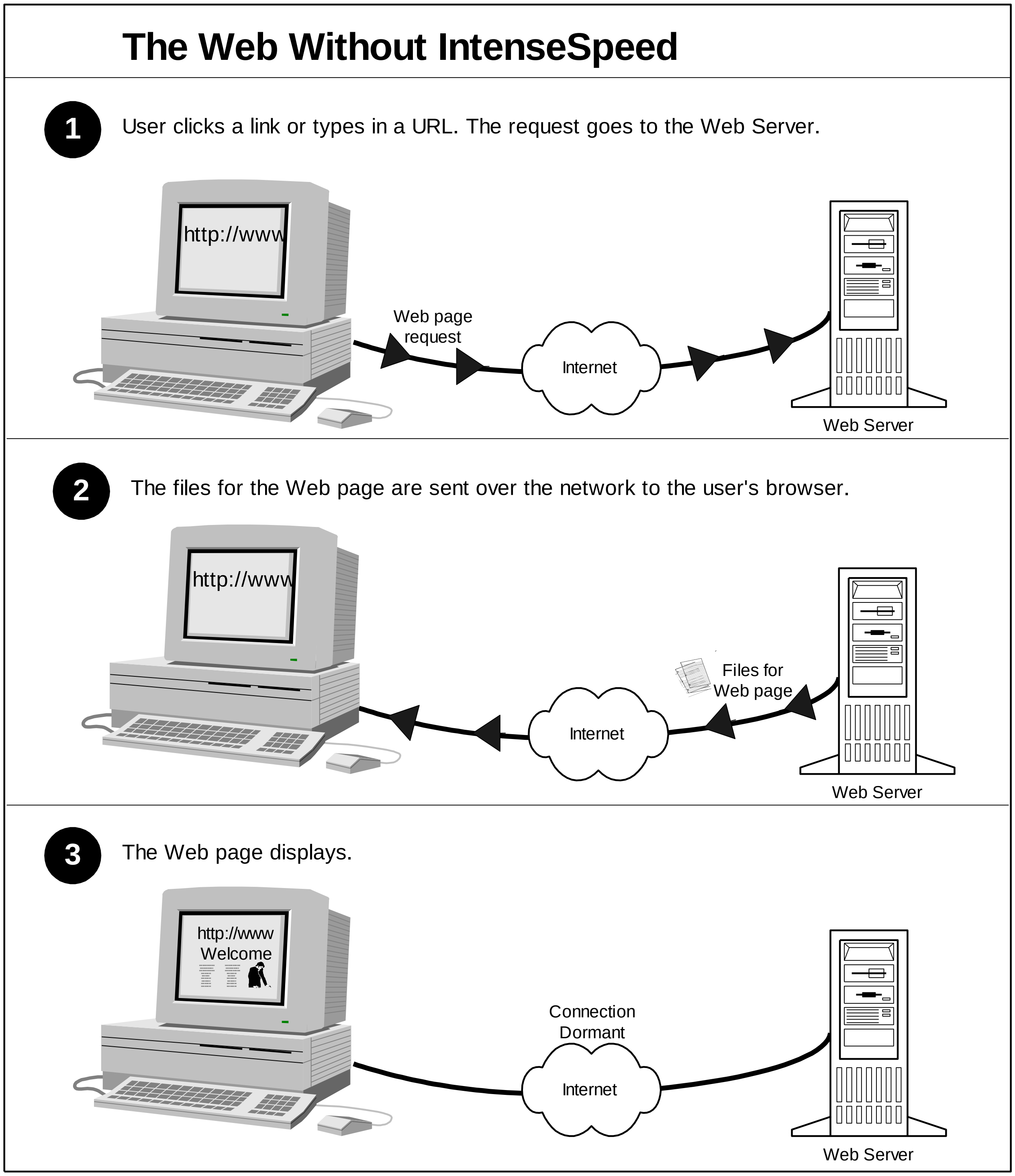
1.1.3. Best Practice Icon

BEST PRACTICE:
This heading is used to denote best practice recommendations.

1.2. How IntenseSpeed Works

IntenseSpeed is software that makes Web pages load faster in a browser, by pre-loSading content into the browser's cache. Normally, when a browser navigates to a Web page, the content for that page is requested from the Web server, the files begin to load, and the page begins to display. The load/display cycle repeats until all the components of a Web page have been completely transferred across the network. These cycles can take a long time if there are many files on the Web page, the files are large, or the network connection is slow or busy.

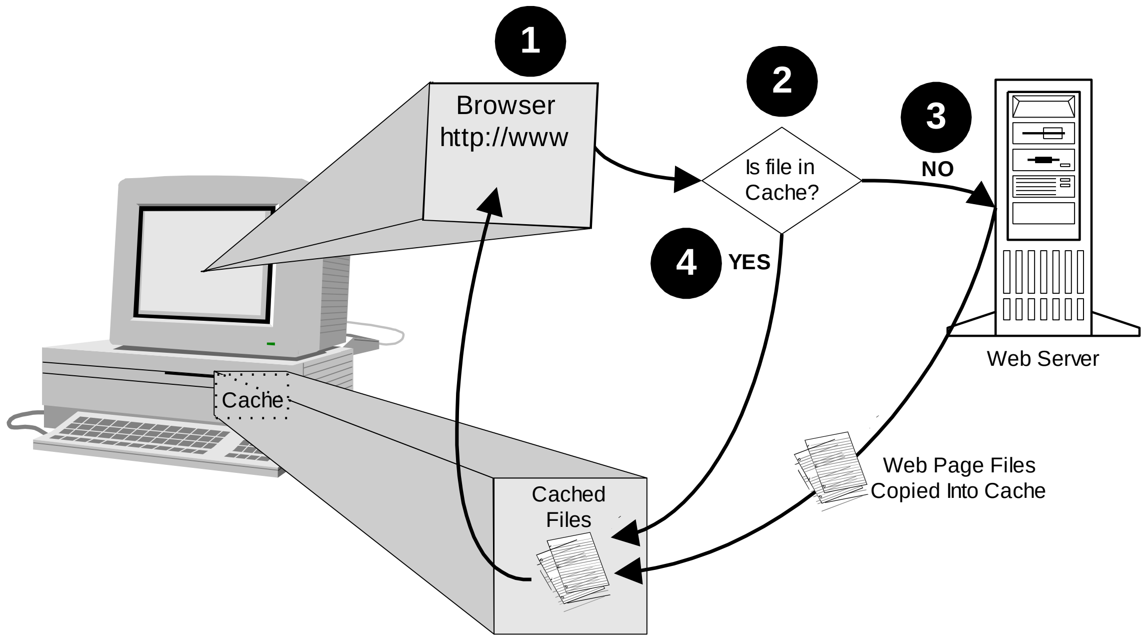
Figure 1-1:The Web Without IntenseSpeed


The Web Without IntenseSpeed
1
User clicks a link or types in a URL. The request goes to the Web Server.
http://www
Web page request
Internet
Web Server
2
The files for the Web page are sent over the network to the user's browser.
http://www
Files for Web page
Internet
Web Server
3
The Web page displays.
http://www
Welcome
Connection Dormant
Internet
Web Server

However, instead of only loading files when they are requested, Web browsers can store files on the user's hard drive at any time during a session. The browser stores the files in a special location on the hard drive (or in the computer's memory) called "the cache". The browser saves the files so that if a user returns to a page recently visited, the browser will use the cached file instead of going back again over the network to re-request the file. This "caching" results in a much faster experience than would occur if the browser had to go back over the network again each time it needed a particular file.

It is always faster for a browser to retrieve a file from its hard drive or memory cache than to traverse the network to obtain the file. If a requested item is not available from cache, the request must travel over the network to the Web server or caching server where it is then retrieved in real-time. Even with a very fast and lightly used connection, this process takes longer than simply displaying content that is stored locally. If an item is not available in cache, congestion on a network can slow the process even more.

Figure 1-2:How A Browser Cache Works



How A Browser Cache Works

1. User types a URL or clicks a link for a Web page.
2. Browser checks to see if files for page are in its cache.
3. If files are not in browser cache, request is sent to the Web Server for the files.
   - Web page files are sent from the Web Server to the browser.
   - Browser stores a copy of each file in its cache and displays the page.
4. If files are already in cache, they are immediately displayed in the Browser.

IntenseSpeed takes advantage of browser caching capabilities to load files for Web pages the user has not yet visited. To load the upcoming files, IntenseSpeed issues the requests for the files before they are actually needed. For example, on a Web site home page, there may be five links. A Web Designer might use IntenseSpeed to load all of the HTML and image files on the

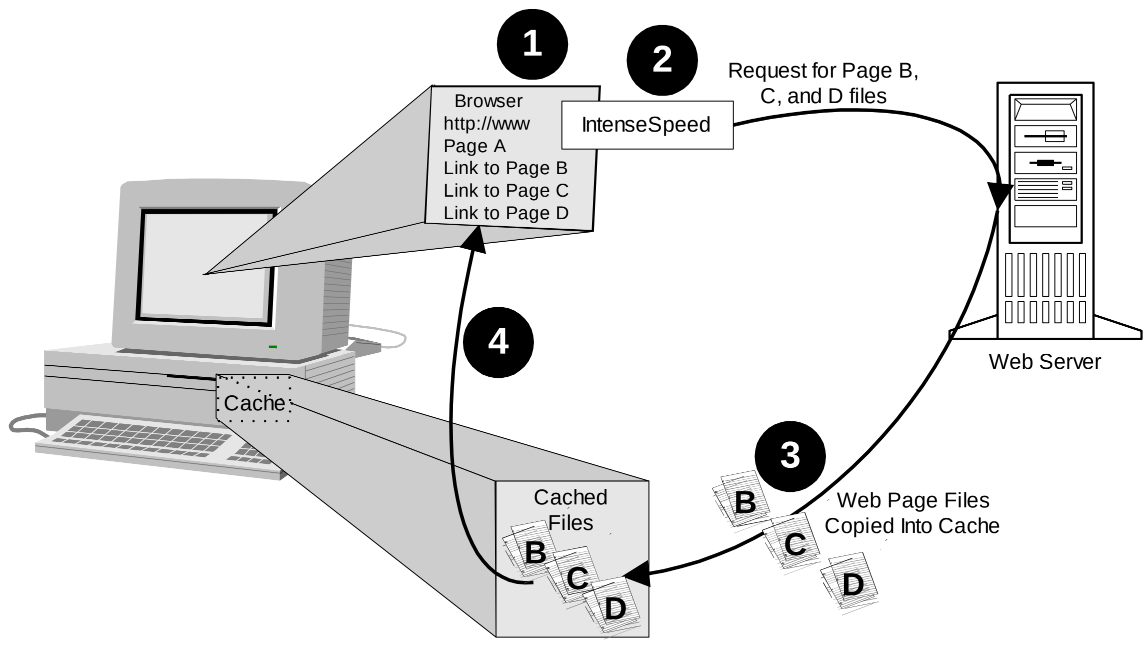
Web pages whose links appear on the home page. In this case, the files for the next page the user visits will most likely already be in the browser cache before the user clicks on the link. When the user does click on a link, the HTML and images for that Web page will most likely already be in the browser's cache. The Web page displays almost instantly, since the browser did not have to go over the network to get the files. On the new page, the Designer can use IntenseSpeed to load all the files needed for the pages its links lead to, and so on.

The amount of time it takes IntenseSpeed to load the requested files depends on the speed of the network connection, just like the loading of normal Web pages. However, by requesting the files before they are needed, IntenseSpeed creates an opportunity for the browser to load the files while the user is experiencing the current Web page. Since files are loaded before they are needed, the load/display cycle is streamlined to just a display cycle.

The time spent on any given Web page depends upon many factors. The user may spend longer on a page if it has audio, a lot of text, or many link choices. The user is also likely to spend more time if it is the first time they have visited the page. Generally, there is plenty of opportunity to load content for upcoming pages while the user is experiencing a page. Because the user is not simply waiting for the page to download, but is actually doing something while the content loads in the background, it does not matter a great deal if the next page's content takes 10 or even 20 seconds to load.

When the concept of a browser cache was originally conceived, it was intended as a place to store frequently accessed files. This is still the case today. However, without IntenseSpeed, the cache is not as smart as it could be. Requests are sent out for content that is already stored in cache and still valid. Even though the browser figures this out based on the Web server's response (“304-Not Modified”), just asking the question can take a lot of time. Also, the cache (without IntenseSpeed) is a dumping ground of Web content. What a user may desire next may or may not be in the cache. With IntenseSpeed, the cache is used intelligently to speed the display of content and store what will be requested next.

Figure 1-3:The Web with IntenseSpeed



The Web With IntenseSpeed

1. Web Page A is displayed in the browser.
2. In the background, IntenseSpeed requests files for pages B, C, and D from the Web Server while the user is looking at Page A.
3. Files for Pages B, C, and D are stored to the browser cache ready to load when user clicks on their links.
4. When user is finished with Page A and clicks a link for another page (Pages B, C, or D) available from Page A, cached files are immediately displayed in the Browser.

1.3. Components of IntenseSpeed

IntenseSpeed is software comprised of two components. The browser (or client) component is required, and the Web server add-in is optional.

1.3.1. IntenseSpeed Client

The client component of IntenseSpeed is loaded into the user's browser as a Web page loads. It is available in several implementations. These include a Java applet, a JavaScript, or a VBScript implementation. Each of these has its own requirements and available functionality. A Web Designer must choose the implementation most appropriate for the Web site.

1.3.2. IntenseSpeed Server Add-In

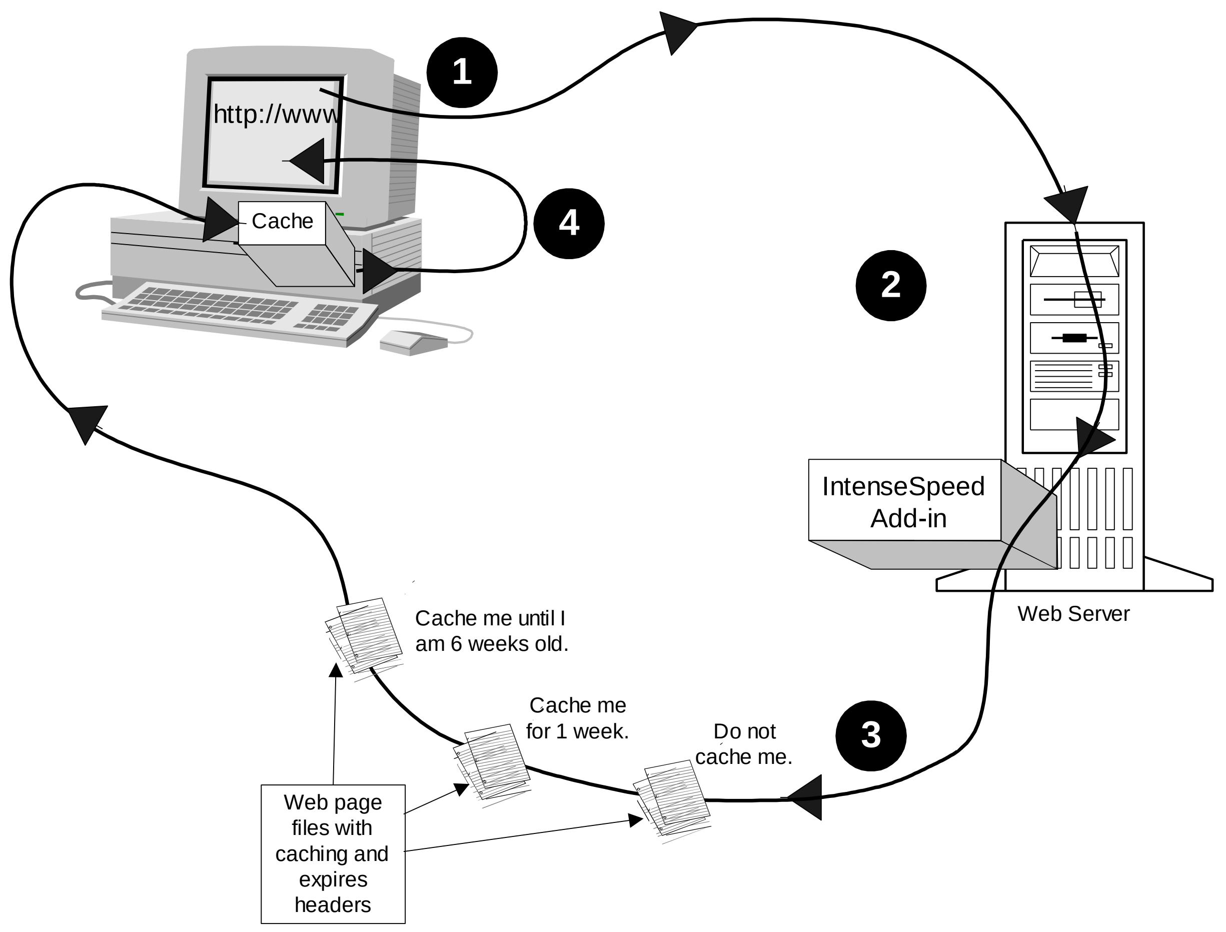
The server component of IntenseSpeed is optional (but highly recommended). This component sets two types of HTTP headers for the files served by the Web server.

- "Expires" header—HTTP headers which specify the expiration date of the Web files.
- "Caching" header—HTTP headers which tell the browser and/or caching server(s) whether or not to store the file in its cache.

Even if a file is in the browser cache, the browser may choose to go back to the Web server to check if the file has been updated since it was cached. The expires headers sent by the IntenseSpeed server add-in allows the browser to determine whether or not a file is up-to-date without going back to the Web server. This improves the performance of the Web site and the browser, while simultaneously decreasing unnecessary traffic back to the Web server to check the freshness of Web content.

Many Web servers have an API (Advanced Programming Interface) which allows developers to build add-in modules for the Web server. IntenseSpeed's server add-in is just such a module. It works with the Apache, Netscape, and Microsoft Web servers.

Figure 1-4: IntenseSpeed Web Server Add-In



Web Server Add-in

1. User clicks a link or types in a URL. The request goes to the Web Server.
2. The IntenseSpeed Web Server Add-in modifies the requested file headers, adding caching and expires information.
3. The Web page files (with caching and expires headers) are sent back to the browser.
4. The browser saves a copy of each file to be cached in its cache, and displays the Web page.

1.4. How Does IntenseSpeed Interact with Caching and Proxy Servers on the Web?

There are many companies today offering caching solutions to increase the efficiency of the Web. Some of these are CacheFlow, Akamai, and Cobalt Networks. Typically, these companies place caching servers at various geographical locations. The intent is that, whenever possible, these servers will cache content for Web sites, giving nearby users a chance to load the content from the caching server instead of from the originating Web server. In theory, this reduces the load on the originating Web server, makes more efficient use of the network, and gives the user a faster response.

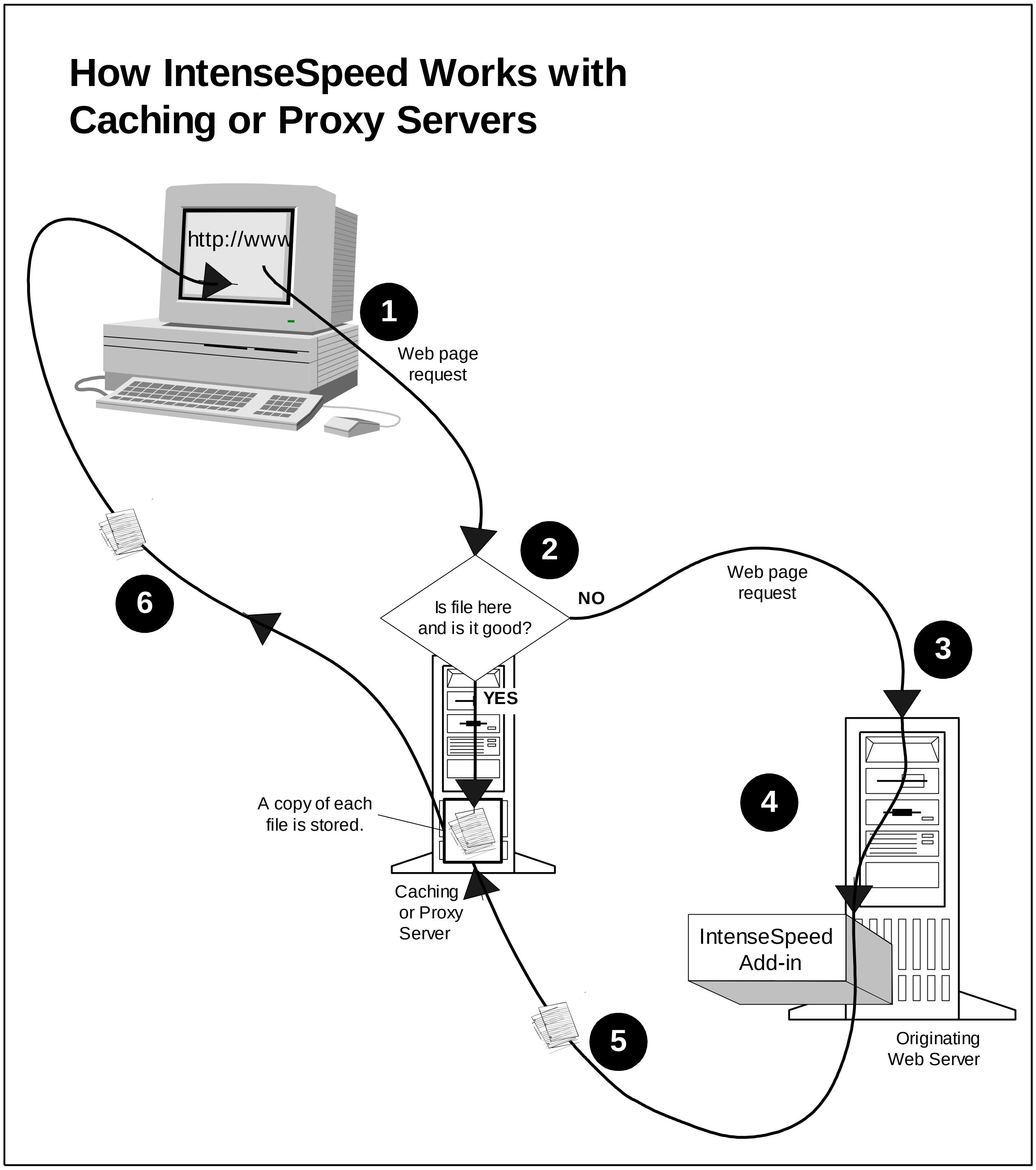
However, caching servers really do need a little help to know when to cache items and when to force a request to go back to the originating server. This can be done through HTTP caching headers (as with IntenseSpeed), which the Web server can insert in front of the content to tell the caching servers whether or not they should cache the files. For example, a dynamically created page that is returned as a result of a search should not be cached. A Web site's banner image normally would be cached, because it would be the same banner image, even if the rest of the content on the Web pages were dynamic.

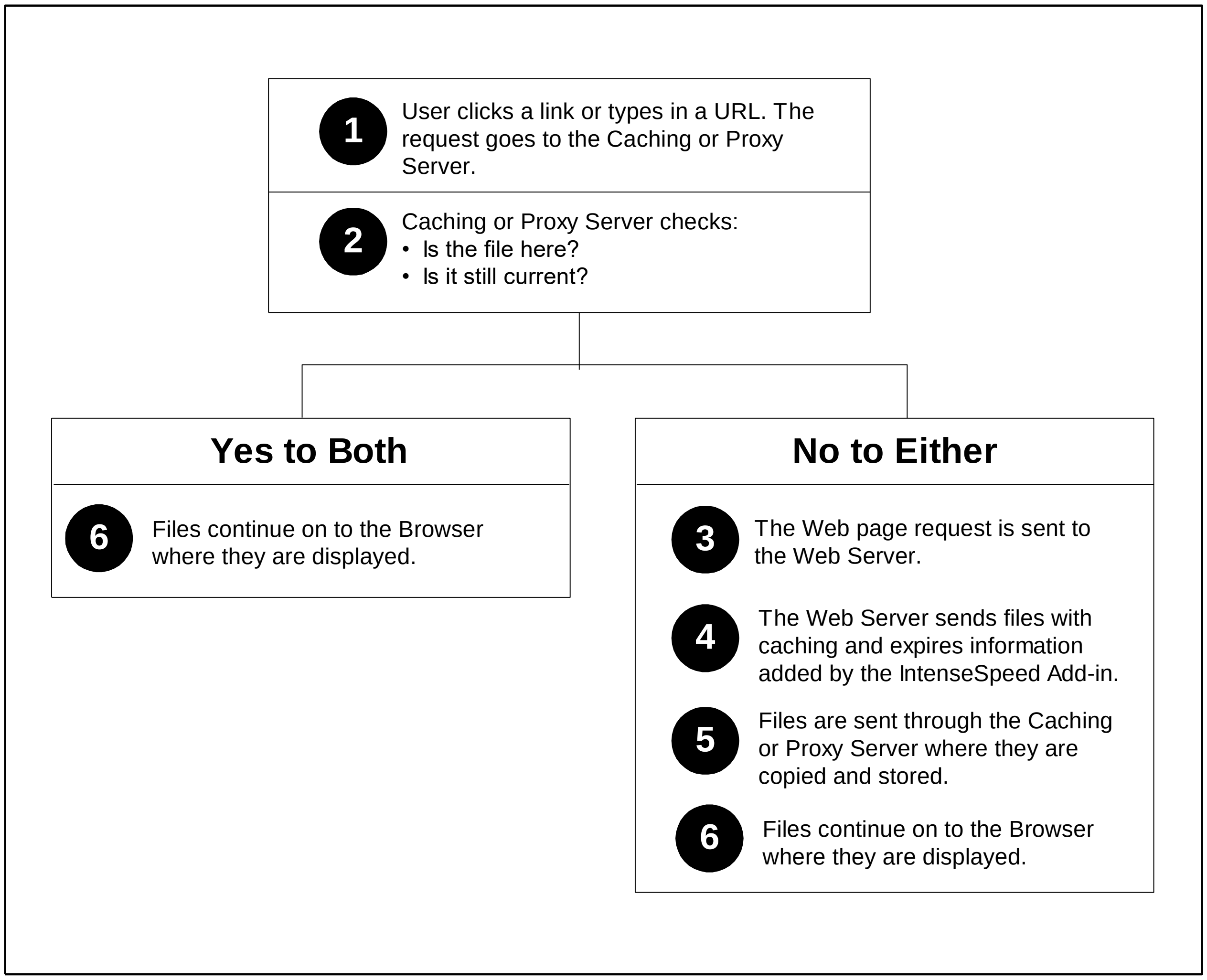
The Web content files also need to contain information that tells the caching servers how long to use them before returning to the originating Web server to determine if they have been updated. If this information is missing from a file's HTTP headers, a caching server might go back to the originating Web server to check the freshness of a file each time it is requested. This makes the caching server much less effective and actually slows the whole process down.

Caching servers use sophisticated methods to determine how long content is good. Some even try to reload or refresh before any request actually arrives. However, these algorithms can never be as good as just simply having Webmasters indicate, through expiration and caching headers, how long content is good. This is exactly what the server component of IntenseSpeed does. IntenseSpeed works well with and can become an indispensable ally of a caching or proxy server.

The optional Web server add-in component of IntenseSpeed adds the expires and caching headers to Web content automatically as the content is served. The Webmaster

Figure 1-5: How IntenseSpeed Works with Caching or Proxy Servers




1 User clicks a link or types in a URL. The request goes to the Caching or Proxy Server.
2 Caching or Proxy Server checks:
• Is the file here?
• Is it still current?
Yes to Both
6 Files continue on to the Browser where they are displayed.
No to Either
3 The Web page request is sent to the Web Server.
4 The Web Server sends files with caching and expires information added by the IntenseSpeed Add-in.
5 Files are sent through the Caching or Proxy Server where they are copied and stored.
6 Files continue on to the Browser where they are displayed.

configures settings in IntenseSpeed to tell it how to set the headers. The settings can be determined by file type, by directory, or both. The files can be set to expire based on any of the following:

- The time of the original request
- Time of the last file modification
- An absolute time determined by the Webmaster

For more information on installing and configuring the IntenseSpeed Web server add-in, see *IntenseSpeed Installation and Configuration Guide*.

An illustration of how IntenseSpeed works with Caching or Proxy Servers starts on page 1-10.