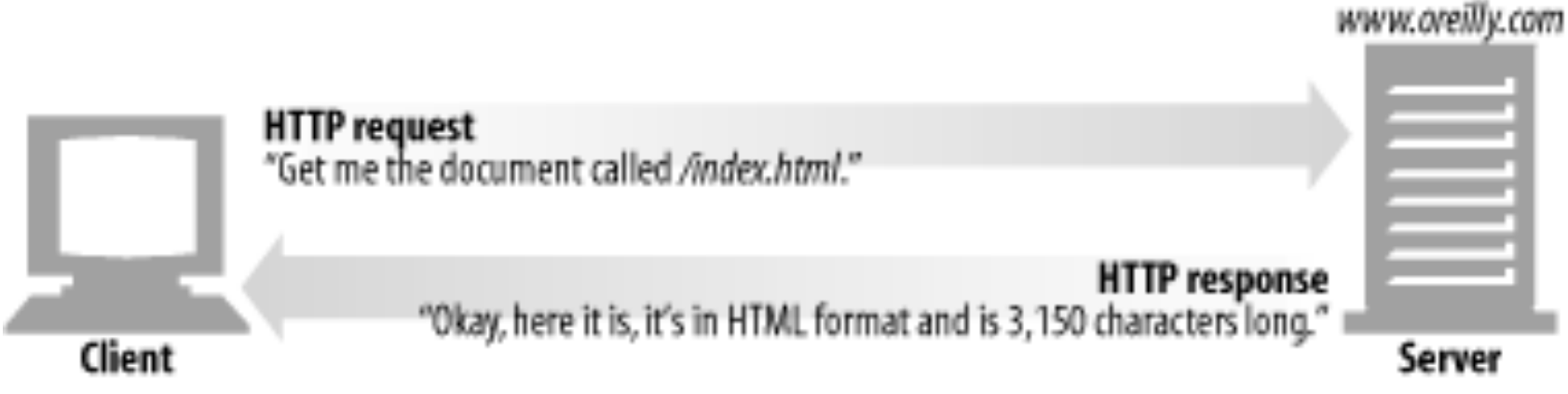
Figure 1-1. Web clients and servers



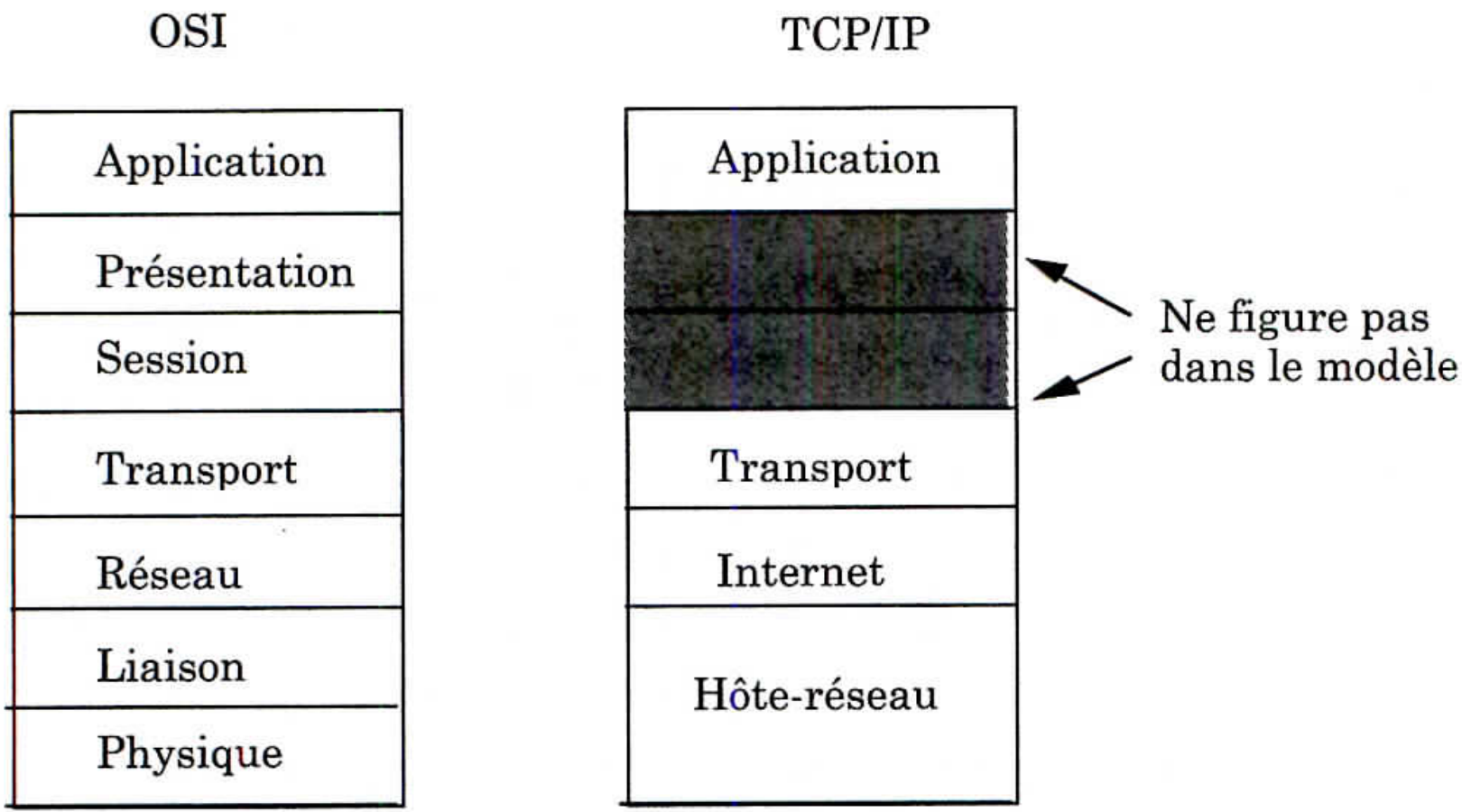


Fig. 1.18 — Le modèle de référence TCP/IP.

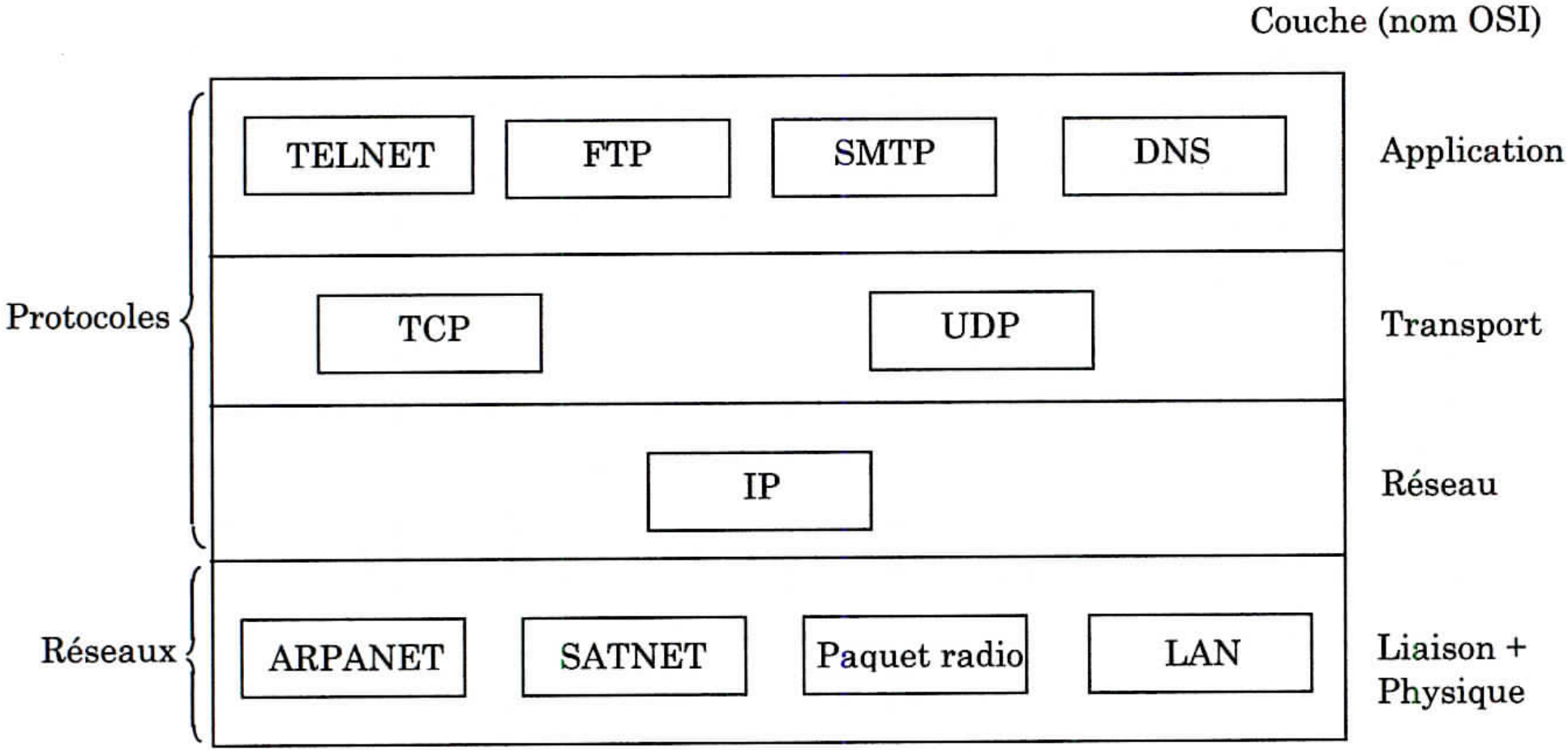


Fig. 1.19 — Protocoles et réseaux dans le modèle TCP/IP initial.

Figure 1-2. A web resource is anything that provides web content

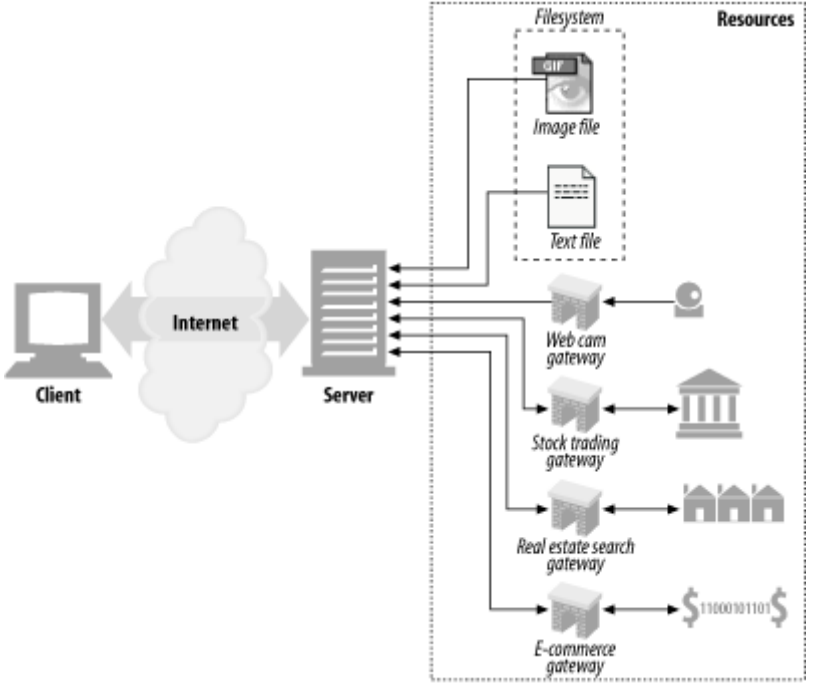


Figure 1-4. URLs specify protocol, server, and local resource

Table 1-1. Example URLs	
URL	**Description**
http://www.oreilly.com/index.html	The home URL for O'Reilly & Associates, Inc.
http://www.yahoo.com/images/logo.gif	The URL for the Yahoo! web site's logo
http://www.joes-hardware.com/inventory-check.cgi?item=12731	The URL for a program that checks if inventory item #12731 is in stock
ftp://joe:tools4u@ftp.joes-hardware.com/locking-pliers.gif	The URL for the *locking-pliers.gif* image file, using password-protected FTP as the access protocol

Figure 1-5. HTTP transactions consist of request and response messages

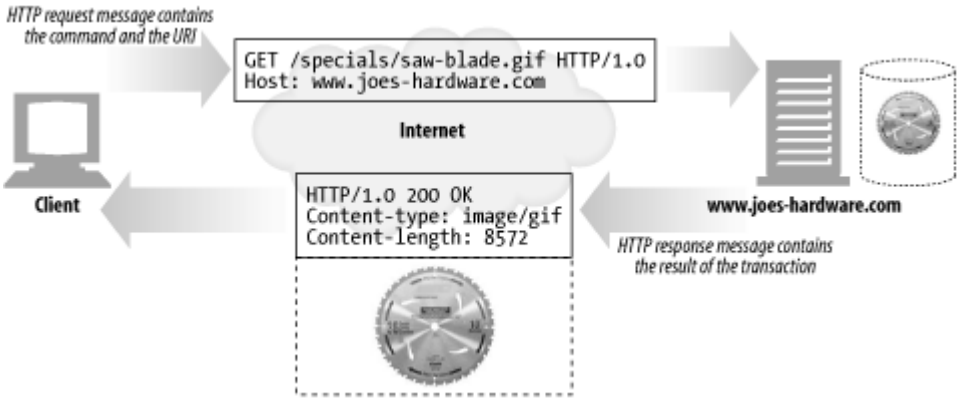


Figure 1-6. Composite web pages require separate HTTP transactions for each embedded resource

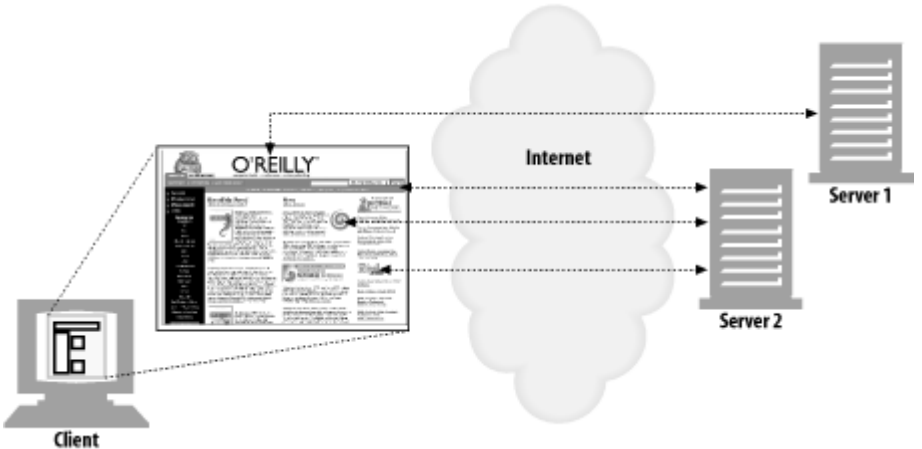
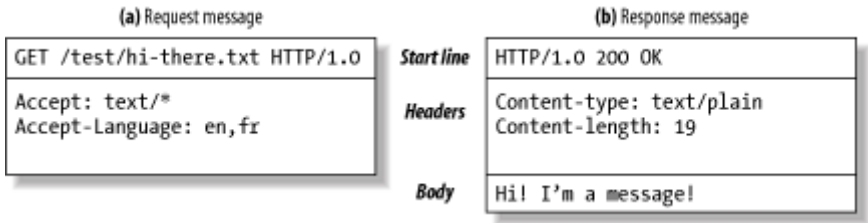


Figure 1-7. HTTP messages have a simple, line-oriented text structure



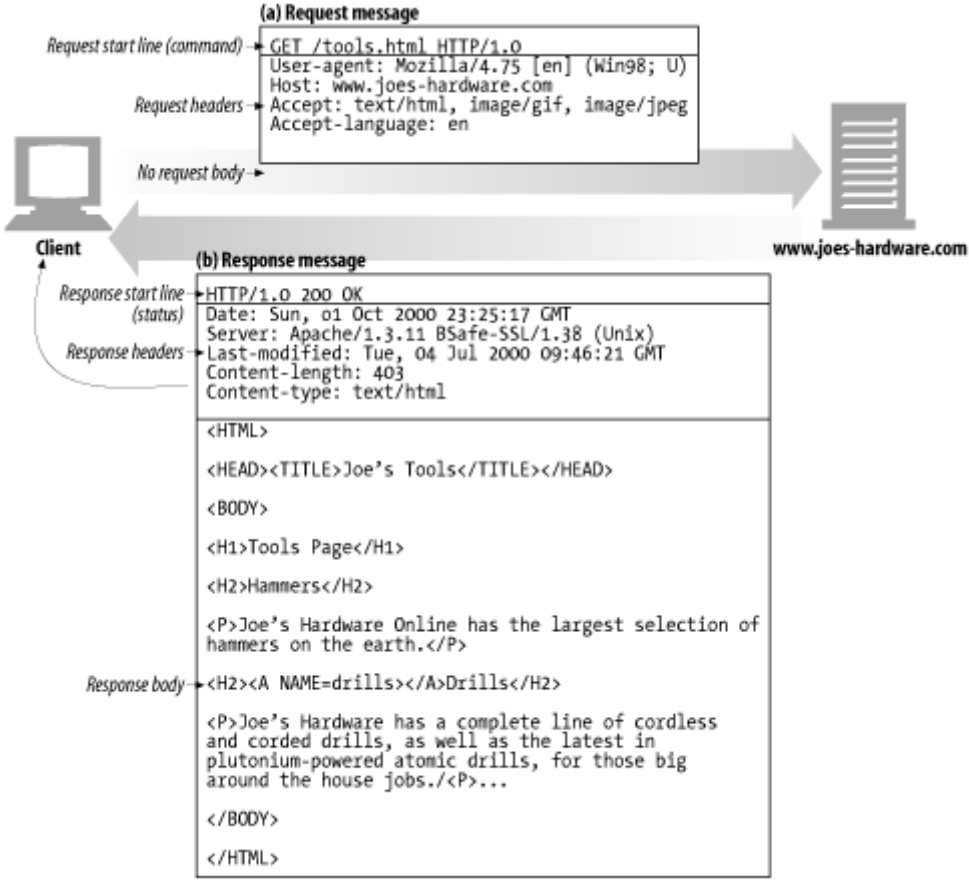

(a) Request message
Request start line (command)
GET /tools.html HTTP/1.0
User-agent: Mozilla/4.75 [en] (Win98; U)
Host: www.joes-hardware.com
Request headers
Accept: text/html, image/gif, image/jpeg
Accept-language: en
No request body
Client
www.joes-hardware.com
(b) Response message
Response start line (status)
HTTP/1.0 200 OK
Date: Sun, 01 Oct 2000 23:25:17 GMT
Server: Apache/1.3.11 BSafe-SSL/1.38 (Unix)
Response headers
Last-modified: Tue, 04 Jul 2000 09:46:21 GMT
Content-length: 403
Content-type: text/html
<HTML>
<HEAD><TITLE>Joe's Tools</TITLE></HEAD>
<BODY>
<H1>Tools Page</H1>
<H2>Hammers</H2>
<P>Joe's Hardware Online has the largest selection of hammers on the earth.</P>
Response body
<H2><A NAME=drills></A>Drills</H2>
<P>Joe's Hardware has a complete line of cordless and corded drills, as well as the latest in plutonium-powered atomic drills, for those big around the house jobs./<P>...
</BODY>
</HTML>

Figure 1-9. HTTP network protocol stack

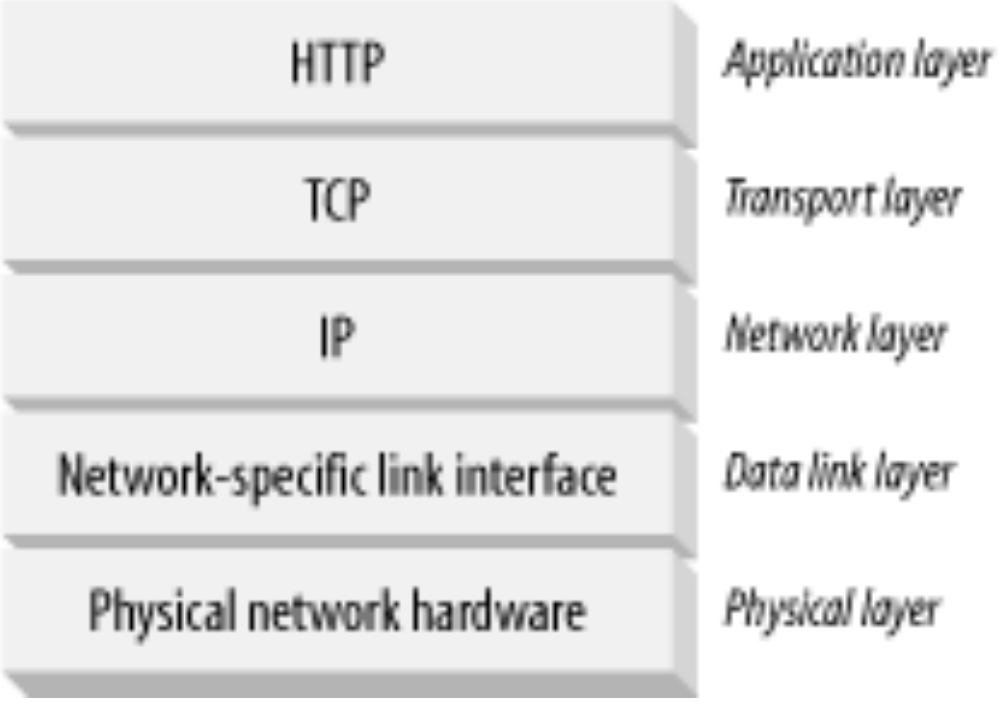
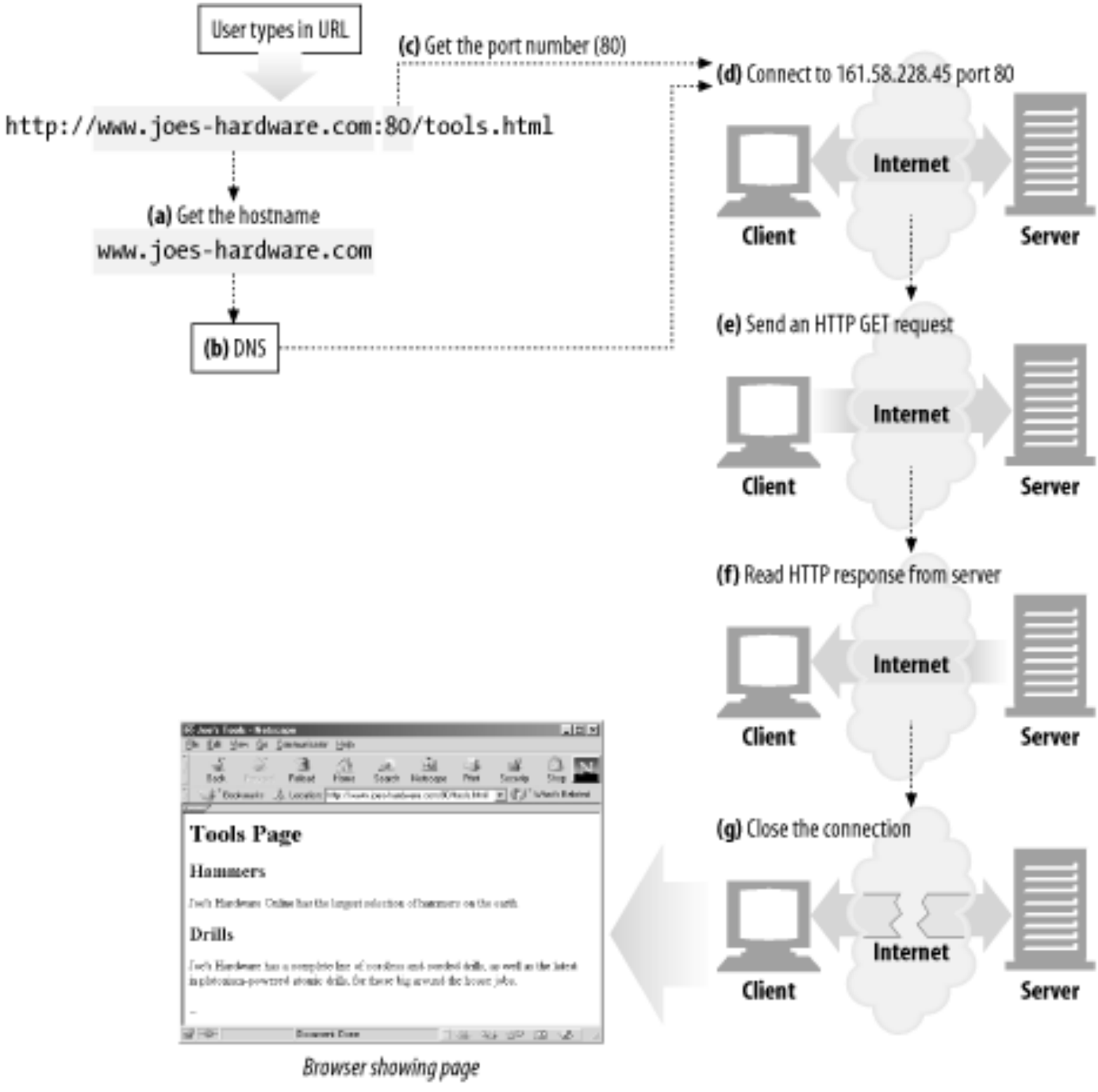


Figure 1-10. Basic browser connection process



Browser showing page

Example 1-1. An HTTP transaction using telnet

```
% telnet www.joes-hardware.com 80
Trying 161.58.228.45...
Connected to joes-hardware.com.
Escape character is '^]'.
GET /tools.html HTTP/1.1
Host: www.joes-hardware.com

HTTP/1.1 200 OK
Date: Sun, 01 Oct 2000 23:25:17 GMT
Server: Apache/1.3.11 BSafe-SSL/1.38 (Unix) FrontPage/4.0.4.3
Last-Modified: Tue, 04 Jul 2000 09:46:21 GMT
ETag: "373979-193-3961b26d"
Accept-Ranges: bytes
Content-Length: 403
Connection: close
Content-Type: text/html

<HTML>
<HEAD><TITLE>Joe's Tools</TITLE></HEAD>
<BODY>
<H1>Tools Page</H1>
<H2>Hammers</H2>
<P>Joe's Hardware Online has the largest selection of hammers on the earth.</P>
<H2><A NAME=drills></A>Drills</H2>
<P>Joe's Hardware has a complete line of cordless and corded drills, as well as the latest
in plutonium-powered atomic drills, for those big around the house jobs.</P> ...
</BODY>
</HTML>
Connection closed by foreign host.
```

1.7 Protocol Versions

There are several versions of the HTTP protocol in use today. HTTP applications need to work hard to robustly handle different variations of the HTTP protocol. The versions in use are:

HTTP/0.9

The 1991 prototype version of HTTP is known as HTTP/0.9. This protocol contains many serious design flaws and should be used only to interoperate with legacy clients. HTTP/0.9 supports only the GET method, and it does not support MIME typing of multimedia content, HTTP headers, or version numbers. HTTP/0.9 was originally defined to fetch simple HTML objects. It was soon replaced with HTTP/1.0.

HTTP/1.0

1.0 was the first version of HTTP that was widely deployed. HTTP/1.0 added version numbers, HTTP headers, additional methods, and multimedia object handling. HTTP/1.0 made it practical to support graphically appealing web pages and interactive forms, which helped promote the wide-scale adoption of the World Wide Web. This specification was never well specified. It represented a collection of best practices in a time of rapid commercial and academic evolution of the protocol.

HTTP/1.0+

Many popular web clients and servers rapidly added features to HTTP in the mid-1990s to meet the demands of a rapidly expanding, commercially successful World Wide Web. Many of these features, including long-lasting "keep-alive" connections, virtual hosting support, and proxy connection support, were added to HTTP and became unofficial, de facto standards. This informal, extended version of HTTP is often referred to as HTTP/1.0+.

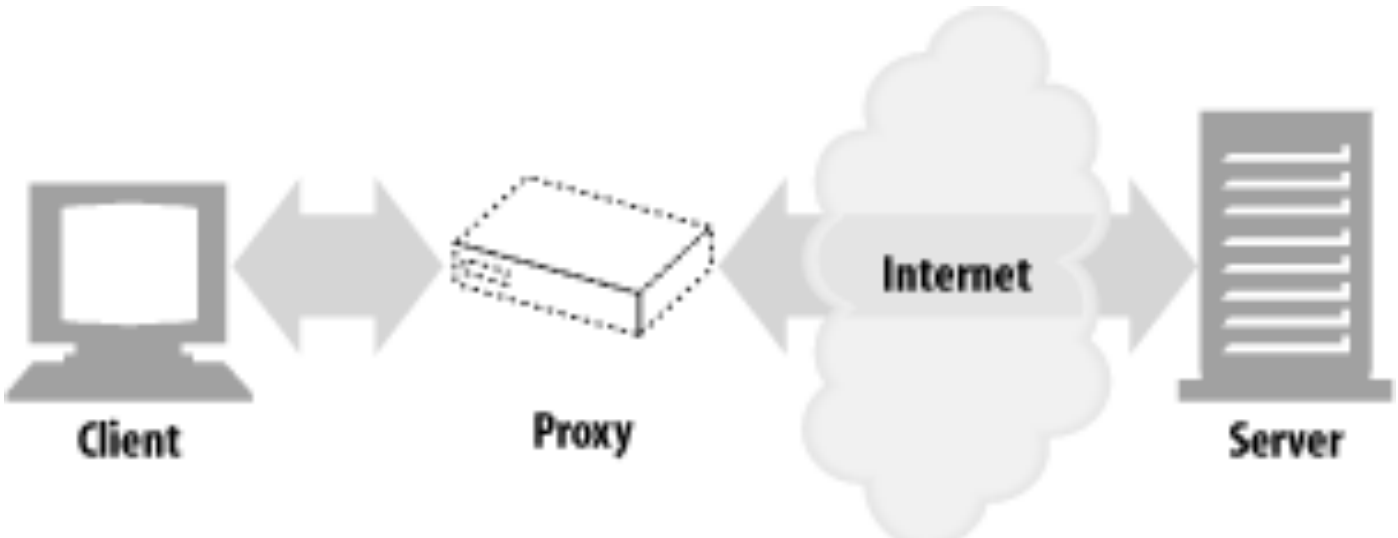
HTTP/1.1

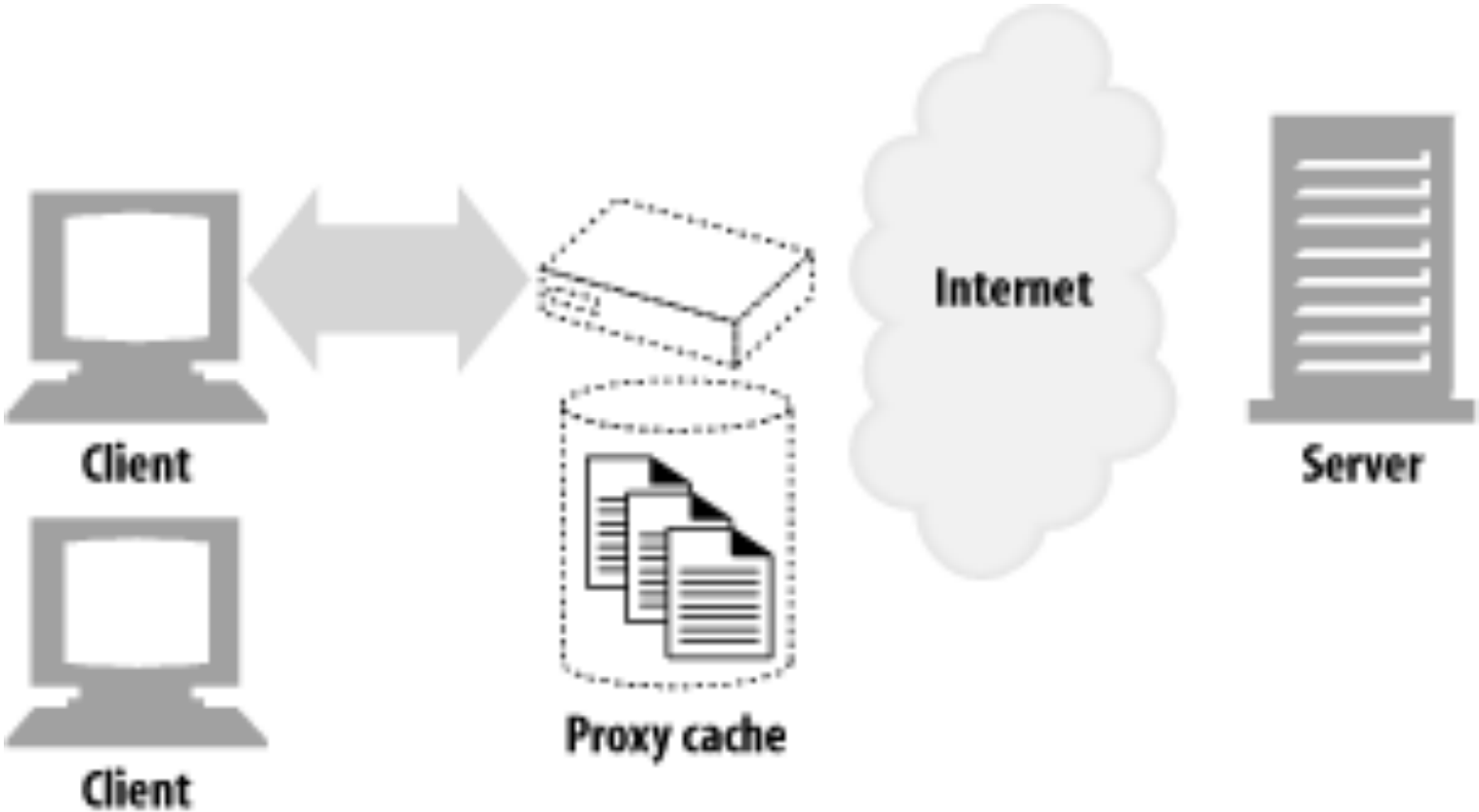
HTTP/1.1 focused on correcting architectural flaws in the design of HTTP, specifying semantics, introducing significant performance optimizations, and removing mis-features. HTTP/1.1 also included support for the more sophisticated web applications and deployments that were under way in the late 1990s. HTTP/1.1 is the current version of HTTP.

HTTP-NG (a.k.a. HTTP/2.0)

HTTP-NG is a prototype proposal for an architectural successor to HTTP/1.1 that focuses on significant performance optimizations and a more powerful framework for remote execution of server logic. The HTTP-NG research effort concluded in 1998, and at the time of this writing, there are no plans to advance this proposal as a replacement for HTTP/1.1. See Chapter 10 for more information.

Figure 1-11. Proxies relay traffic between client and server




Client
Client
Proxy cache
Internet
Server

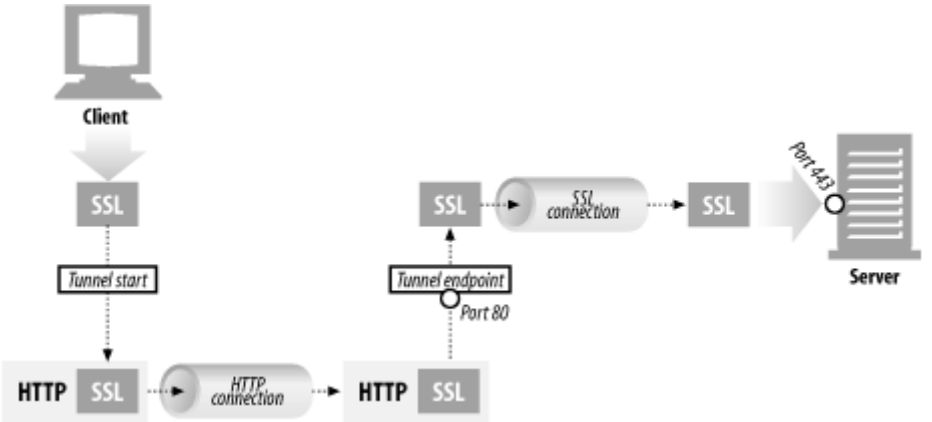

Client
SSL
Tunnel start
HTTP
SSL
HTTP connection
HTTP
SSL
Tunnel endpoint
Port 80
SSL
SSL connection
SSL
Port 443
Server

Figure 1-13. HTTP/FTP gateway

re 1-15. Automated search engine "spiders" are agents, fetching web pages aro
world

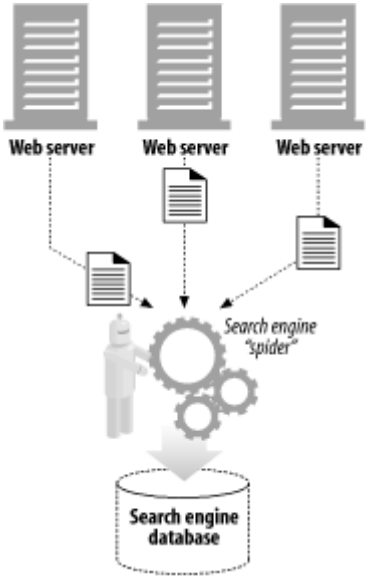


Figure 2-1. How URLs relate to browser, machine, server, and location on the server's filesystem

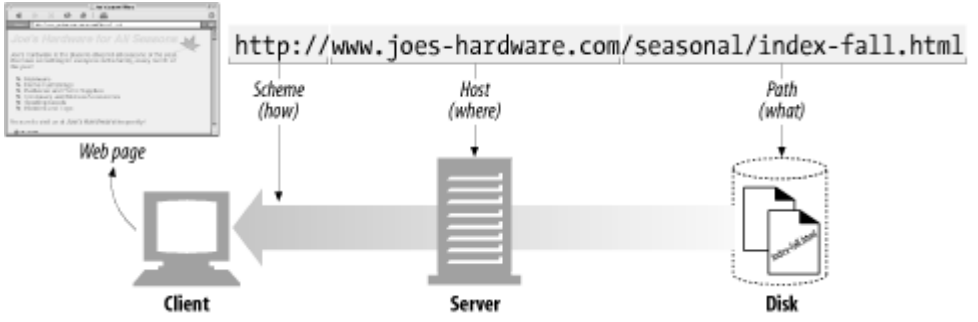


Table 2-1. General URL components		
Component	**Description**	**Default value**
scheme	Which protocol to use when accessing a server to get a resource.	None
user	The username some schemes require to access a resource.	anonymous
password	The password that may be included after the username, separated by a colon (:).	<Email address>
host	The hostname or dotted IP address of the server hosting the resource.	None
port	The port number on which the server hosting the resource is listening. Many schemes have default port numbers (the default port number for HTTP is 80).	Scheme-specific
path	The local name for the resource on the server, separated from the previous URL components by a slash (/). The syntax of the path component is server- and scheme-specific. (We will see later in this chapter that a URL's path can be divided into segments, and each segment can have its own components specific to that segment.)	None
params	Used by some schemes to specify input parameters. Params are name/value pairs. A URL can contain multiple params fields, separated from themselves and the rest of the path by semicolons (;).	None
query	Used by some schemes to pass parameters to active applications (such as databases, bulletin boards, search engines, and other Internet gateways). There is no common format for the contents of the query component. It is separated from the rest of the URL by the "?" character.	None
frag	A name for a piece or part of the resource. The frag field is not passed to the server when referencing the object; it is used internally by the client. It is separated from the rest of the URL by the "#" character.	None

Figure 2-2. The URL query component is sent along to the gateway application

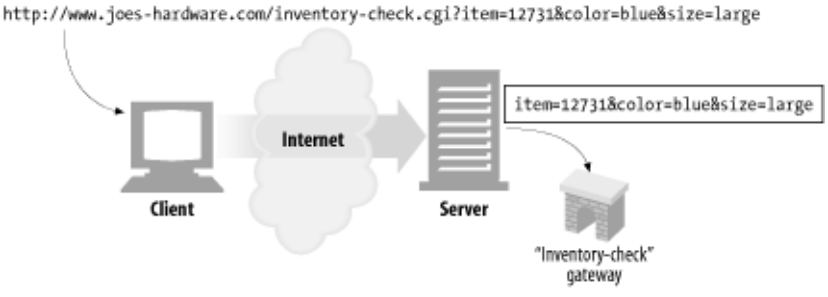


Figure 2-3. The URL fragment is used only by the client, because the server deals with entire objects

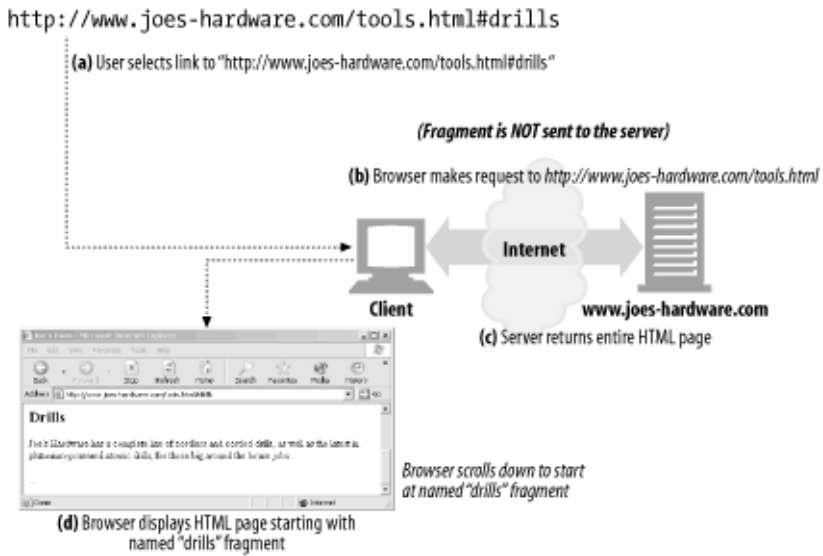


Figure 2-4. Using a base URL

Figure 2-5. Converting relative to absolute URLs

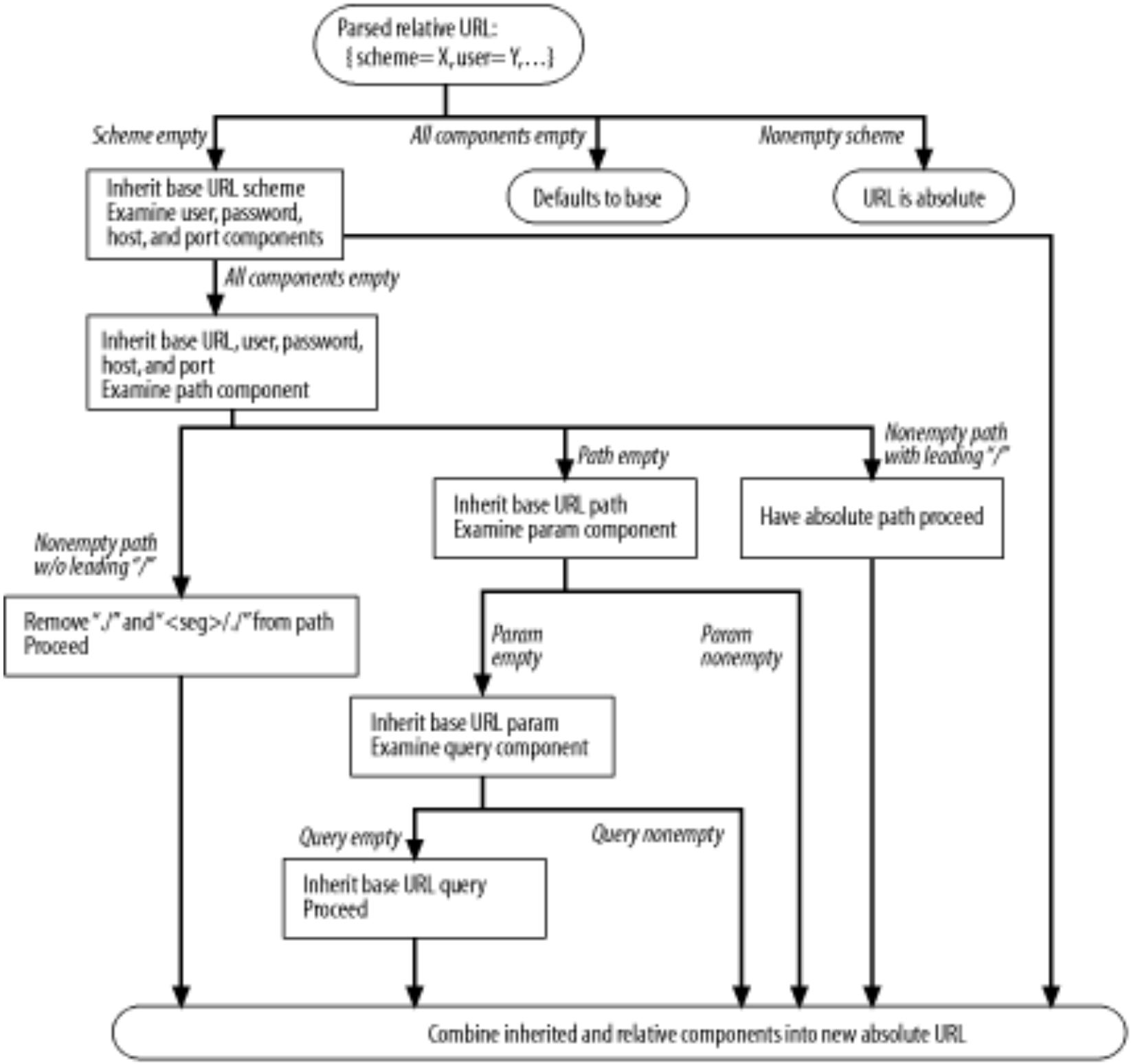


Figure 2-6. PURLs use a resource locator server to name the current location of a resource