How Web Servers Work

by Marshall Brain

Have you ever wondered about the mechanisms that delivered this page to you? Chances are you are sitting at a computer right now, viewing this page in a browser -- so when you clicked on the link for this page, or typed in its URL (**uniform resource locator**), what happened behind the scenes to bring this page onto your screen?

If you've ever been curious about the process, or have ever wanted to know some of the specific mechanisms that allow you to surf the Internet, then read on. In this edition of **HowStuffWorks**, you will learn how Web servers bring pages into your home, school or office. Let's get started!

The Basic Process

Let's say that you are sitting at your computer, surfing the Web, and you get a call from a friend who says, "I just read a great article! Type in this URL and check it out! It's at http://computer.howstuffworks.com/web-server.htm." So you type that URL into your browser and press return. And magically, no matter where in the world that URL lives, the page pops up on your screen!

At the most basic level possible, the following diagram shows the steps that brought that page to your screen:

Your browser formed a connection to a Web server, requested a page and received it. If you want to get into a bit more detail, here are the basic steps that occurred behind the scenes:

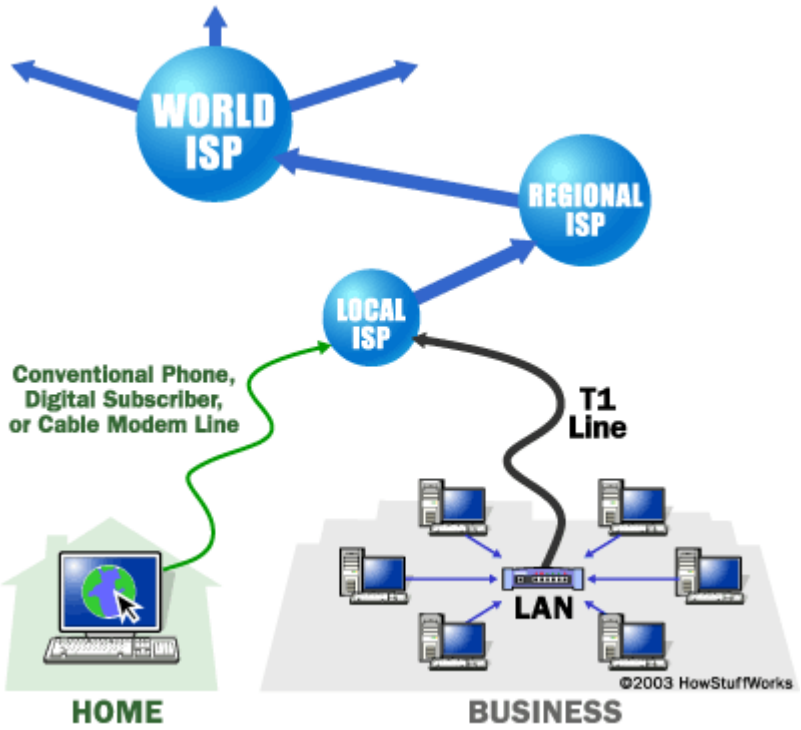
- The browser broke the URL into three parts:
    1. The protocol ("http")
    2. The server name ("www.howstuffworks.com")
    3. The file name ("web-server.htm")
- The browser communicated with a name server to translate the server name "www.howstuffworks.com" into an **IP Address**, which it uses to connect to the server machine.
- The browser then formed a connection to the server at that IP address on port 80. (We'll discuss ports later in this article.)
- Following the HTTP protocol, the browser sent a GET request to the server, asking for the file "http://computer.howstuffworks.com/web-server.htm." (Note that **cookies** may be sent from browser to server with the GET request -- see How Internet Cookies Work for details.)
- The server then sent the HTML text for the Web page to the browser. (Cookies may also be sent from server to browser in the header for the page.)
- The browser read the HTML tags and formatted the page onto your screen.

If you've never explored this process before, that's a lot of new vocabulary. To understand this whole process in detail, you need to learn about IP addresses, ports, protocols... The following sections will lead you through a complete explanation.

The Internet

So what is "the Internet"? The Internet is a gigantic collection of millions of computers, all linked together on a **computer network**. The network allows all of the computers to communicate with one another. A home computer may be linked to the Internet using a phone-line modem, DSL or cable modem that talks to an Internet service provider (**ISP**). A computer in a business or university will usually have a network interface card (**NIC**) that directly connects it to a local area network (**LAN**) inside the business. The business can then connect its LAN to an ISP using a high-speed phone line like a **T1 line**. A T1 line can handle approximately 1.5 million bits per second, while a normal phone line using a modem can typically handle 30,000 to 50,000 bits per second.

ISPs then connect to larger ISPs, and the largest ISPs maintain fiber-optic "backbones" for an entire nation or region. Backbones around the world are connected through fiber-optic lines, undersea cables or satellite links (see this page for a nice backbone and connection diagram). In this way, every computer on the Internet is connected to every other computer on the Internet.



Clients and Servers

In general, all of the machines on the Internet can be categorized as two types: servers and clients. Those machines that provide services (like Web servers or FTP servers) to other machines are **servers**. And the machines that are used to connect to those services are **clients**. When you connect to Yahoo! at www.yahoo.com to read a page, Yahoo! is providing a machine (probably a cluster of very large machines), for use on the Internet, to service your request. Yahoo! is providing a server. Your machine, on the other hand, is probably providing no services to anyone else on the Internet. Therefore, it is a user machine, also known as a client. It is possible and common for a machine to be both a server and a client, but for our purposes here you can think of most machines

as one or the other.

A server machine may provide one or more services on the Internet. For example, a server machine might have software running on it that allows it to act as a Web server, an e-mail server and an FTP server. Clients that come to a server machine do so with a specific intent, so clients direct their requests to a specific software server running on the overall server machine. For example, if you are running a Web browser on your machine, it will most likely want to talk to the Web server on the server machine. Your Telnet application will want to talk to the Telnet server, your e-mail application will talk to the e-mail server, and so on...

IP Addresses

To keep all of these machines straight, each machine on the Internet is assigned a unique address called an **IP address**. IP stands for **Internet protocol**, and these addresses are 32-bit numbers, normally expressed as four "octets" in a "dotted decimal number." A typical IP address looks like this:

```
216.27.61.137
```

The four numbers in an IP address are called **octets** because they can have values between 0 and 255, which is 2^8 possibilities per octet.

Every machine on the Internet has a unique IP address. A server has a static IP address that does not change very often. A home machine that is dialing up through a modem often has an IP address that is assigned by the ISP when the machine dials in. That IP address is unique for that session -- it may be different the next time the machine dials in. This way, an ISP only needs one IP address for each modem it supports, rather than for each customer.

If you are working on a Windows machine, you can view a lot of the Internet information for your machine, including your current IP address and hostname, with the command **WINIPCFG.EXE** (IPCONFIG.EXE for Windows 2000/XP). On a UNIX machine, type **nslookup** at the command prompt, along with a machine name, like www.howstuffworks.com -- e.g. "nslookup www.howstuffworks.com" -- to display the IP address of the machine, and you can use the command **hostname** to learn the name of your machine. (For more information on IP addresses, see IANA.)

As far as the Internet's machines are concerned, an IP address is all you need to talk to a server. For example, in your browser, you can type the URL **http://209.116.69.66** and arrive at the machine that contains the Web server for HowStuffWorks. On some servers, the IP address alone is not sufficient, but on most large servers it is -- keep reading for details.

Name Servers

Because most people have trouble remembering the strings of numbers that make up IP addresses, and because IP addresses sometimes need to change, all servers on the Internet also have human-readable names, called **domain names**. For example, www.howstuffworks.com is a permanent, human-readable name. It is easier for most of us to remember www.howstuffworks.com than it is to remember 209.116.69.66.

The name www.howstuffworks.com actually has three parts:

1. The host name ("www")
2. The domain name ("howstuffworks")
3. The top-level domain name ("com")

Domain names are managed by a company called VeriSign. VeriSign creates the top-level domain names and guarantees that all names within a top-level domain are unique. VeriSign also maintains contact information for each site and runs the "whois" database. The host name is created by the company hosting the domain. "www" is a very common host name, but many places now either omit it or replace it with a different host name that indicates a specific area of the site. For example, in encarta.msn.com, the domain name for Microsoft's Encarta encyclopedia, "encarta" is designated as the host name instead of "www."

A set of servers called domain name servers (DNS) maps the human-readable names to the IP addresses. These servers are simple databases that map names to IP addresses, and they are distributed all over the Internet. Most individual companies, ISPs and universities maintain small name servers to map host names to IP addresses. There are also central name servers that use data supplied by VeriSign to map domain names to IP addresses.

The whois Command

On a UNIX machine, you can use the **whois** command to look up information about a domain name. You can do the same thing using the whois form at VeriSign. If you type in a domain name, like "howstuffworks.com," it will return to you the registration information for that domain, including its IP address.

If you type the URL "http://computer.howstuffworks.com/web-server.htm" into your browser, your browser extracts the name "www.howstuffworks.com," passes it to a domain name server, and the domain name server returns the correct IP address for www.howstuffworks.com. A number of name servers may be involved to get the right IP address. For example, in the case of www.howstuffworks.com, the name server for the "com" top-level domain will know the IP address for the name server that knows host names, and a separate query to that name server, operated by the HowStuffWorks ISP, may deliver the actual IP address for the HowStuffWorks server machine.

On a UNIX machine, you can access the same service using the **nslookup** command. Simply type a name like "www.howstuffworks.com" into the command line, and the command will query the name servers and deliver the corresponding IP address to you.

So here it is: The Internet is made up of millions of machines, each with a unique IP address. Many of these machines are **server machines**, meaning that they provide services to other machines on the Internet. You have heard of many of these servers: e-mail servers, Web servers, FTP servers, Gopher servers and Telnet servers, to name a few. All of these are provided by server machines.

Ports

Any server machine makes its services available to the Internet using numbered **ports**, one for each service that is available on the server. For example, if a server machine is running a Web server and an FTP server, the Web server would typically be available on port 80, and the FTP server would be available on port 21. Clients connect to a service at a specific IP address and on a specific port.

Each of the most well-known services is available at a well-known port number. Here are some common port numbers:

- echo 7
- daytime 13
- qotd 17 (Quote of the Day)
- ftp 21
- telnet 23
- smtp 25 (Simple Mail Transfer, meaning e-mail)
- time 37
- nameserver 42
- nicname 43 (Who Is)
- gopher 70
- finger 79

- WWW 80

If the server machine accepts connections on a port from the outside world, and if a firewall is not protecting the port, you can connect to the port from anywhere on the Internet and use the service. Note that there is nothing that forces, for example, a Web server to be on port 80. If you were to set up your own machine and load Web server software on it, you could put the Web server on port 918, or any other unused port, if you wanted to. Then, if your machine were known as xxx.yyy.com, someone on the Internet could connect to your server with the URL **http://xxx.yyy.com:918**. The ":918" explicitly specifies the port number, and would have to be included for someone to reach your server. When no port is specified, the browser simply assumes that the server is using the well-known port 80.

Protocols

Once a client has connected to a service on a particular port, it accesses the service using a specific protocol. The **protocol** is the pre-defined way that someone who wants to use a service talks with that service. The "someone" could be a person, but more often it is a computer program like a Web browser. Protocols are often text, and simply describe how the client and server will have their conversation.

Perhaps the simplest protocol is the **daytime protocol**. If you connect to port 13 on a machine that supports a daytime server, the server will send you its impression of the current date and time and then close the connection. The protocol is, "If you connect to me, I will send you the date and time and then disconnect." Most UNIX machines support this server. If you would like to try it out, you can connect to one with the Telnet application. In UNIX, the session would look like this:

```
%telnet web67.ntx.net 13
Trying 216.27.61.137...
Connected to web67.ntx.net.
Escape character is '^]'.
Sun Oct 25 08:34:06 1998
Connection closed by foreign host.
```

On a Windows machine, you can access this server by typing "telnet web67.ntx.net 13" at the MSDOS prompt.

In this example, web67.ntx.net is the server's UNIX machine, and 13 is the port number for the daytime service. The Telnet application connects to port 13 (telnet naturally connects to port 23, but you can direct it to connect to any port), then the server sends the date and time and disconnects. Most versions of Telnet allow you to specify a port number, so you can try this using whatever version of Telnet you have available on your machine.

Most protocols are more involved than daytime and are specified in Request for Comment (RFC) documents that are publicly available (see http://sunsite.auc.dk/RFC/ for a nice archive of all RFCs). Every Web server on the Internet conforms to the HTTP protocol, summarized nicely in this article. The most basic form of the protocol understood by an HTTP server involves just one command: GET. If you connect to a server that understands the HTTP protocol and tell it to "GET filename," the server will respond by sending you the contents of the named file and then disconnecting. Here's a typical session:

```
%telnet www.howstuffworks.com 80
Trying 216.27.61.137...
Connected to howstuffworks.com.
```

```
Escape character is '^]'.
GET http://computer.howstuffworks.com/
<html>
<head>
<title>Welcome to How Stuff Works</title>
  ...
</body>
</html>
Connection closed by foreign host.
```

In the original HTTP protocol, all you would have sent was the actual filename, such as "/" or "/web-server.htm." The protocol was later modified to handle the sending of the complete URL. This has allowed companies that host **virtual domains**, where many domains live on a single machine, to use one IP address for all of the domains they host. It turns out that hundreds of domains are hosted on 209.116.69.66 -- the HowStuffWorks IP address.

Putting It All Together

Now you know a tremendous amount about the Internet. You know that when you type a URL into a browser, the following steps occur:

- The browser breaks the URL into three parts:
    1. The protocol ("http")
    2. The server name ("www.howstuffworks.com")
    3. The file name ("web-server.htm")
- The browser communicates with a name server to translate the server name, "www.howstuffworks.com," into an **IP address**, which it uses to connect to that server machine.
- The browser then forms a connection to the Web server at that IP address on port 80.
- Following the HTTP protocol, the browser sends a GET request to the server, asking for the file "http://computer.howstuffworks.com/web-server.htm." (Note that **cookies** may be sent from browser to server with the GET request -- see How Internet Cookies Work for details.)
- The server sends the HTML text for the Web page to the browser. (Cookies may also be sent from server to browser in the header for the page.)
- The browser reads the HTML tags and formats the page onto your screen.

Extras

You can see from this description that a Web server can be a pretty simple piece of software. It takes the file name sent in with the GET command, retrieves that file and sends it down the wire to the browser. Even if you take into account all of the code to handle the ports and port connections, you could easily create a C program that implements a simple Web server in less than 500 lines of code. Obviously, a full-blown enterprise-level Web server is more involved, but the basics are very simple.

Most servers add some level of **security** to the serving process. For example, if you have ever gone to a Web page and had the browser pop up a dialog box asking for your name and password, you have encountered a password-protected page. The server lets the owner of the page maintain a list of names and passwords for those people who are allowed to access the page; the server lets only those people who know the proper password see the page. More advanced servers add further security to allow an encrypted connection between server and browser, so that sensitive information like credit card numbers can be sent on the Internet.

That's really all there is to a Web server that delivers standard, static pages. Static pages are those that do not change unless the creator edits the page.

But what about the Web pages that are **dynamic**? For example:

- Any guest book allows you to enter a message in an HTML form, and the next time the guest book is viewed, the page will contain the new entry.
- The whois form at VeriSign allows you to enter a domain name on a form, and the page returned is different depending on the domain name entered.
- Any search engine lets you enter keywords on an HTML form, and then it dynamically creates a page based on the keywords you enter.

In all of these cases, the Web server is not simply "looking up a file." It is actually processing information and generating a page based on the specifics of the query. In almost all cases, the Web server is using something called **CGI scripts** to accomplish this feat. CGI scripts are a topic unto themselves, and are described in the HowStuffWorks article How CGI Scripting Work.