# How Does the Internet Work?

**Introduction**

How does the Internet work? Good question! The Internet's growth has become explosive and it seems impossible to escape the bombardment of *www.com*'s seen constantly on television, heard on radio, and seen in magazines. Because the Internet has become such a large part of our lives, a good understanding is needed to use this new tool most effectively.

This whitepaper explains the underlying infrastructure and technologies that make the Internet work. It does not go into great depth, but covers enough of each area to give a basic understanding of the concepts involved. For any unanswered questions, a list of resources is provided at the end of the paper. Any comments, suggestions, questions, etc. are encouraged and may be directed to the author at rshuler@gobcg.com.

**Where to Begin? Internet Addresses**

Because the Internet is a global network of computers each computer connected to the Internet **must** have a unique address. Internet addresses are in the form **nnn.nnn.nnn.nnn** where nnn must be a number from 0 - 255. This address is known as an IP address. (IP stands for Internet Protocol; more on this later.)

The picture below illustrates two computers connected to the Internet; your computer with IP address 1.2.3.4 and another computer with IP address 5.6.7.8. The Internet is represented as an abstract object in-between. (As this paper progresses, the Internet portion of Diagram 1 will be explained and redrawn several times as the details of the Internet are exposed.)

|  |
| --- |
| Diagram 1 |
| Diagram 1 |

If you connect to the Internet through an Internet Service Provider (ISP), you are usually assigned a temporary IP address for the duration of your dial-in session. If you connect to the Internet from a local area network (LAN) your computer might have a permanent IP address or it might obtain a temporary one from a DHCP (Dynamic Host Configuration Protocol) server. In any case, if you are connected to the Internet, your computer has a unique IP address.

**Protocol Stacks and Packets**

So your computer is connected to the Internet and has a unique address. How does it 'talk' to other computers connected to the Internet? An example should serve here: Let's say your IP address is 1.2.3.4 and you want to send a message to the computer 5.6.7.8. The message you want to send is "Hello computer 5.6.7.8!". Obviously, the message must be transmitted over whatever kind of wire connects your computer to the Internet. Let's say you've dialed into your ISP from home and the message must be transmitted over the phone line. Therefore the message must be translated from alphabetic text into electronic signals, transmitted over the Internet, then translated back into alphabetic text. How is this accomplished? Through the use of a **protocol stack**. Every computer needs one to communicate on the Internet and it is usually built into the computer's operating system (i.e. Windows, Unix, etc.). The protocol stack used on the Internet is refered to as the TCP/IP protocol stack because of the two major communication protocols used. The TCP/IP stack looks like this:

|  |  |
| --- | --- |
| **Protocol Layer** | **Comments** |
| Application Protocols Layer | Protocols specific to applications such as WWW, e-mail, FTP, etc. |
| Transmission Control Protocol Layer | TCP directs packets to a specific application on a computer using a port number. |
| Internet Protocol Layer | IP directs packets to a specific computer using an IP address. |
| Hardware Layer | Converts binary packet data to network signals and back. (E.g. ethernet network card, modem for phone lines, etc.) |

If we were to follow the path that the message "Hello computer 5.6.7.8!" took from our computer to the computer with IP address 5.6.7.8, it would happen something like this:

|  |
| --- |
| Diagram 2 |
| Diagram 2 |

1. The message would start at the top of the protocol stack on your computer and work it's way downward.
2. If the message to be sent is long, each stack layer that the message passes through may break the message up into smaller chunks of data. This is because data sent over the Internet (and most computer networks) are sent in manageable chunks. On the Internet, these chunks of data are known as **packets**.
3. The packets would go through the Application Layer and continue to the TCP layer. Each packet is assigned a **port number**. Ports will be explained later, but suffice to say that many programs may be using the TCP/IP stack and sending messages. We need to know which program on the destination computer needs to receive the message because it will be listening on a specific port.
4. After going through the TCP layer, the packets proceed to the IP layer. This is where each packet receives it's destination address, 5.6.7.8.
5. Now that our message packets have a port number and an IP address, they are ready to be sent over the Internet. The hardware layer takes care of turning our packets containing the alphabetic text of our message into electronic signals and transmitting them over the phone line.
6. On the other end of the phone line your ISP has a direct connection to the Internet. The ISPs **router** examines the destination address in each packet and determines where to send it. Often, the packet's next stop is another router. More on routers and Internet infrastructure later.
7. Eventually, the packets reach computer 5.6.7.8. Here, the packets start at the bottom of the destination computer's TCP/IP stack and work upwards.
8. As the packets go upwards through the stack, all routing data that the sending computer's stack added (such as IP address and port number) is stripped from the packets.
9. When the data reaches the top of the stack, the packets have been re-assembled into their original form, "Hello computer 5.6.7.8!"

**Domain Names and Address Resolution**

But what if you don't know the IP address of the computer you want to connect to? What if you need to access a web server referred to as *www.anothercomputer.com*? How does your web browser know where on the Internet this computer lives? The answer to all these questions is the **Domain Name Service** or **DNS**. The DNS is a distributed database which keeps track of computer's names and their corresponding IP addresses on the Internet.

Many computers connected to the Internet host part of the DNS database and the software that allows others to access it. These computers are known as DNS servers. No DNS server contains the entire database; they only contain a subset of it. If a DNS server does not contain the domain name requested by another computer, the DNS server re-directs the requesting computer to another DNS server.

|  |
| --- |
| Diagram 6 |
| Diagram 6 |

The Domain Name Service is structured as a hierarchy similar to the IP routing hierarchy. The computer requesting a name resolution will be re-directed 'up' the hierarchy until a DNS server is found that can resolve the domain name in the request. Figure 6 illustrates a portion of the hierarchy. At the top of the tree are the domain roots. Some of the older, more common domains are seen near the top. What is not shown are the multitude of DNS servers around the world which form the rest of the hierarchy.

When an Internet connection is setup (e.g. for a LAN or Dial-Up Networking in Windows), one primary and one or more secondary DNS servers are usually specified as part of the installation. This way, any Internet applications that need domain name resolution will be able to function correctly. For example, when you enter a web address into your web browser, the browser first connects to your primary DNS server. After obtaining the IP address for the domain name you entered, the browser then connects to the target computer and requests the web page you wanted.

|  |
| --- |
| **Check It Out - Disable DNS in Windows** |
| If you're using Windows 95/NT and access the Internet, you may view your DNS server(s) and even disable them.  *If you use Dial-Up Networking:* Open your Dial-Up Networking window (which can be found in Windows Explorer under your CD-ROM drive and above Network Neighborhood). Right click on your Internet connection and click Properties. Near the bottom of the connection properties window press the TCP/IP Settings... button.  *If you have a permanent connection to the Internet:* Right click on Network Neighborhood and click Properties. Click TCP/IP Properties. Select the DNS Configuration tab at the top.  You should now be looking at your DNS servers' IP addresses. Here you may disable DNS or set your DNS servers to 0.0.0.0. (Write down your DNS servers' IP addresses first. You will probably have to restart Windows as well.) Now enter an address into your web browser. The browser won't be able to resolve the domain name and you will probably get a nasty dialog box explaining that a DNS server couldn't be found. However, if you enter the corresponding IP address instead of the domain name, the browser will be able to retrieve the desired web page. (Use ping to get the IP address prior to disabling DNS.) Other Microsoft operating systems are similar. |

**Internet Protocols Revisited**

As hinted to earlier in the section about protocol stacks, one may surmise that there are many protocols that are used on the Internet. This is true; there are many communication protocols required for the Internet to function. These include the TCP and IP protocols, routing protocols, medium access control protocols, application level protocols, etc. The following sections describe some of the more important and commonly used protocols on the Internet. Higher level protocols are discussed first, followed by lower level protocols.

**Application Protocols: HTTP and the World Wide Web**

One of the most commonly used services on the Internet is the World Wide Web (WWW). The application protocol that makes the web work is **Hypertext Transfer Protocol** or **HTTP**. Do not confuse this with the Hypertext Markup Language (HTML). HTML is the language used to write web pages. HTTP is the protocol that web browsers and web servers use to communicate with each other over the Internet. It is an application level protocol because it sits on top of the TCP layer in the protocol stack and is used by specific applications to talk to one another. In this case the applications are web browsers and web servers.

HTTP is a connectionless text based protocol. Clients (web browsers) send requests to web servers for web elements such as web pages and images. After the request is serviced by a server, the connection between client and server across the Internet is disconnected. A new connection must be made for each request. Most protocols are connection oriented. This means that the two computers communicating with each other keep the connection open over the Internet. HTTP does not however. Before an HTTP request can be made by a client, a new connection must be made to the server.

When you type a URL into a web browser, this is what happens:

1. If the URL contains a domain name, the browser first connects to a domain name server and retrieves the corresponding IP address for the web server.
2. The web browser connects to the web server and sends an HTTP request (via the protocol stack) for the desired web page.
3. The web server receives the request and checks for the desired page. If the page exists, the web server sends it. If the server cannot find the requested page, it will send an HTTP 404 error message. (404 means 'Page Not Found' as anyone who has surfed the web probably knows.)
4. The web browser receives the page back and the connection is closed.
5. The browser then parses through the page and looks for other page elements it needs to complete the web page. These usually include images, applets, etc.
6. For each element needed, the browser makes additional connections and HTTP requests to the server for each element.
7. When the browser has finished loading all images, applets, etc. the page will be completely loaded in the browser window.

|  |
| --- |
| **Check It Out - Use Your Telnet Client to Retrieve a Web Page Using HTTP** |
| Telnet is a remote terminal service used on the Internet. Its use has declined lately, but it is a very useful tool to study the Internet. In Windows find the default telnet program. It may be located in the Windows directory named telnet.exe. When opened, pull down the Terminal menu and select Preferences. In the preferences window, check Local Echo. (This is so you can see your HTTP request when you type it.) Now pull down the Connection menu and select Remote System. Enter www.google.com for the Host Name and 80 for the Port. (Web servers usually listen on port 80 by default.) Press Connect. Now type  GET / HTTP/1.0  and press Enter twice. This is a simple HTTP request to a web server for its root page. You should see a web page flash by and then a dialog box should pop up to tell you the connection was lost. If you'd like to save the retrieved page, turn on logging in the Telnet program. You may then browse through the web page and see the HTML that was used to write it. |

Most Internet protocols are specified by Internet documents known as a **Request For Comments** or **RFC**s. RFCs may be found at several locations on the Internet. See the Resources section below for appropriate URL's. HTTP version 1.0 is specified by RFC 1945.

**Transmission Control Protocol**

Under the application layer in the protocol stack is the TCP layer. When applications open a connection to another computer on the Internet, the messages they send (using a specific application layer protocol) get passed down the stack to the TCP layer. **TCP is responsible for routing application protocols to the correct application on the destination computer**. To accomplish this, port numbers are used. Ports can be thought of as seperate channels on each computer. For example, you can surf the web while reading e-mail. This is because these two applications (the web browser and the mail client) used different port numbers. When a packet arrives at a computer and makes its way up the protocol stack, the TCP layer decides which application receives the packet based on a port number.

TCP works like this:

* When the TCP layer receives the application layer protocol data from above, it segments it into manageable 'chunks' and then adds a TCP header with specific TCP information to each 'chunk'. The information contained in the TCP header includes the port number of the application the data needs to be sent to.
* When the TCP layer receives a packet from the IP layer below it, the TCP layer strips the TCP header data from the packet, does some data reconstruction if necessary, and then sends the data to the correct application using the port number taken from the TCP header.

This is how TCP routes the data moving through the protocol stack to the correct application.

TCP is not a textual protocol. **TCP is a connection-oriented, reliable, byte stream service**. Connection-oriented means that two applications using TCP must first establish a connection before exchanging data. TCP is reliable because for each packet received, an acknowledgement is sent to the sender to confirm the delivery. TCP also includes a checksum in its header for error-checking the received data. The TCP header looks like this:

|  |
| --- |
| Diagram 7 |
| Diagram 7 |

Notice that there is no place for an IP address in the TCP header. This is because TCP doesn't know anything about IP addresses. TCP's job is to get application level data from application to application reliably. The task of getting data from computer to computer is the job of IP.

|  |
| --- |
| **Check It Out - Well Known Internet Port Numbers** |
| Listed below are the port numbers for some of the more commonly used Internet services.   |  |  | | --- | --- | | FTP | 20/21 | | Telnet | 23 | | SMTP | 25 | | HTTP | 80 | | Quake III Arena | 27960 | |

**Internet Protocol**

Unlike TCP, **IP is an unreliable, connectionless protocol**. IP doesn't care whether a packet gets to its destination or not. Nor does IP know about connections and port numbers. **IP's job is to send and route packets to other computers**. IP packets are independent entities and may arrive out of order or not at all. It is TCP's job to make sure packets arrive and are in the correct order. About the only thing IP has in common with TCP is the way it receives data and adds its own IP header information to the TCP data. The IP header looks like this:

|  |
| --- |
| Diagram 8 |
| Diagram 8 |

Above we see the IP addresses of the sending and receiving computers in the IP header. Below is what a packet looks like after passing through the application layer, TCP layer, and IP layer. The application layer data is segmented in the TCP layer, the TCP header is added, the packet continues to the IP layer, the IP header is added, and then the packet is transmitted across the Internet.

|  |
| --- |
| Diagram 9 |
| Diagram 9 |