How does the Internet work?

To help you understand how the Internet works, we'll look at the things that happen when you do a typical Internet operation -- pointing a browser at my home page. This site is:

http://www.iusb.edu/~mhollowa/

which means it lives in the directory of ~mhollowa under the World Wide Web export directory of the host www.iusb.edu.

Names and locations

The first thing your browser has to do is to establish a network connection to the machine where the document lives. To do that, it first has to find the network location of the host www.isub.edu (‘host' is short for `host machine' or `network host'; www.iusb.edu is the hostname). The corresponding location is actually a number called an IP address (we'll explain the `IP' part of this term later).

To do this, your browser queries a program called a name server. The name server may live on your machine, but it's more likely to run on a service machine that yours talks to. When you sign up with an ISP (Internet Service Provider), part of your setup procedure will almost certainly involve telling your Internet software the IP address of a name server on the ISP's network.

The name servers on different machines talk to each other, exchanging and keeping up to date all the information needed to resolve hostnames (map them to IP addresses). Your name server may query three or four different sites across the network in the process of resolving www.iusb.edu, but this usually happens very quickly (as in less than a second).

The name server will tell your browser that Iusb.edu IP address is 152.2.22.81( this is just an example, not actual IP address); knowing this, your machine will be able to exchange bits with IUSB directly.

Packets and routers

What the browser wants to do is send a command to the Web server on IUSB that looks like this:

GET /~mhollowa/1.0

Here's how that happens. The command is made into a packet, a block of bits like a telegram that is wrapped with three important things; the source address (the IP address of your machine), the destination address (152.2.22.81), and a service number or port number (80, in this case) that indicates that it's a World Wide Web request.

Your machine then ships the packet down the wire (modem connection to your ISP, or local network) until it gets to a specialized machine called a router. The router has a map of the Internet in its memory -- not always a complete one, but one that completely describes your
network neighborhood and knows how to get to the routers for other neighborhoods on the Internet.

Your packet may pass through several routers on the way to its destination. Routers are smart. They watch how long it takes for other routers to acknowledge having received a packet. They use that information to direct traffic over fast links. They use it to notice when another routers (or a cable) have dropped off the network, and compensate if possible by finding another route.

There's an urban legend that the Internet was designed to survive nuclear war. This is not true, but the Internet's design is extremely good at getting reliable performance out of flaky hardware in an uncertain world. This is directly due to the fact that its intelligence is distributed through thousands of routers rather than a few massive switches (like the phone network). This means that failures tend to be well localized and the network can route around them.

Once your packet gets to its destination machine, that machine uses the service number to feed the packet to the web server. The web server can tell where to reply to by looking at the command packet's source IP address. When the web server returns this document, it will be broken up into a number of packets. The size of the packets will vary according to the transmission media in the network and the type of service.

**TCP and IP**

To understand how multiple-packet transmissions are handled, you need to know that the Internet actually uses two protocols, stacked one on top of the other.

The lower level, IP (Internet Protocol), knows how to get individual packets from a source address to a destination address (this is why these are called IP addresses). However, IP is not reliable; if a packet gets lost or dropped, the source and destination machines may never know it. In network jargon, IP is a connectionless protocol; the sender just fires a packet at the receiver and doesn't expect an acknowledgement.

IP is fast and cheap, though. Sometimes fast, cheap and unreliable is OK. When you play networked Doom or Quake, each bullet is represented by an IP packet. If a few of those get lost, that's OK.

The upper level, TCP (Transmission Control Protocol), gives you reliability. When two machines negotiate a TCP connection (which they do using IP), the receiver knows to send acknowledgements of the packets it sees back to the sender. If the sender doesn't see an acknowledgement for a packet within some timeout period, it resends that packet. Furthermore, the sender gives each TCP packet a sequence number, which the receiver can use you reassemble packets in case they show up out of order. (This can happen if network links go up or down during a connection.)

TCP/IP packets also contain a checksum to enable detection of data corrupted by bad links. So, from the point of view of anyone using TCP/IP and name servers, it looks like a reliable way to pass streams of bytes between hostname/service-number pairs. People who write network protocols almost never have to think about all the packetizing, packet reassembly, error checking, check summing, and retransmission that goes on below that level.
HTTP, an application protocol

Now let's get back to our example. Web browsers and servers speak an application protocol that runs on top of TCP/IP, using it simply as a way to pass strings of bytes back and forth. This protocol is called HTTP (Hyper-Text Transfer Protocol) and we've already seen one command in it -- the GET shown above.

When the GET command goes to www.iusb.edu's webserver with service number 80, it will be dispatched to a server daemon listening on port 80. Most Internet services are implemented by server daemons that do nothing but wait on ports, watching for and executing incoming commands.

If the design of the Internet has one overall rule, it's that all the parts should be as simple and human-accessible as possible. HTTP, and its relatives (like the Simple Mail Transfer Protocol, SMTP, that is used to move electronic mail between hosts) tend to use simple printable-text commands that end with a carriage-return/line feed.

This is marginally inefficient; in some circumstances you could get more speed by using a tightly-coded binary protocol. But experience has shown that the benefits of having commands be easy for human beings to describe and understand outweigh any marginal gain in efficiency that you might get at the cost of making things tricky and opaque.

Therefore, what the server daemon ships back to you via TCP/IP is also text. The beginning of the response will look something like this (a few headers have been suppressed):

```
HTTP/1.1 200 OK
Date: Sat, 10 Oct 1998 18:43:35 GMT
Server: Apache/1.2.6 Red Hat
Content-Length: 2982
Content-Type: text/html
```

These headers will be followed by a blank line and the text of the web page (after which the connection is dropped). Your browser just displays that page. The headers tell it how (in particular, the Content-Type header tells it the returned data is really HTML).