Well, enough questions about the history of the information highway system. It's time to walk to the edge of the road, try and hitch a ride, and be on your way.

CHAPTER THREE

How the Internet Works

Moving Bits from One Place to Another
Making the Network Friendly

It's nice to know a bit about how things work. It allows you to make sense out of some of the hints you will see in this book. They will make sense, rather than seeming like capricious rules to be learned by rote. Lest you be scared away, we will explore this with a maximum amount of handwaving. We'll never say "this field is 3 bits long . . . "; we won't even think about it! If you want to know more, several books on the Internet's implementation are available.*

In this chapter, we will look at packet switching networks and how, by putting TCP/IP on top of such a network, something useful happens. We will talk about the basic protocols that govern how the Internet communicates: TCP and its poor cousin, UDP. These are the network's building blocks. At this point the Internet is fairly boring (frustrating and hard to use). When you put the Domain Name System and a few applications on top of it, it becomes something useful.

If you decide this isn't your cup of tea, feel free to skip the beginning of this chapter. Do read the section on the Domain Name System. It is something that you will be using indirectly for your entire Internet career.

Moving Bits from One Place to Another

Modern networking is built around the concept of "layers of service." You start out trying to move bits from here to there, losing some along the way. This level consists of wires and hardware, and not necessarily very good wires. Then you add a layer of basic software to shield yourself from the problems of hardware. You add another layer of software to give the basic software some desirable features. You continue to add functionality and smarts to the network, one layer at a time, until

*Comer, Douglas, *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, Volumes I and II (Prentice Hall).

you have something that's friendly and useful. Well, let's start at the bottom and work our way up.

Packet Switch Networks

When you try to imagine what the Internet is and how it operates, it is natural to think of the telephone system. After all, they are both electronic, they both let you open a connection and transfer information, and the Internet is primarily composed of dedicated telephone lines. Unfortunately, this is the wrong picture, and causes many misunderstandings about how the Internet operates. The telephone network is what is known as a *circuit switched* network. When you make a call, you get a piece of the network dedicated to you. Even if you aren't using it (for example, if you are put on hold), your piece of the network is unavailable to others wishing to do real work. This leads to underutilization of a very expensive resource, the network.

A better model for the Internet, which may not instill confidence in you, is the U.S. Postal Service. The Postal Service is a *packet switched* network. You have no dedicated piece of the network. What you want to send is mixed together with everyone else's stuff, put in a pipe, transferred to another Post Office, and sorted out again. Although the technologies are completely different, the Postal Service is a surprisingly accurate analogy; we'll continue to use it throughout this chapter.

The Internet Protocol (IP)

A wire can get data from one place to another. However, you already know that the Internet can get data to many different places, distributed all over the world. How does that happen?

The different pieces of the Internet are connected by a set of computers called *routers*, which connect networks together. These networks are sometimes Ethernets, sometimes token rings, and sometimes telephone lines, as shown in Figure 3-1.

The telephone lines and Ethernets are equivalent to the trucks and planes of the Postal Service. They are means by which mail is moved from place to place. The routers are postal substations; they make decisions about how to route data ("packets"), just like a postal substation decides how to "route" envelopes containing mail. Each substation or router does not have a connection to every other one. If you put an envelope in the mail in Dixville Notch, New Hampshire, addressed to Boonville, California, the Post Office doesn't reserve a plane from New Hampshire to California to carry it. The local Post Office sends it to a substation; the substation sends it to another substation; and so on, until it reaches the destination. That is, each sub-station only needs to know what connections are available, and what is the best "next hop" to get a packet closer to its destination. Similarly, with the Internet: a router looks at where your data is going and decides where to send it next. It just decides which pipe is best and uses it.

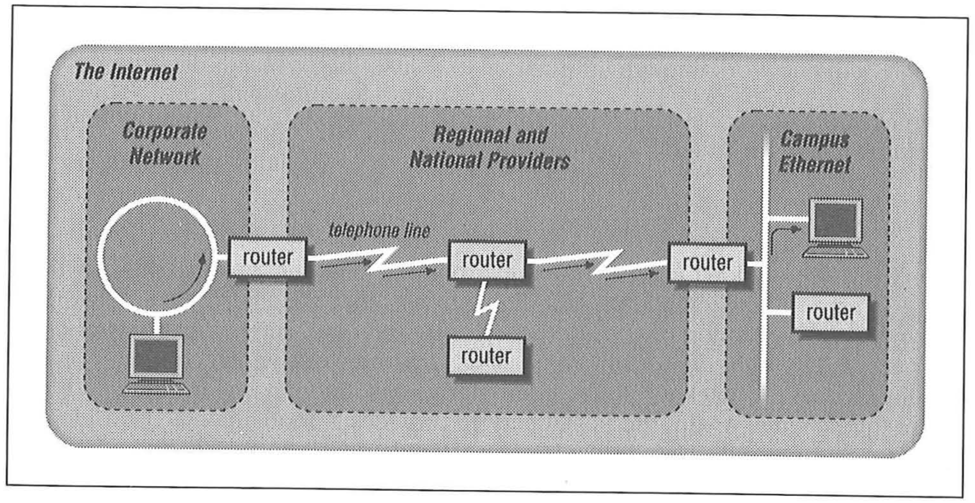

Figure 3-1: Internet hardware

How does the Net know where your data is going? If you want to send a letter, you can't just drop the typed letter into the mailbox and expect delivery. You need to put the paper in an envelope, write an address on it, and stick a stamp on it. Just as the Post Office has rules about how to use its network, the Internet has rules about how to use it. The rules are called *protocols*. The Internet Protocol (IP) takes care of addressing, or making sure that the routers know what to do with your data when it arrives. Sticking with our Post Office analogy, the Internet Protocol works just like an envelope (Figure 3-2).

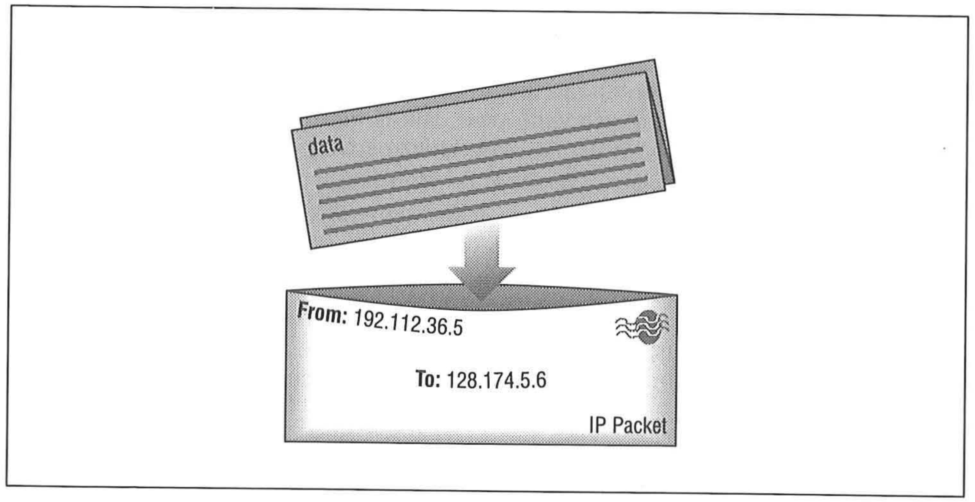

Figure 3-2: IP envelopes

Some addressing information goes at the beginning of your message; this information gives the network enough information to deliver the *packet* of data.

Internet addresses consist of four numbers each less than 256. When written out, the numbers are separated by periods like this:

```
192.112.36.5
128.174.5.6
```

(Don't worry; you don't need to remember numbers like these to use the network.) The address is actually made up of multiple parts. Since the Internet is a network of networks, the beginning of the address tells the Internet routers what network you are part of. The right end of the address tells that network which computer or *host* should receive the packet.* Every computer on the Internet has a unique address under this scheme. Again, the Postal Service provides a good analogy. Consider the address "50 Kelly Rd., Hamden, CT." The "Hamden, CT" portion is like a network address; it gets the envelope to the right local Post Office, the Post Office that knows about streets in a certain area. "50 Kelly Rd." is like the host address; it identifies a particular mailbox within the Post Office's service area. The Postal Service has done its job when it has delivered the mail to the right local office, and when that local office has put it into the right mailbox. Similarly, the Internet has done its job when its routers have gotten data to the right network, and when that local network has given the data to the right computer, or host, on the network.

For a lot of practical reasons (notably hardware limitations), information sent across IP networks is broken up into bite-sized pieces, called *packets*. The information within a packet is usually between 1 and about 1500 characters long. This prevents any one user of the network from monopolizing the network and allows everyone to get a fair shot. It also means that if the network isn't fast enough, as more people try to use it, it gets slower for everyone.

One of the amazing things about the Internet is that, on a basic level, IP is all you need to participate. It wouldn't be very friendly but, if you were clever enough, you could get some work done. As long as your data is put in an IP envelope, the network has all the information it needs to get your packet from your computer to its destination. Now, however, we need to deal with several problems:

- Most information transfers are longer than 1500 characters. You would be disappointed, indeed, if the Post Office would only carry postcards, but refused anything larger.
- Things can go wrong. The Post Office occasionally loses a letter; networks sometimes lose packets, or damage them in transit. Unlike the Post Office, we'll see that the Internet can deal with these problems successfully.

*Where the network portion ends and the host portion begins is a bit complicated. It varies from address to address based on an agreement between adjacent routers. Fortunately, as a user you'll never need to worry about this; it only makes a difference when you're setting up a network.

- Packets may arrive out of sequence. If you mail two letters to the same place on successive days, there's no guarantee that they will take the same route or arrive in order. The same is true of the Internet.

So, the next layer of the network will give us a way to transfer bigger chunks of information, and will take care of the many "distortions" that can creep in because of the network.

The Transmission Control Protocol (TCP)

TCP is the protocol, frequently mentioned in the same breath as IP, that is used to get around these problems. What would happen if you wanted to send a book to someone, but the Post Office only accepted letters? What could you do? You could rip each page out of the book, put it in a separate envelope, and dump them all in a mailbox. The recipient would then have to make sure the pages all arrived and paste them together in the right order. This is what TCP does.

TCP takes the information you want to transmit and breaks it into pieces. It numbers each piece so receipt can be verified and the data can be put back in the proper order. In order to pass this sequence number across the network, it has an envelope of its own which has the information it requires "written on it" (Figure 3-3). A piece of your data is placed in a TCP envelope. The TCP envelope is, in turn, placed inside an IP envelope and given to the network. Once you have something in an IP envelope, the network can carry it.

On the receiving side, a TCP software package collects the envelopes, extracts the data, and puts it in the proper order. If some are missing, it asks the sender to retransmit them. Once it has all the information in the proper order, it passes the data to whatever application program is using its services.

This is actually a slightly utopian view of TCP. In the real world not only do packets get lost, they can also be changed by glitches on telephone lines in transit. TCP also handles this problem. As it puts your data into an envelope, it calculates something called a *checksum*. A checksum is a number that allows the receiving TCP to detect errors in the packet.* When the packet arrives at its destination, the receiving TCP calculates what the checksum should be and compares it to the one sent by transmitter. If they don't match, an error has occurred in the packet. The receiving TCP throws that packet away and requests a retransmission.

*Here's a simple example, if you're interested. Let's assume that you're transmitting raw computer data in 8-bit chunks, or bytes. A very simple checksum would be to add all of these bytes together. Then stick an extra byte onto the end of your data that contains the sum. (Or, at least, as much of the sum as fits into 8 bits.) The receiver makes the same calculation. If any byte was changed during transmission, the checksums will disagree, and you'll know there was an error. Of course, if there were two errors, they might cancel each other out. But more complicated computations can handle multiple errors.

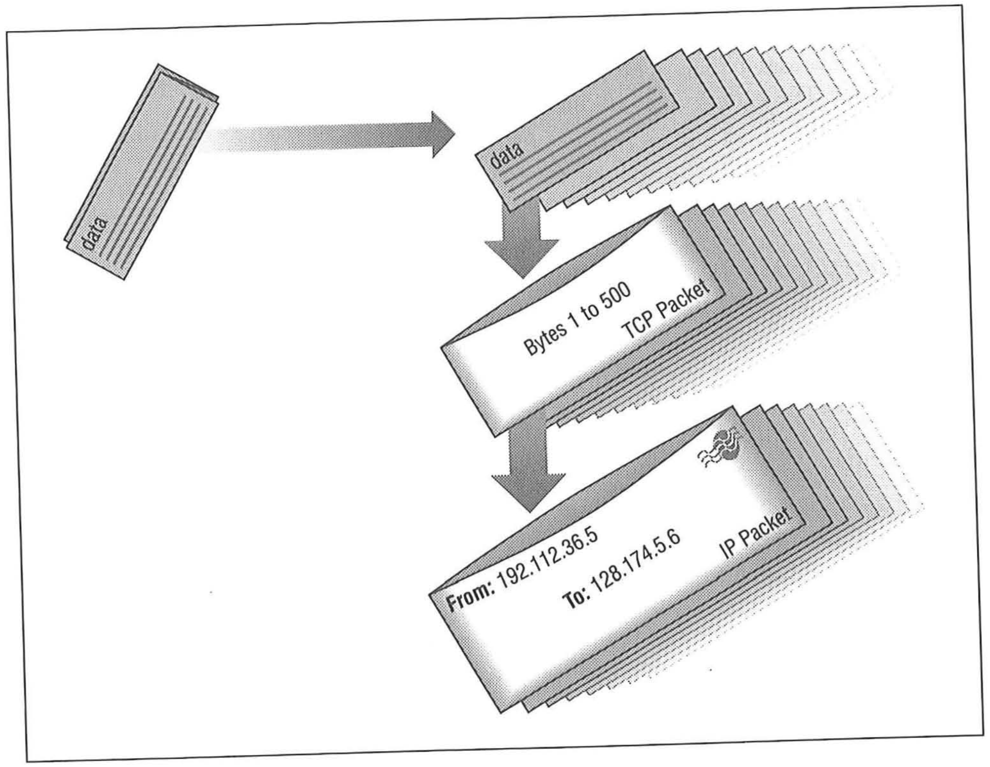

Figure 3-3: TCP packet encapsulation

Other Transmission Protocols

TCP creates the appearance of a dedicated wire between the two applications, guaranteeing that what goes in one side comes out the other. You don't have a dedicated link between the sender and receiver (other people can use the same routers and network wires in the gaps between your packets); but, for all practical purposes, it looks like you do.

Ideal as this may sound, it is not the best approach for every program to use. Setting up a TCP connection requires a fair amount of overhead and delay; if this machinery isn't needed, it's better not to use it. If all the data you want to send will fit in one packet and you don't particularly care to guarantee delivery, TCP may be overkill.

It turns out that there is another standard protocol that does away with this overhead. This protocol is called the *user datagram protocol* or *UDP*. It is used instead of TCP in some applications; that is, instead of wrapping your data in a TCP envelope and putting that inside an IP envelope, the application puts your data into a UDP envelope, which goes in the IP envelope.

UDP is a lot simpler than TCP because it doesn't worry about missing packets, keeping data in the right order, or any of those niceties. UDP is used for programs that only send short messages, and can just resend the message if a response does

not come in a short time. For example, assume that you're writing a program that looks up phone numbers in a database somewhere else on the network. There is no reason to set up a TCP connection to transmit 20 or so characters in each direction. You can just put the name into one UDP packet, stick that into an IP packet, and send it. The other side of the application gets the packet, reads the name, looks up the phone number, puts that into another UDP packet, and sends it back. What happens if the packet gets lost along the way? Your program has to handle that: if it waits too long without getting a response, it justs sends another request.

Making the Network Friendly

Now that we have the ability to transfer information between places on the network, we can start working on making the Internet more friendly. This is done by having software tailored to the task at hand, and using names rather than addresses to refer to computers.

Applications

Most people don't get really excited about having a guaranteed bit stream between machines, no matter how fast the lines or exotic the technology that creates it. They want to use that bit stream to do something useful, whether that is to move a file, access some data, or play a game. Applications are pieces of software that allow this to happen easily. They are yet another "layer" of software, built on top of the TCP or UDP services. Applications give you, the user, a way to do the task at hand.

What an application is varies greatly. Applications can range from home-grown programs to proprietary programs supplied by a vendor. There are three "standard" Internet applications: remote login, file transfer, and electronic mail, as well as other commonly used but not standardized applications. Chapters 5 through 14 of this book describe how to use most of the common Internet applications.

One problem with talking about applications is that the application's appearance to you is determined by your local system. The commands, messages, prompts, etc., may be slightly different on your screen than in the book or on someone else's screen. So, don't worry because the book says the message is "connection refused" and the error message you receive is "Unable to connect to remote host: refused"; they are the same. Try and distill the essence of the message, rather than matching the exact wording. And don't worry if some of the commands are named slightly differently; most of the applications have reasonable "help" facilities that will let you figure out the right command.

The Domain Name System

Fairly early on, people realized that addresses were fine for machines communicating with machines, but humans preferred names. It is hard to talk using addresses (who would say, "I was connected to 192.112.36.5 yesterday and. . . "?), and even harder to remember them. Therefore, computers on the Internet were given names for the convenience of their human users. The preceding conversation becomes "I was connected to the NIC* yesterday and . . . ". All of the Internet applications let you use system names, rather than host addresses.

Of course, naming introduces problems of its own. For one thing, you have to make sure that no two computers that are connected to the Internet have the same name. You also have to provide a way to convert names into numeric addresses. After all, names are just fine for people; but the computers really prefer numbers, thank you. You can give a program a name, but it needs some way to look that name up and convert it into an address. (You do the same thing whenever you look someone up in the phone book.)

In the beginning, when the Internet was a small folksy place, dealing with names was easy. The NIC (*Network Information Center*) set up a registry. You would send in a form, electronically of course, and they would maintain a file of names and addresses. This file, called the *hosts* file, was distributed regularly to every machine on the network. The names were simple words, every one chosen to be unique. If you used a name, your computer would look it up in the file and substitute the address. It was good.

Unfortunately, when the Internet went forth and multiplied, so did the size of the file. There were significant delays in getting a name registered, and it became difficult to find names that weren't already used. Also, too much network time was spent distributing this large file to every machine contained in it. It was obvious that a distributed, online system was required to cope with the rate of change. This system is called the *Domain Name System* or *DNS*.

The Domain System Structure

The Domain Name System is a method to administer names by giving different groups responsibility for subsets of the names. Each level in this system is called a *domain*. The domains are separated by periods:

```
ux.cso.uiuc.edu
nic.ddn.mil
yoyodyne.com
```

There can be a variable number of domains within the name but practically there are usually five or less. As you proceed left to right through the domains, the number of names contained in the group gets bigger.

In the first line above (**ux.cso.uiuc.edu**), **ux** is the name of a host, a real computer with an IP address (Figure 3-4). The name for that computer is created and maintained by the **cso** group, which happens to be the department where the computer resides. The department **cso** is a part of the University of Illinois at Urbana Champaign (**uiuc**). **uiuc** is a portion of the national group of educational institutions (**edu**). So the zone **edu** contains all computers in all U.S. educational institutions; the zone **uiuc.edu** contains all computers at the University of Illinois; and so on.

*A Network Information Center is a repository for information about a network.

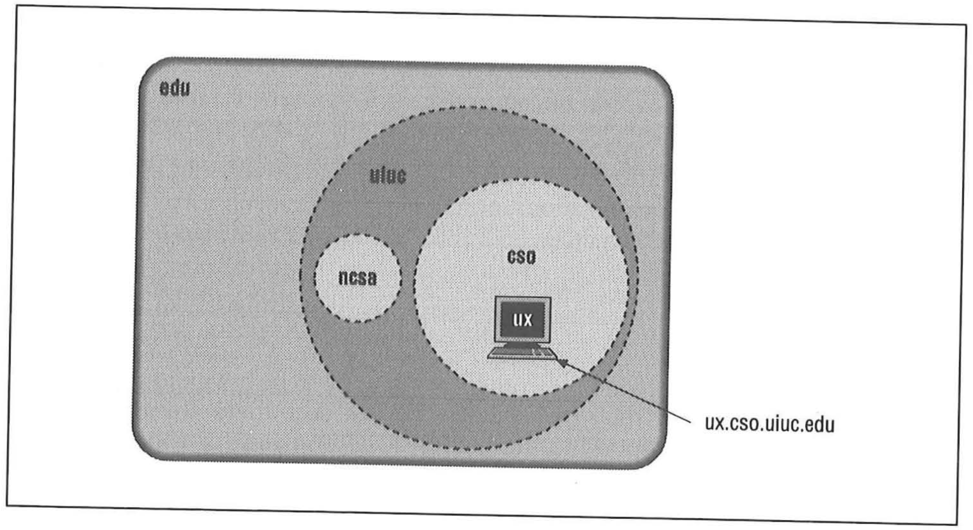

Figure 3-4: Domain authority

Each group can create or change whatever lies within it. If **uiuc** decided to create another group called **ncsa**, it could do so without asking anyone's permission. All it has to do is add the new names to its part of the worldwide database, and sooner or later everyone who needs to know will find out about the new name. Similarly, **cso** can buy a new computer, assign it a name, and add it to the network without asking anyone's permission. If every group from **edu** on down plays by the rules and makes sure that the names it assigns are unique, then no two systems anywhere on the Internet will have the same name. You could have two machines named **fred**, but only if they are in different domains (for example, **fred.cso.uiuc.edu** and **fred.ora.com**).

In practice, being the name administrator for a group requires certain skills, and is not fun. Therefore, at some level around the enterprise level (**uiuc**) or one level below it, there is a person who is responsible for maintaining all lower levels. There is some locally defined procedure for requesting that a name get created or changed.

It's easy to see where domains and names come from within an organization like a university or a business. However, where do the "top level" domains like **edu** come from? They were created by *fiat* when the domain system was invented. Originally, there were six highest level domains (see Table 3-1).

Table 3-1: Original High-level Domains

Domain	Usage
com	For commercial organizations (i.e., businesses)
edu	Educational organizations (universities, secondary schools, etc.)
gov	Governmental organizations, non-military
mil	Military (army, navy, etc.)
org	Other organizations
net	Network resources

As the Internet was a worldwide network, there needed to be a way to give foreign countries responsibility for their own names. To this end, there are a set of two letter domains which correspond to the highest level domains for countries. Since **ca** is the country code for Canada, a Canadian computer might be named:

```
hockey.guelph.ca
```

There are almost 300 country codes, about 100 of which have some kind of computer networking. There is a list of the country codes in Appendix B, *International Network Connectivity*, in case you want to see where mail you received came from.

It's worth noting that the U.S. has its own country code, although it isn't used too often; in the U.S., most network sites use the "organizational" domains (like **edu**), rather than the "geographical" domains (like **va.us**—Virginia). However, you will see both kinds of names. One computer may even have both kinds of names just for completeness. There's no way to "convert" between organizational names and geographical names. For example, even though **uxc.cso.uiuc.edu** happens to be in Urbana, Illinois, U.S.A., there is *not* necessarily a name **uxc.urbana.il.us**. Even if there is, they aren't necessarily the same computer.

Domain Name Lookup

Now you know how domains relate to each other and how a name gets created. Now you might just wonder how to use this marvelous system. You use it automatically, whenever you use a name on a computer that knows about it. You never need to look a name up "by hand," or give some special command to find out about some name, although you can if you want. All computers on the Internet can use the domain system, and most do.

When you use a name like **ux.cso.uiuc.edu**, the computer needs to turn it into an address. To do so, it starts asking DNS servers for help, starting at the right end and working left. First, it asks the local DNS servers to look up the address. At this point, there are three possibilities:

- The local server knows the address, because the address is in the local server's part of the worldwide database. For example, if you're in the computer science department of the University of Illinois, your local server probably has information about the computers in your department.

- The local server knows the address because someone else has asked for the same address recently. Whenever you ask for an address, the DNS server keeps it on hand for a while, just in case someone else wants the same address later; this makes the system a lot more efficient.
- The local server doesn't know the address, but it knows how to find out.

How does the local server find out? Its software knows how to contact a *root* server. This is the server that knows the addresses of name servers for the highest level (rightmost) zone (**edu**). It asks the root server for the address of the computer responsible for the **edu** zone. Having that information, it contacts that server and asks that server for the address of the **uiuc** server. Your software then contacts that computer and asks for the address of the server for **cso**. Finally, it contacts that machine and gets the address of **ux**, the host that was the target of the application.

A few computers are still configured to use the old-style *hosts* file. If you find yourself on one of these, you may have to ask its administrator to look up the address you need by hand (or look it up yourself); then the administrator will have to add the machine you want to contact to the local *hosts* file. While you're doing this, you can hint that the administrator *really ought to install* the DNS software so you won't have to do this again.

Domain Name System Hints

There are a few common misconceptions that you may encounter dealing with names. Here are a few we can dispel now:

- The pieces of a domain-style name tell you who is responsible for maintaining the name. It may not tell you anything about who maintains the computer corresponding to that IP address, or even (despite the country codes) where that machine is located. It would be perfectly legal for me to have the name **oz.cso.uiuc.edu** (part of the University of Illinois' name space) point to a machine in Australia. It isn't normally done, but it could be.
- The pieces of a domain name don't even necessarily tell you what network a computer is located on. Domain names and networking often overlap, but there's no necessary connection between them; two machines in the same domain may not be on the same network. For example, the systems **uxc.cso.uiuc.edu** and **ux1.cso.uiuc.edu** may be on different networks. Once again, domain names only tell you who is responsible for the domain.
- A machine can have multiple names. This is especially true of machines that offer services, where the service may be moved to a different computer in the future. My Sun workstation may be known by **ek.cso.uiuc.edu**. It also might be the computer where you can go to get publicly available files at the University of Illinois. So it might also have the name **ftp.uiuc.edu** (**ftp** being the name of the file moving program). Some time in the future, this service might be moved to some other computer. When this happens, the name **ftp.uiuc.edu** would move along with the service (my computer gets to keep its old name **ek.cso.uiuc.edu**). People wanting the particular service use the same name regardless of which computer is providing the service. Names that symbolically

refer to a service are called "canonical names" or *cnames*. You will see them frequently as you wander about the Internet.

- Names aren't necessary for communication. Unless the error message you receive is "host unknown," the name worked fine. A message like "host unknown" means your system could not translate the name you gave into an address. Once your system has the address in hand, it never uses the name again.
- It is better to remember names than addresses. Some people feel that the name system is "just one more thing to go wrong." The problem is that an address is tied to a network. If the computer providing a service is moved from one building to another, its network and hence its address will likely change. The name needn't change. When the administrator assigns the new address, he only needs to update the name record so that the name points to the new address. Since the name still works, you don't particularly care if the computer or function has changed locations.

The Domain Name System may sound complicated, but it's one of the things that make the Internet a comfortable place to live. If you don't like the periods wandering around, forget about what they mean: they're just names. However, pretty soon you'll start realizing, "yes, this resource is at the University of Virginia; this person works for IBM in Germany; this is the address for reporting bugs in Nutshell Handbooks (**nuts@ora.com**)" and so on. The real advantage of the domain system is that it breaks the gigantic worldwide Internet into a bunch of manageable pieces. Although hundreds of thousands of computers are "on the Net," they're all named; and the names are organized in a convenient, perhaps even rational way, making it easier for you to remember the ones you need.

CHAPTER FOUR

What's Allowed on the Internet?

In earlier chapters, I told you very generally what the Internet is good for, where it came from, and how it works. Now it's time to get to the real nitty-gritty. We will talk about what you are allowed to do on the network; in the next chapter, we will start discussing "how to do it."

What you are allowed to do is a very complex issue. It is influenced by law, ethics, and politics. How these inter-relate and which is paramount vary from place to place. The Internet isn't a network—it's a network of networks—and each network may have its own policies and rules. Lest you should give up before starting, the rules are reasonably uniform, and you'll be safe if you keep a few guidelines in mind. Fortunately, these guidelines aren't terribly restrictive. As long as you stay within those guidelines, you can do whatever you want. If you feel yourself getting near the edges, contact your network provider to determine exactly what is allowed and what isn't. It may be possible to do what you want, but it's your responsibility to find out. Let's look at the issues so you can see where the borders are.

Legal Implications

Three areas of the law affect the Internet:

- Federal subsidies pay for large sections of the Internet. These subsidies exclude purely commercial use.
- The Internet is not just a nationwide network, but a true global network. When shipping anything across a national boundary, including bits, export laws come into effect and local laws change.
- Whenever you are shipping software (or, for that matter, ideas) from one place to another, you must consider intellectual property and license issues.

First, let's deal with the federal dollars.

The Whole Internet

User's Guide & Catalog

Ed Krol

O'Reilly & Associates, Inc.
103 Morris Street, Suite A
Sebastopol CA 95472
(800) 998-9938 • (707) 829-0515
Email: *nuts@ora.com* or *uunet!ora!nuts*

The Whole Internet User's Guide & Catalog
by Ed Krol

Editor: Mike Loukides

Printing History:

September 1992:	First Edition.
November 1992:	Minor corrections.
December 1992:	Minor corrections.

ISBN: 1-56592-025-2

TABLE OF CONTENTS