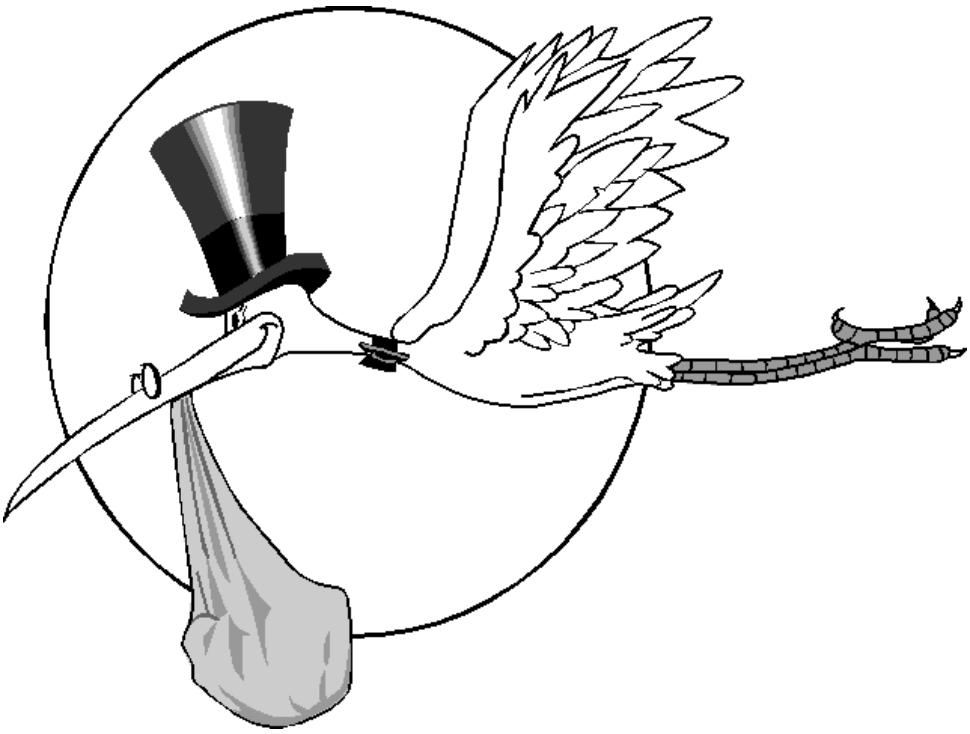
TCP/IP

Transmission Control Protocol / Internet Protocol



Gary A. Donahue

December 10, 1996

lordgad@planet.net

Comments welcome

Table Of Contents

Introduction

"In the beginning the Universe was created. This has made a lot of people very angry and been widely regarded as a bad move."

Douglas Adams
The Restaurant at the End of the Universe

Introduction

When you send a package across the country, do you really care how it gets there? Not usually (unless it doesn't get there!). All you really care about is whether or not it gets there in one piece, and in a reasonable amount of time. The Internet pretty much works by the same principal. You send a packet from point A to point B, and you don't care how it gets there, just that it does, and in one piece.

In this picture you can see one of the poor slobs who's job it is to figure out where the packages are supposed to go. This person is roughly analogous to a router on the Internet.

When you send a package to another state you aren't really concerned with the fact that your shipping company has to figure out where the destination address is and how to get it there. Similarly when you view a web page in another state, you don't really care how many routers your packets have to go through to get there and back. You don't care how those routers work either.

History

In 1969 the Defense Advanced Research Projects Agency (DARPA) funded a research and development project to create an experimental packet switching network. This network, called the ARPANET, was built to study techniques of reliable vendor-independent data communications. Many techniques of modern data communications were developed in the ARPANET.

The ARPANET was so successful, that many of the organizations attached to it began to use it for daily communications. In 1975, the ARPANET changed from an experiment to an operational network. This did not mean that development stopped however; the basic TCP/IP protocols were developed after the ARPANET was operational.

In 1983 the military adopted the TCP/IP protocols as standard (MIL-STD), and all hosts attached to the network were required to convert to the new protocols (imagine that cut-over nightmare!). To make the transition a little smoother, DARPA funded Bolt, Beranek, and Newman (BBN) to implement TCP/IP in Berkeley UNIX. Thus began the marriage UNIX and TCP/IP.

In 1983, the old ARPANET was divided into the MILNET, the unclassified portion of the Defense Data Network (DDN), and a new, smaller ARPANET. These two networks together were now referred to as the Internet. In 1990, the ARPANET formally ceased to exist, however the Internet lives on, and brother is it a monster.

Features

Why is are the TCP/IP protocols so popular? Because they met the need of world wide communications at a time when such a thing was in it's infancy. TCP/IP also includes some important features that help meet the needs of those using it.

- Open protocol standards, freely available and developed independently from any specific hardware or operating system. Because of this, TCP/IP is ideal for connecting different type of hardware and OS's, even if they are not connected to the Internet. Most of the information regarding these standards is found in Request For Comments (RFC's). If you would like to know more about this process, get a copy of RFC 1310; The Internet standards process.

- Independence from specific network hardware. This means that the TCP/IP suite can be run over Ethernet, token ring, dial-up lines, X.25, or just about any other kind of physical media.

- A common addressing scheme that allows any TCP/IP device to uniquely address any other device in the entire network, even if that network is as large as the Internet.

- Standardized high level protocols for consistent, widely available user services. Examples of these include FTP, TELNET, finger, and http (the World Wide Web).

The OSI Model vs. the TCP/IP Stack

If you've ever learned about networking or protocols in depth, you've no doubt come across references to the Open Systems Interconnect (OSI) reference model. This model is the common reference for data communications, and is the standard by which networking protocols are designed. Many protocols do not strictly adhere to this model, instead we compare these models with the OSI standard. This way communications professionals have a common ground on which they may communicate about differing data communications subjects.

The OSI model is divided into seven distinct layers. Each layer does not define a protocol, rather it describes a function within data communications. These functions may be performed by any number of protocols.

Level	*Layer Name*	*Description*
7	Application Layer	Consists of application programs that use the network.
6	Presentation Layer	Standardizes data presentation to the applications.
5	Session Layer	Manages sessions between applications.
4	Transport Layer	Provides end-to-end error detection and correction.
3	Network Layer	Manages connections across the network for the upper layers.
2	Data Link Layer	Provides reliable data delivery across the physical link.
1	Physical Layer	Defines the physical characteristics of the network media

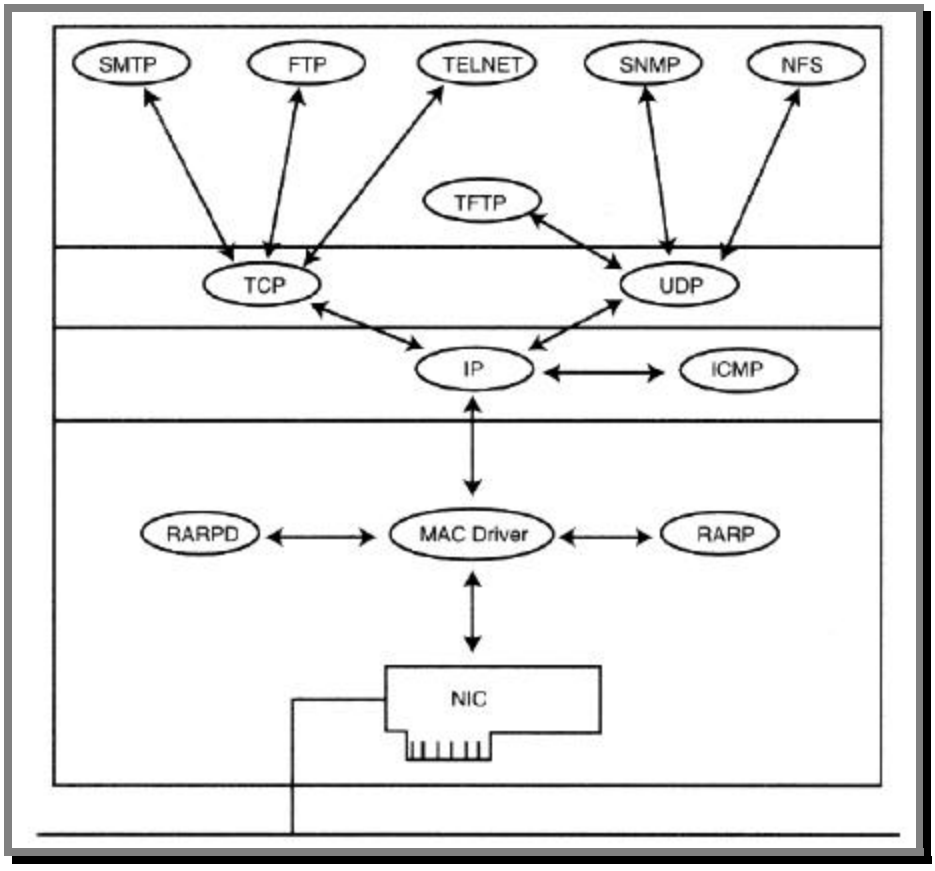
Each layer relies on the layer beneath it. As a general rule, data is created at the top layer then passed down layer by layer until the physical layer physically sends the information across the network media. As each layer passes the data downward, that layer adds it's own header to the packet. Each layer considers everything it receives from the layer above as data. The process of wrapping that data with the layers header is called encapsulation.

When the packet is received at the other end, the headers are stripped at each layer and the data is passed upwards through the stack until it reaches it's destination layer.

TCP/IP does not strictly adhere to this model, but the TCP/IP model is comparable in the way in which it functions. The TCP/IP model has only four layers (In actuality no one can agree on this, so what you see here is what I've seen the most).

Level	*Layer Name*	*Description*
4	Application Layer	This the user level. Protocols like TELNET, FTP, and SMTP reside here.
3	Transport Layer	Data delivery is handled here. This is where the TCP and UDP protocols reside.
2	Internet Layer	Provides the basic delivery system for packets (datagrams) This is the layer where IP resides.
1	Network Access Layer	These protocols provide a means for the system to deliver data to other devices on the directly attached network.

A simpler way of visualizing what relies on what in the grand scheme of IP protocols is shown in this graphic:



In fact, forget all about the OSI model stuff. Unless you are taking a test (none given here), or you are deep into data communications, you just don't need to remember that stuff. The image above is useful however, as you need to understand why some things rely on other things. We

will use the TCP/IP model to subdivide the protocols contained herein, but you do not need to remember the model itself, just the general idea.

Routers versus Gateways

Just a quick word about the terms router and gateway.

In traditional TCP/IP terminology, there are only two types of network devices, hosts and gateways. Gateways forward packets between networks, and hosts do not.

Using current terminology a gateway moves data from one protocol to another, while a router moves data from one network to another.

Many books on TCP/IP use the term gateway exclusively. In real life (Which rarely follows the manual), the term router is used more often than not to refer to any device connecting multiple networks together.

This document will use both terms interchangeably. This is due mostly to the fact that I used many books as sources, and I use whichever term that book used in that particular passage.

The Protocols

"The most overlooked advantage to owning a computer is that if they foul up there's no law against whacking them around a little."

Porterfield

Introduction

As we stated earlier, we will be dividing the TCP/IP suite of protocols into their respective TCP/IP model layers. Again you don't need to memorize the layers themselves, just understand what's going on between the protocols and how they relate to each other.

Network Access Layer

Level	*Layer Name*	*Description*
4	Application Layer	This the user level. Protocols like TELNET, FTP, and SMTP reside here.
3	Transport Layer	Data delivery is handled here. This is where the TCP and UDP protocols reside.
2	Internet Layer	Provides the basic delivery system for packets (datagrams) This is the layer where IP resides.
1	Network Access Layer	These protocols provide a means for the system to deliver data to other devices on the directly attached network.

The lowest level of the TCP/IP model, the Network Access Layer, is usually only seen as a device driver and some supporting files. The end user rarely has to deal with anything pertaining to this level. Instead, it's the programmer that must make TCP/IP available to a new operating system or computer must be aware of what goes on in this layer. When you load the TCP/IP driver for Win-95 or NT or whatever, you are loading the Network Access Layer protocols.

Internet Layer

Level	*Layer Name*	*Description*
4	Application Layer	This the user level. Protocols like TELNET, FTP, and SMTP reside here.
3	Transport Layer	Data delivery is handled here. This is where the TCP and UDP protocols reside.
2	Internet Layer	Provides the basic delivery system for packets (datagrams) This is the layer where IP resides.
1	Network Access Layer	These protocols provide a means for the system to deliver data to other devices on the directly attached network.

The second layer from the bottom, the Internet Layer, is where the Internet Protocol (IP) resides. IP is the very core of the TCP/IP suite. In fact all protocols above this layer use IP to deliver data. Regardless of the destination, all data flows through IP.

Internet Protocol

The internet Protocol is the basic building block of the internet. Some of the functions of IP include:

- defining the datagram (or packet), which is the basic unit of transmission in the Internet.
- defining the Internet addressing scheme
- moving data between the Network Access Layer and the Host-to-Host Transport Layer
- routing datagrams to remote hosts
- performing fragmentation and re-assembly of datagrams

IP is a connectionless protocol which simply means that there is no control information exchanged to establish a connection before sending data. Connectionless vs. Connection-oriented protocols will be covered in more detail later. Suffice to say that IP relies on protocols in other layers to make connections and perform error correction if such things are required.

Remember the ARPANET? Well the TCP/IP protocols were designed to transmit data over the ARPANET. The ARPANET was designed to be a packet switching network. A packet is a block of data that carries with it the information necessary to deliver it — much like a letter at the post office. The envelope has the delivery and return address on it, and the important data is contained inside.

Imagine now that you want to send your manifesto to Time Magazine, but you don't have a box big enough. You take your manifesto and break it into it's chapters. You take each chapter and put it in it's own envelope and send them separately. You bring them all to the Post Office in order and ship them. Once they leave your possession you don't know what happens to them. Each letter could go a different route and you wouldn't know the difference or care. When Time Magazine gets all the pieces, someone opens all the envelopes, puts the chapters back in order and has the whole document in it's original intended form. That's the idea behind a packet switching network.

The Network uses the destination address in the datagram to switch packets from one network to another, hopefully getting closer and closer to the destination network, and finally, the destination host.

The datagram is the packet format defined by the Internet Protocol. If you ever experience the joy of decoding packets by hand, here is the layout of the IP datagram:

← BYTES →

<table>
<tr><th colspan="2">0</th><th>1</th><th colspan="2">2</th><th>3</th></tr>
<tr><td colspan="2">Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragmentation Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source Address</td></tr>
<tr><td colspan="6">Destination Address</td></tr>
<tr><td colspan="5">Options</td><td>Padding</td></tr>
<tr><td colspan="6">data begins here . . .</td></tr>
</table>

Don't worry about the layout in detail, just understand that there are fields within the datagram header, and that the source and destination address information is found there.

Datagram Fragmentation

Routers can connect multiple networks and multiple network types. When a router connects a token ring network with an Ethernet network (Both of which can run TCP/IP), there is a problem of communication.

Remember that manifesto you mailed earlier? Well here's why TCP/IP needs to do the same thing:

Each type of network has what is called a maximum transmission unit (MTU), which is the size of the largest packet that can be sent over the wire, using that protocol. If the MTU for one network connected to a router is 500 bytes and the MTU for another network connected to the same router is 2000 bytes, then packets originating on the later network will have to be broken up into 500 byte fragments, sent, then re-assembled on the destination host. Only the final destination will reassemble the packets, never the routers between the host and the destination.

The format of the fragmented datagrams is the same as any other datagram. So how does the destination host know that the fragments *are* fragments and need to be reassembled? And how does the destination host know what order to reassemble them in? In the IP header, the field *Identification* contains the same number in every fragmented piece, thereby identifying what datagram the fragment belongs to. The *Fragmentation Offset* field tells what piece of the datagram this packet is. The field *Flag*s has a *More Fragments bit* that tells IP it has assembled all of the datagram fragments.

Passing Datagrams to the Transport Layer

Once the datagrams are received and reassembled if necessary, they are passed upward to the Transport Layer. What protocol they are passed to within the transport layer is determined by the *Protocol Number* in the IP header. This will be covered in more detail later.

Internet Control Message Protocol (ICMP)

A very important part of the IP protocol is ICMP. ICMP performs the following control, error reporting, and informational functions for TCP/IP:

Flow control

When datagrams arrive too fast for the destination to keep up with the demand, a message called a Source Quench is sent to the originating host. This message tells the offending host to temporarily stop sending datagrams.

Detecting unreachable destinations

When a destination is unreachable an ICMP message is sent to the source host advising of the problem. In the case of a host being unreachable, the message is sent by an intermediate gateway or router. If the unreachable destination is a port (covered later) the ICMP message is sent by the destination host.

Redirecting routes

A gateway sends the ICMP Redirect Message to tell a host that the destination network is better reached using another gateway. This only happens on a network with more than one gateway, such as a network with more than one router on it.

Checking remote hosts

A host can send an ICMP echo packet to any other host to see if the other host has it's Internet Protocol running. If the other system is alive, that system sends the same packet back to the source. The common command to do this is **ping**.

Transport Layer

Level	*Layer Name*	*Description*
4	Application Layer	This the user level. Protocols like TELNET, FTP, and SMTP reside here.
3	Transport Layer	Data delivery is handled here. This is where the TCP and UDP protocols reside.
2	Internet Layer	Provides the basic delivery system for packets (datagrams) This is the layer where IP resides.
1	Network Access Layer	These protocols provide a means for the system to deliver data to other devices on the directly attached network.

The Host-to-Host Transport Layer, or more commonly the Transport Layer, is the home of the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP provides reliable delivery of data with end-to-end error detection and correction. UDP provides low-overhead connectionless delivery of data. Both protocols provide delivery between the Internet Layer and the Application Layer. It is up to the programmer to decide which protocol best meets the needs of the application.

User Datagram Protocol (UDP)

UDP gives application programs direct access to a datagram delivery service, like the service that IP provides. This direct access allows applications to exchange messages over the network with a minimum of overhead.

UDP is classified as a connectionless, unreliable protocol. This means that the protocol does no error checking, and in fact does not care if the messages are received after they are sent. There are no techniques built into this protocol to ensure proper delivery of datagrams. UDP uses 16 bit source port and destination port numbers (covered in detail later) to ensure the data is passed to the proper application. Here is the layout of the UDP datagram:

Again, there is no need to memorize this chart, it is given for you to reference. Remember that the data in a UDP datagram contains all the header information from the application layer too!

If UDP is considered unreliable then why use it? There are many reasons. Suppose the data to be sent is so small that the overhead of a connection negotiation and error correction far exceeds the size of the data itself, then bandwidth is wasted as well as time and resources. Higher protocols that have their own error correction do not need another layer of error correction. To reduce redundancy in this case, UDP would be a logical choice. Another example is the query-response type of application. In this model, a query is submitted, and if no answer is received in a certain period of time, then another query is sent. There is no need for a

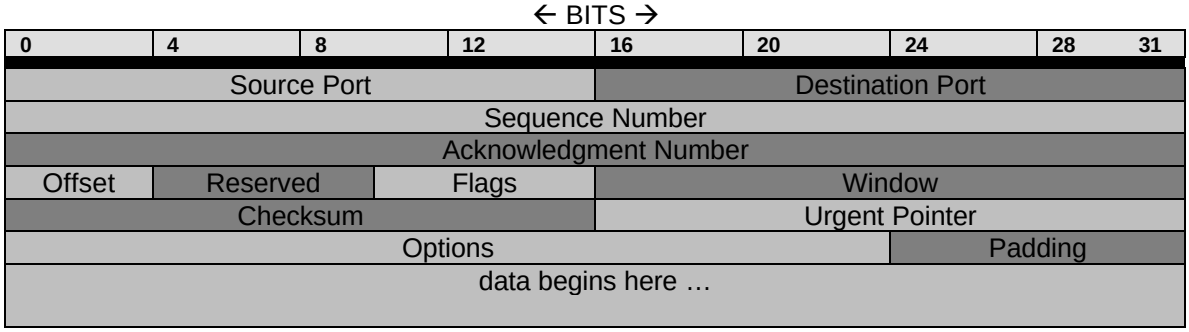
connection. Protocols such as SNMP, where overhead must be kept to a strict minimum use the UDP protocol. One way communication often, but by no means always, incorporates UDP.

Some of the applications that use UDP include: DNS, SNMP, and RIP. If you think about it, these applications do not deliver application critical information, nor does it really matter if the data is sent perfectly every time. This makes UDP the intelligent choice for these apps.

Transmission Control Protocol (TCP)

TCP is a reliable, connection oriented byte-stream protocol. Connection oriented means that a logical end to end connection is established between the communicating hosts. Before communications actually takes place, handshaking is done to establish this link. The fact that error correction is also present for all data sent makes this protocol reliable. The fact that data is viewed as a byte stream instead of in packets makes this a byte-stream protocol. If you remember anything about TCP, remember that it is connection oriented, and reliable.

Here is the datagram layout for TCP. Remember this is for reference, not for you to bang your head over trying to memorize.



As you can see, compared to the UDP datagram, there is considerably more information that is sent with each TCP datagram. Reliability comes with a price. Here we can also see that the additional data needed for TCP to be as reliable as it is, makes for a much larger datagram.

TCP makes sure that data is sent and received in the same order, hence the sequence number in the datagram. Since reliability is a must in TCP, the acknowledgment field returns the sequence number of the last byte received at the remote host.

Part of what the Transport Layer protocols must do is pass data to the proper application in the Application Layer. This is done by the use of Source and Destination Port numbers. This will be covered in detail later.

The details of how TCP works can be rather involved, and has been kept to a minimum here. Just remember that TCP is reliable and connection oriented, while UDP is unreliable and connectionless.

Some of the applications that use TCP instead of UDP are: FTP, TELNET, and SMTP. These applications all deliver data that needs to be delivered properly. If you are downloading a file via FTP, you certainly want that file to be received in the right order and intact. TCP fits the bill nicely for these apps.

Application Layer

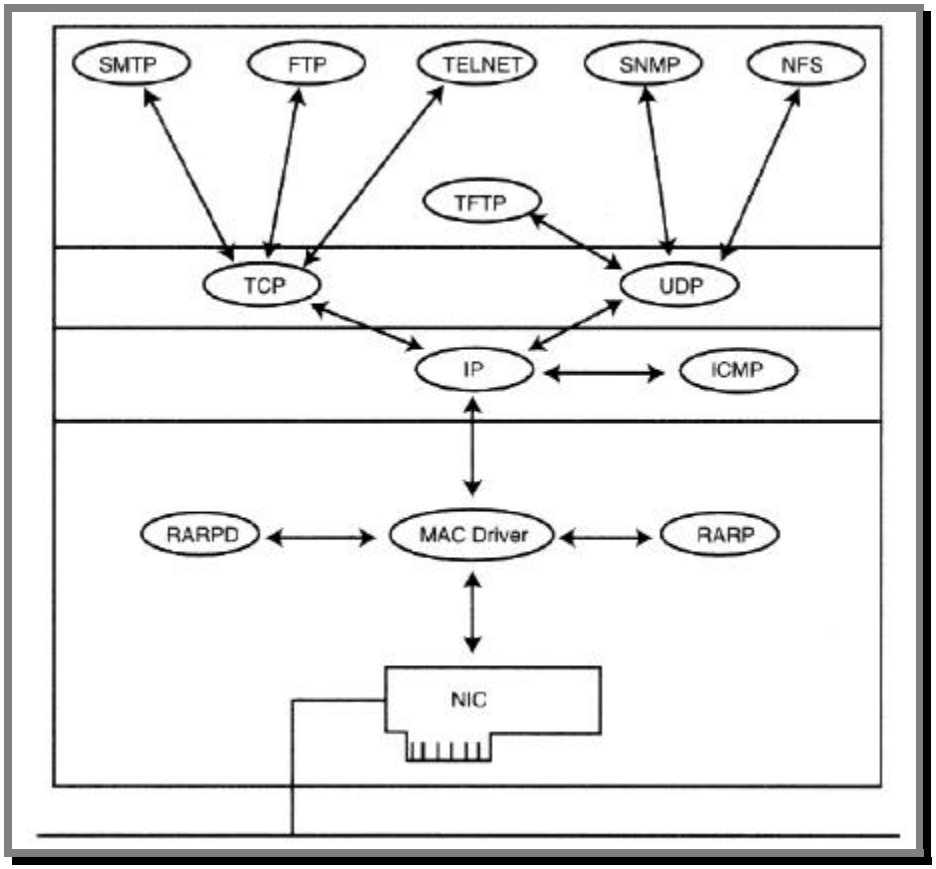
Level	*Layer Name*	*Description*
4	Application Layer	This the user level. Protocols like TELNET, FTP, and SMTP reside here.
3	Transport Layer	Data delivery is handled here. This is where the TCP and UDP protocols reside.
2	Internet Layer	Provides the basic delivery system for packets (datagrams) This is the layer where IP resides.
1	Network Access Layer	These protocols provide a means for the system to deliver data to other devices on the directly attached network.

This is the top of the TCP/IP architecture. This layer provides user services, which use the Transport Layer to deliver data. Some of the more widely know applications in this layer are: TELNET, FTP, SMTP, DNS, RIP, SNMP, and NFS. Note that not all applications require a user to run. RIP and SNMP for the most part run without user intervention.

It is not the purpose of this document to go into detail about the applications of TCP/IP so there will not be any detail regarding them here.

Summary

Remember this drawing?



Here we can see the relationship of the protocols through the layers. On the top we can see all of the applications which communicate to the Transport Layer via TCP or UDP. That layer connects with the Internet Layer via IP, which connects to the Network Access Layer on the bottom.

In summary, the Application Layer is roughly where the applications that a user would use reside. They are the highest level protocols in the stack, and as far as the user is concerned, probably the most useful.

The next layer down is the Transport layer, which contains UDP and TCP.

The layer second from the bottom is the Internet Layer, where IP resides.

Finally the bottom layer is the Network Layer, where the device drivers reside.

How Does It All Work?

"Any sufficiently advanced technology is indistinguishable from magic."

Arthur C. Clarke,
Technology and the Future

Introduction

How does it all work? Frankly at times I'm amazed that it does. When you consider all that goes on and that it works from any host to any other host in the world, well to me that's pretty amazing.

In order for the data to properly travel from one host to another, is necessary to move that data through probably more than one network, to the right host on the destination network, then to the right process running on that host. This is accomplished via three schemes:

Addressing IP addresses, which uniquely identify every host on the Internet, ensure that the data goes to the right host.

Routing Gateways or Routers deliver data from one network to the next nearest network.

Multiplexing Protocols and Port numbers deliver data to the correct application or protocol within the remote host.

Addressing

The destination address of the IP datagram contains the unique IP address of the host where the data is to be sent. The source address contains the unique IP address of the host originating the datagram. An IP address is 32 bits long, and is usually written in the dotted decimal notation (i.e. 198.177.251.123) and contains the network address, and the host address. The length of each varies depending on a number of factors.

IP addresses are separated into classes. These classes specify how many bits within the 32 bit IP address make up the network address. Here's the rules as to how that works:

First bits*	First Byte Value	Class	# Networks	Hosts / Network
0XXX	**001 - 127**	**A**	126	16,777,214
10XX	**128 - 191**	**B**	16,382	65,534
110X	**192 - 223**	**C**	2,097,152	254
1110	**223 →**	**D**	N/A	

** X's in the bit positions indicate that these bits are used by the network address in this class.*

God what a mess… Let's look at what's important.

First Bits Forget about it. This is important because that's how it works. The only people that truly care how it works are the experts and the people that design and program routers. Don't worry about it for now. Besides, there will be more detail later than you'll care to read.

First Byte Value This is what you will see instead of bits. If you can manage to memorize this part of the table you're in good shape. Me, I need to look it up all the time.

Class This is what it's all about. Simply a reference for talking about IP networks.

 # Networks Trivia mostly. Of course this combined with Hosts / Network tell the administrator what kind class of IP network he/she needs to have to

accomplish their goal. If that goal requires a class A network, then they start planning again, as there are no class A networks left anymore.

Hosts / Network — The result of making the # Networks bigger is ending up with less hosts per network. Simple right?

Here's another way to look at how the IP address is broken down. Since the IP address is always 32 bits in length, you can see why when the size of the network address increases, the size of the host address decreases. This accounts for the number of available networks and hosts in any given network class.

Class A Addressing Format

0 1 7 8
31

0	**Network (7 bits)**	**Host Address (24 bits)**

Class B Addressing Format

0 1 2 15 16
31

1	**0**	**Network (14 bits)**	**Host Address (16 bits)**

Class C Addressing Format

0 1 2 3 23 24
31

1	**1**	**0**	**Network (21 bits)**	**Host Address (8 bits)**

Class D Addressing Format (Multicast)

0 1 2 3 4
31

1	**1**	**1**	**0**	**Multicast Address**

Don't go nuts trying to remember how many bits are used where. Try instead to remember what addresses starting with what number are what class. Huh?

Here's some examples:

Address	**Class**	**Why?**
192.104.242.150	Class **C**	**192** is in the range of 192 to 223 First three bits of address are **1-1-0**
93.134.105.222	Class **A**	**93** is less than **128** First bit of address is **0**
135.16.102.96	Class **B**	**135** is in the range of **128 to 191** First two bits of address are **1-0**

Which way is easier to remember and figure out? The first decimal number of the IP address, or it's binary equivalent? You decide.

In actual use, after figuring out what network class you are using, seeing what portion of the address is network and which is host is quite easy.

Class A using address 22.120.205.16

Net	Host		
22	120	205	16

Class B using address 131.66.23.101

Network		Host	
131	66	23	101

Class C using address 192.104.242.150

Network			Host
192	104	242	150

Class D: Don't worry about it, you probably won't see it much.

Simply put, Class A uses the first number as the network address, Class B uses the first two numbers as the network address, and Class C uses the first three numbers as the network address.

Not all of the available combinations of addresses are available for use. There are a few addresses that are reserved.

Network of all 0's	This network
Host of all 0's	Network Address
Host of all 1's (binary)	All hosts
0.0.0.0	Default route
127.X.X.X	Loopback Address
224.0.0.0 - 239.255.255.255	Class D Multicast
10.0.0.0	Private Network. Cannot be routed to the Internet.
172.16.0.0	Private Network. Cannot be routed to the Internet.
192.168.0.0	Private Network. Cannot be routed to the Internet.
Network starting with bits 1-1-1-1-0	Class E networks (future expansion)

The default route is where packets are sent when the routers don't know what to do with them. The loopback address is an address that can be used to address yourself, while using a network address other than your own. Private networks are networks that will not be routed to the Internet. These are provided for small companies or testing networks where they can be used and abused without interfering with the proper functioning of the Internet.

A note about the top three cases. When the host address is all 0's this refers to the network itself. When the host address is all 1's this is usually used as a broadcast for that network. Because of this, the first and last hosts in any IP network cannot be used. This becomes important when using subnets.
Remember also that IP addresses are assigned to network interfaces, not to hosts as the term, host address, might imply. Thus a router has more than one IP address.

Subnets

Ah yes… The part you've no doubt been waiting for. It seems that if there is one problem that everyone who learns about TCP/IP has a problem with, it's subnets. It's not as difficult as you might think.

What is a subnet? Well, say you've got your very own class C network. You've got 256 unique host addresses and one unique network address. We'll use this example:

198.177.254.0 through **198.177.254.255**

You have two divisions, the Sneetches and the Grinches, both of whom need their own networks, but you only have the one Class C network. Luckily, each division only needs about 20 host addresses, and will probably not expand in the near future. It would be extremely wasteful to get another Class C Network for so few hosts. What to do?

Wouldn't it be cool if you could just tell the Sneetches to use hosts 0-31 and the Grinches to only use hosts 32-63? Well you can. With subnets.

There are two ways to understand subnets. Basically or technically. We'll cover technically first.

Remember that an IP address is 32 bits in length, and is divided into the network address and the host address, based on the addresses class.

Class C Addressing Format

0 1 2 3 23 24 31

1	**1**	**0**	**Network (21 bits)**	**Host Address (8 bits)**

What subnetting allows us to do is move the dividing line that separates the network and host address portions of the IP address. If we move that dividing line 4 bits to the right, we then get an additional 4 bits allocated to the network address, and 4 less bits in the host address.

Class C Subneted Addressing Format

0 1 2 3 27 28 31

1	**1**	**0**	**Network (now 25 bits)**	**Host Address (now 4 bits)**

Big deal you say? Check this out:

Now that there are more than the standard 21 bits available in the network address portion, we can subdivide our normal class C network into 16 networks! The downside is that since we have taken four bits away from the host portion, there can only be 16 hosts on each of those networks.

This dividing line can be moved anywhere in the HOST portion of the address in it's original form. This means that with a Class C address you have 8 bits to play with. On a Class B network you have 16 bits to play with. You cannot move the subnet divider into the network address portion of the IP address.

The way we symbolize the subnet is through the subnet mask. This is representation of where that dividing line is. The subnet mask is usually referred to in dotted decimal notation, just like the IP address itself.

Basically we need to tell all the devices on the network where that dividing line is in the IP address. All devices on the same network must agree about the subnet mask! The easiest way for computers to understand things is binary. The easiest way for humans to understand things is usually decimal (although a good rap on the head sometimes works too).

Class C Addressing Format

0 1 2 3 23 24 31

1	1	0	Network (21 bits)	Host Address (8 bits)

In this normal Class C IP address format, the dividing line between the network and host is at the bit 23/24 boundary. If we symbolize the network portion of the address as all 1's and the host portion as all 0's we could write it like this:

11111111111111111111111100000000

If we break that into octets (bytes) it looks like this:

11111111 11111111 11111111 00000000

If we then convert those 1's and 0's to decimal we get:

255.255.255.0 Which just happens to be the subnet mask for a class C network!

Thus we say that a the class C network **192.104.242.0** has a subnet mask of **255.255.255.0**! In fact, if not subnetted, all class C networks have a subnet mask of 255.255.255.0.

Now if we move the dividing line as we did in the example above we get:

Class C SUBNETed Addressing Format

0 1 2 3 27 28 31

1	1	0	Network (now 25 bits)	Host Address (now 4 bits)

Let's break it down. 28 bits of 1's and 4 bits of 0's looks like this:

11111111111111111111111111110000

which is:

11111111 11111111 11111111 11110000

which becomes in decimal:

255.255.255.240 which is the subnet mask for the above IP scheme.

Remember that a host of all 0's and a host of all 1's signifies the network address and the broadcast address of the network. When a network is subnetted, you now have X number of networks instead of the original one. Each one of those networks has a network address and a broadcast address, which cannot be used for hosts. The network address is the first host address in that network. Let's look at an example where a class C network has been divided into 8 networks of 32 addresses each.

198.177.254.0 has been subnetted with a mask of **255.255.255.224** making the following list of networks:

#	Network Address	Broadcast Address
1	198.177.254.0	198.177.254.31
2	198.177.254.32	198.177.254.63
3	198.177.254.64	198.177.254.95
4	198.177.254.96	198.177.254.127
5	198.177.254.128	198.177.254.159
6	198.177.254.160	198.177.254.191
7	198.177.254.192	198.177.254.223
8	198.177.254.224	198.177.254.255

So looking at the above example, we can see that the class C network which started with 256 hosts (0-255) is now 8 networks with 32 hosts on each network.

Well, that's how it works. And that's pretty much why nobody ever remembers how it works! Computers think that way, not people. Here's some reference tables that will help out if you need to make or figure out subnet masks in the future:

On a class C network you can only subnet so far. Here's the breakdown

Subnet Mask	# Networks	Hosts / Network
255.255.255.0	**1**	**256**
255.255.255.128	**2**	**128**
255.255.255.192	**4**	**64**
255.255.255.224	**8**	**32**
255.255.255.240	**16**	**16**
255.255.255.248	**32**	**8**
255.255.255.252	**64**	**4**
255.255.255.254	**128**	**2**

Of course in a perfect world you could use all of those addresses. But in the real world there are rules, and sadly, they must be adhered to. Remember you cannot use a host address of all 0's or all 1's. Now this becomes an issue because if you use a subnet mask of 255.255.255.254, then there are only 4 hosts per network, and you cannot use the first or last ones! So out of 4 you lose 2. On every subnet you made no less! 50% of all your available addresses just went away. The only way to circumvent this little gem of a conundrum is to not use super small subnets.

Unfortunately there is a rule that also says you are not supposed to use the first and last subnets in a subneted network! Now that really sucks because is you subnet your class C into 4 networks, you lose half of your available addresses again! As they say up north, “You can’t win for losing!”. Some routers will allow you to use the first and last subnets but some will not. Usually the manual will tell you, but it’s a good idea to stick to the rules whenever possible.

Routing

The Internet can be viewed as a collection of networks, connected by routers. Each host on a network can communicate directly with any other host on the same network. If communication with a host on a different network is desired, then the packets are sent to the appropriate router, and the router sends the packets in the right direction.

An important thing to remember is that routers forward packets based on the destination network address, and not on the destinations host or physical address. remember that TCP/IP addressing is in the Internet layer and not the Network layer, which means a router must decode a packet to determine if the packet needs to be forwarded, and if so, to where.

Direct Routing

When two hosts are on the same network, they can communicate directly with each other, without the aid of a router. This is called direct routing.

In the case of direct routing, if there are no routers on the network (a stand-alone network), then the use of IP is considered to be unnecessary overhead. Since their is no need for internetworking, much of the overhead created by IP for that purpose is wasted. Therefore if there is no routing to be done and all communication is done via direct routing. IP should not be used.

Indirect Routing

Indirect routing occurs when two hosts communicate on different networks. This is accomplished with the aid of a router that connects those networks. This is much more involved than direct routing for two reasons:

- The source host must be able to identify a router in which to address the datagrams.
- The router must have obtained the proper routing information to properly forward the datagrams towards the destination network.

Indirect routing is performed by the IP protocol and is completely transparent to TCP or UDP.

To determine if a datagram needs to be indirectly routed, the source host examines the destination network address of the datagram. If this address matches the host's own network address, then the datagram is sent directly to the destination host. If the two network addresses differ, then the datagram is sent to the appropriate router for forwarding. Typically a default gateway is configured by the network administrator on each host. This default gateway is the router where the host may send packets that must be forwarded to other networks.

Routing Tables

Don't think that only routers route. A host actually decides where a packet should be sent first. Often that decision is to send the packet to a router, but technically the host has done routing by deciding to send the packet there in the first place. Here's how it works:

Every host has a local table of what it knows about the local network, as well as other networks it can reach via routers. When a packet is sent, the host first checks the network address of the destination host against its own network address. If they match, then the packet is sent to that host. If the network addresses differ, then the host examines its own routing table and looks for the destination address there. If it is found, the host will then send the packet to the router corresponding to the destination network address. Confused? Don't be.

This is a sample routing table pulled of f of a Linux machine on a network with two routers. The command used to get this information is ***netstat -rn***.

	Destination	Gateway	Flags	Ref	Iface
	206.160.71.209	192.104.242.37	UGHDM	1513	eth0
	131.99.66.101	192.104.242.37	UGHDM	453	eth0
	131.99.221.91	192.104.242.37	UGHDM	3072	eth0
1)	204.214.144.2	192.104.242.1	UGHDM	5	eth0
	131.99.222.134	192.104.242.37	UGHDM	1044	eth0
	131.99.221.255	192.104.242.37	UGHDM	102	eth0
	129.52.1.81	192.104.242.37	UGHDM	940	eth0
2)	129.65.1.101	192.104.242.37	UGHDM	102	eth0
	131.99.99.101	192.104.242.37	UGHDM	6	eth0
	129.57.1.78	192.104.242.37	UGHDM	112	eth0
	192.104.242.0	0.0.0.0	U	1528429	eth0
3)	127.0.0.0	0.0.0.0	U	778	lo
4)	0.0.0.0	192.104.242.1	UG	771724	eth0

Here's what the columns mean:

Destination The destination network or host

Gateway The gateway (router) used to get to that destination network or host

Flags
- **U** Route is useable (Up and running)
- **G** Destination is a gateway
- **H** Destination is a host entry
- **D** Route created dynamically via ICMP redirect
- **M** Route modified dynamically via ICMP redirect

Ref Reference count for this route

Iface Name of the network interface this route belongs to

Looking at the example routing table, we can see some interesting examples. Lets look a little deeper at the highlighted examples above.

204.214.144.2	192.104.242.1	UGHDM	5	eth0

(1) This example shows that in order for this host to send a packet to host **204.214.144.2**, the packet must be sent to the router at **192.104.242.1**. The flags are explained above.

129.65.1.101	192.104.242.37	UGHDM	102	eth0

(2) Same deal here, only in order to access this destination host, the packet must be sent to a different router (**192.104.242.37**).

127.0.0.0	0.0.0.0	U	778	lo

(3) This is a special route. The destination network 127.0.0.0 is the local network. Note that the Interface for this route is ***lo***. This means that the NIC on the system is not even used, rather it is all handled by the IP driver.

0.0.0.0	192.104.242.1	UG	771724	eth0

(4) Another special route. This one indicates that the default gateway is **192.104.242.1**. When a destination host is searched for and not found in the table, this is where the packet is sent. Hopefully the router set up as the default gateway knows where to send it. If it doesn't, then that router has a default gateway too! And so on, and so on… Get the idea?

In a nutshell here's how a packet gets routed:

1) Source host decides if the destination is on its own network

If it is, it is sent directly to the destination. Otherwise…

2) The host searches its local routing table. If the destination network is found, the packet is forwarded to the appropriate gateway. Otherwise…

3) The host forwards the packet to the default gateway.

The default gateway then decides the best route for the packet to take according based on that gateways own routing table.

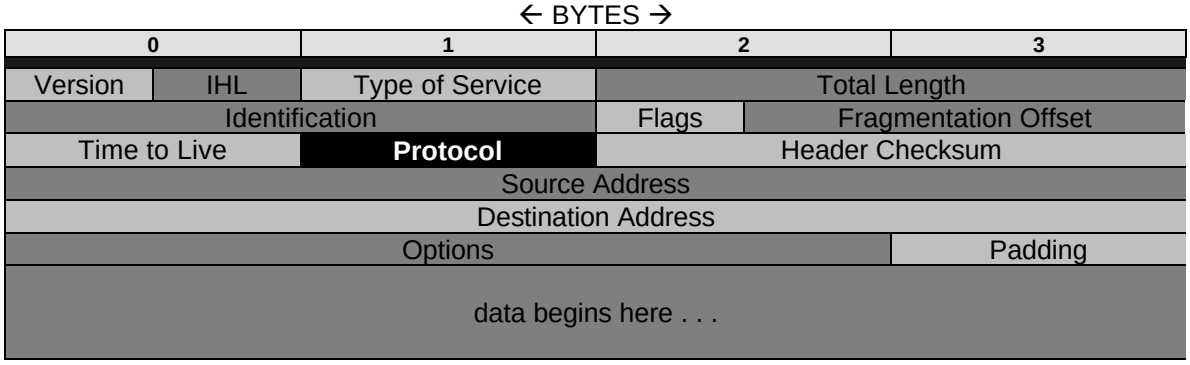
Multiplexing

Once the packet gets to the destination host. What does that host do with it? Well, simply put the packet is passed up the protocol stack until it gets where it belongs. Lets say for now that we are dealing with packets destined for the application layer (TELNET or FTP perhaps). Once the packet gets to that layer, how does it know what application it belongs to? On a multitasking system there could be many applications running at the same time. After all the extensive travel the packet has taken, possibly across the world, wouldn't it suck if that packet, having finally arrived at the destination host, had no idea what application it was for?

Protocol Numbers

When the packet gets to the Internet Layer, the IP protocol header contains a field called Protocol. This field contains a code which indicates what protocol IP should hand this packet off to.

IP



On a Unix machine, these protocols are defined in a file called */etc/protocols*. This is a sample of that file:

% cat /etc/protocols

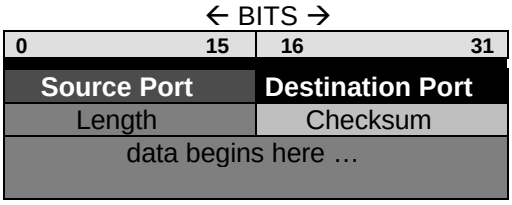
```
# /etc/protocols
#
# Comments are good. I love comments!
#
ip      0       IP      # internet protocol, pseudo protocol number
icmp    1       ICMP    # internet control message protocol
igmp    2       IGMP    # internet group multicast protocol
ggp     3       GGP     # gateway-gateway protocol
tcp     6       TCP     # transmission control protocol
pup     12      PUP     # PARC universal packet protocol
udp     17      UDP     # user datagram protocol
idp     22      IDP     # What's This?
raw     255     RAW     # RAW IP interface
```

Looking at this file, we can see that TCP has a protocol value of 6. This means that when a datagram is received by IP, and the protocol field contains a 6, then IP will hand this packet off to TCP. If the value contained in protocol was a 17, then UDP would get the packet and so on.

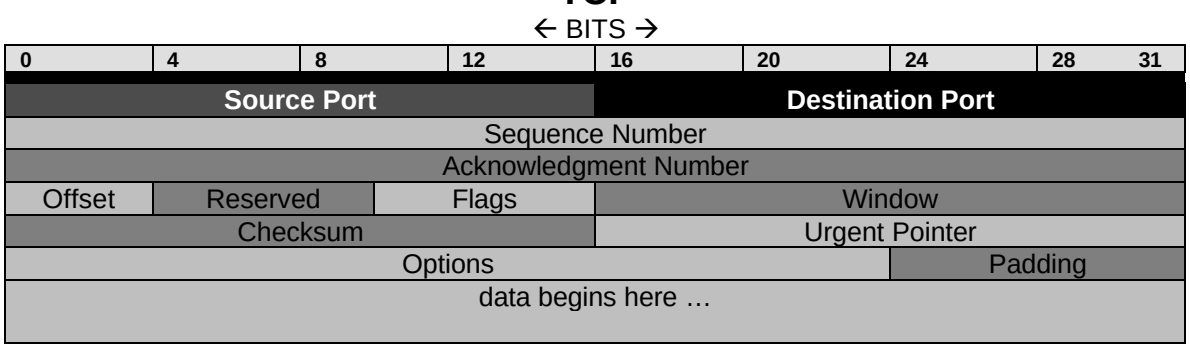
Port Numbers

Remember that TCP or UDP is not the final step up the ladder in the TCP/IP suite. These protocols must then hand off to the proper application residing above them. This is done by way of the *port* field. In fact TCP and UDP both have fields called *source port* and *destination port*.

UDP



TCP



When an application creates a packet, it puts its own port identifier into the source port field. It puts the port identifier for the target application in the destination port field. While the application may be the same on both ends, the port might not be. Many applications provide different port numbers for the client side and the server side (See Socket Numbers).

Note that the port numbers are not unique among the protocols. In other words FTP could use TCP *or* UDP depending on the situation. It is the combination of protocol number *and* port number that ensures proper delivery of the packet to the right application.

In UNIX, these port numbers are maintained in the file */etc/services*. The following page shows an example */etc/services* file from a Linux machine. This list has been abbreviated slightly in the interest of space.

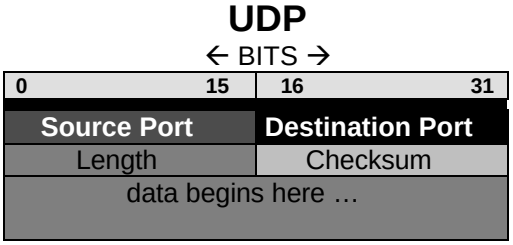
% cat /etc/services

```
#
# Comments are cool!!
#
# “Well Known” Services
#
echo            7/tcp
echo            7/udp
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp      mail
time            37/tcp      timserver
time            37/udp      timserver
name            42/udp      nameserver
whois           43/tcp      nicname             # usually to sri-nic
domain          53/tcp
domain          53/udp
bootps          67/udp      #  bootp server
bootpc          68/udp      #  bootp client
tftp            69/udp
gopher          70/tcp      #  gopher server
finger          79/tcp
pop-3           110/tcp     #  PostOffice V.3
pop             110/tcp     #  PostOffice V.3
nntp            119/tcp     usenet              # Network News Transfer
ntp             123/tcp     #  Network Time Protocol
ntp             123/udp     #  Network Time Protocol
snmp            161/udp
snmp-trap       162/udp
#
# “UNIX-Specific” Services
#
exec            512/tcp     #  BSD rexecd(8)
login           513/tcp     #  BSD rlogind(8)
who             513/udp     whod                # BSD rwhod(8)
shell           514/tcp     cmd                 # BSD rshd(8)
syslog          514/udp     #  BSD syslogd(8)
printer         515/tcp     spooler             # BSD lpd(8)
route           520/udp     router routed       # 521/udp too
timed           525/udp     timeserver
netnews         532/tcp     readnews
uucp            540/tcp     uucpd               # BSD uucpd(8) UUCP service
mount           635/udp     #  NFS Mount Service
pcnfs           640/udp     #  PC-NFS DOS Authentication
#
# “Other” services
#
nfs             2049/udp    #  NFS Fle Service
irc             6667/tcp    #  Internet Relay Chat
```

Sockets

All of these ports and protocols are great, but what happens when two users try to TELNET into a server at the same time?

What happens is this: The server assigns all users the destination port of 23 (TELNET). Each user is then assigned a unique *dynamically assigned* port number. Whoa! What the hell does that mean you ask? Well if you look at the datagram layout;



you can see that the port fields are each 16 bits long. This allows for a number from 0 to FFFF (65,535). As you can see from the /etc/services file, they are not even close to all being accounted for.

In the TELNET example sited, the server would assign the first user a dynamic port of say 3044 as a source port and the destination port of 23 (TELNET). The second user would be assigned a different dynamic source port of say 3052 but the same destination port of 23. In this way TELNET can serve multiple users, and multiple users can look for TELNET at the same port number.

The combination of protocol number and port number is called a socket.

Summary

Lets look at the hierarchy of address, ports and protocols. Here the host 192.104.242.150 is communicating with host 192.104.242.2 using TELNET.

Packets originating on host 150 would have the following values:

	Source	Destination
Port	3044 (Dynamic)	23 (TELNET)
Protocol	6	6
IP Address	192.104.242.150	192.104.242.2

Packets originating on host 2 and bound for host 150 would have the following values

	Source	Destination
Port	23 (TELNET)	3044 (Dynamic)
Protocol	6	6
IP Address	192.104.242.2	192.104.242.150

If the same host initiated another TELNET session concurrently, the only thing that would be different is the dynamic port number.

The combination of all these numbers ensures that the proper datagrams get to and from the proper hosts and then to their proper applications. Pretty cool huh?

Appendices

" The daintiest last, to make the end most sweet."

William Shakespeare
King Richard II
Act I, Scene III

Bibliography

(In strictly non bibliographical format)

TCP/IP Network Administration

Craig Hunt
O'Reilly & Associates, Inc.
103 Morris Street, Suite A
Sebastopol, CA 95472
http://www.ora.com

IP Routing Student Guide

Chuck Semeria
3Com
5400 Bayfront Plaza
Santa Clara, CA 95052
http://www.3com.com

Computer Communications: A Business Perspective

Keith Bearpark and Peter Beevor
McGraw-Hill Book Company
Shoppenhangers Road
Maidenhead
Berkshire England
http://www.books.mcgraw-hill.com

Acknowledgments

Proof Readers Lauren Donahue, Dave Jump

Technical Proof Reader Any volunteers?

Equipment Used This document was created entirely in Microsoft Word for Windows-95 version 7. The hardware involved includes A Compaq ProLinea 575, and a homemade Pentium Overdrive 83 MHz machine. This document was printed on either a Hewlett Packard 4M or a Hewlett Packard IVsi at 600 dpi.

Main Reference The book I refer to the most when dealing with TCP/IP is O'Reilly's TCP/IP Network Administration. The structure of this document is based on the first portion of that book. I highly recommend it to anyone interested in more information on the subject. In fact I recommend all of the O'Reilly books. You can check out their catalog of books at http://www.ora.com.

Comments Please let me know your opinions! What other documents and/or classes would you like to see? I would love to hear from you. I can be reached at the following places:

lordgad@planet.net

This document may also be available in HTML format on the world wide web. Take a look at:

http://www.gad.net
or
http://www.planet.net/plordgad

Other Documents by Gary A. Donahue

Networking	**Astronomy**
TCP/IP	The Sun and Solar Eclipses
DNS	Meteors and Meteor Showers
	The Jupiter Comet Crash of 1994

All documents are in Microsoft Word Format.

Glossary

TCP/IP	Transport Control Protocol Internet Protocol. The common name and acronym for a suite of protocols designed for internetworking.
IP	Intenet Protocol
TCP	
UDP	
TELNET	
FTP	
SNMP	
SMTP	
Source Quench	

Index

F

G

H

I

L

M

N

O

P

R

S

T

U

W

X

Y