NOLUG -- TCP/IP and Linux course

Scott Harney

Barry McCormick

NOLUG -- TCP/IP and Linux course

by Scott Harney and Barry McCormick

This book is a detailed expansion of three sessions we taught at the New Orleans Linux User’s Group concerning implementing TCP/IP and Linux. It is aimed primarily at the novice but also will help the more experts amongst the group brush up on their knowledge as well.

Users often "fill in the blanks" in their networking setups without having any real understanding of what the information means or how it works. Having a framework of knowledge allows the user to adapt to new situations and environments and create solutions to match his/her needs. We will detail both the theory and show actual configuration details for current Linux distributions for our sample networks. We actually set up those sample networks in the meetings. Our ultimate intention, however, is to demonstrate that by understanding the workings of TCP/IP, the user can apply that knowledge to set up TCP/IP networking in any environment. The only way to find an answer to a question is to first understand and define the question (or questions) being asked.

You may also get a pdf suitable for printing here (tcp-ip-s1.pdf). You’ll need a pdf viewer of course. Or if you have the cygnus docbook tools or sgmltools, you can grab the original sgml source here (tcp-ip-s1.sgml) and format it yourself.

Table of Contents

1. Introduction to TCP/IP 9
1.1. History of TCP/IP and the Internet 9
1.2. TCP/IP - the OSI Model 10
1.3. TCP/IP - The Math Behind It All 13
1.3.1. The mathematical AND operator..15
1.4. Subnetting and subnet masks...16
1.5. TCP/IP Address Classes & Reserved IP Space 16
1.6. CIDR notation. Breaking out of Classful subnetting......................................18
1.7. Packets & Ports (You want me to put what where?).......................................22
1.8. Network Troubleshooting 27
2. Building Your Home Network 39
2.1. Acquiring Equipment(dumpster diving at it's best)..39
2.2. Network Cards 40
2.3. Hubs & Switches...41
2.4. Cables & Connectors 42
2.5. DHCP - Assigning IP addresses 43
2.6. NAT/IP Masquerade...45
2.7. Cable/DSL..47
2.8. Building a home Linux router (freesco)..48
2.9. Dialup to the Internet (you poor soul)..52

List of Tables

1-1. Layer Model 10
1-2. Layer Model 11
1-3. Binary to decimal translations 14
1-4. IP address classes....................17
1-5. Subnetting examples table 21
1-6. Common ports 25

Chapter 1. Introduction to TCP/IP

1.1. History of TCP/IP and the Internet

TCP/IP has become the de-facto protocol for computers to talk to each other over networks and the Internet. How did this come to be the most used protocol ever? First let’s look at the history of the Internet and tcp/ip. They are intermingled together such that it is impossible to talk about one without the other.

First we go back to the late 60’s and early 70’s. There were NO desktop computers as we think of them today. Most of the computers were mainframes at either very large companies, or the Government facilities. Most of them were either military or at universities. There were several different brands/types. You had UniVac’s, IBMs, Burroughs, etc. There were NO standards for these computers, so them swapping data was a very tough proposition at best. It involved saving data to a huge tape spool in a decided upon format, then shipping this data to the other computer, and uploading it. Very expensive, time consuming, and not very efficient.

The government wanted a way to share data between these very different systems in which they had invested a lot of money to develop. That is the reason they funded the development of a networking protocol that has become tcp/ip. This work was done by a government agency named DARPA (Defense Advanced Research Projects). This work not only developed TCP/IP, but started the Internet.

The Internet was in the beginning, a WAN[1] network used by universities and military institutions. There were no private connections to the Internet, and the idea behind the Internet was to develop a network that even if part of the network was destroyed by an enemy, there was redundant paths that would continue to send the information. So, tcp/ip and the Internet were actually products of the cold war.

This is why no one owns tcp/ip, and it is a free protocol to use and distribute, unlike many of the other proprietary protocols that have come and gone over the years (Appletalk, Novell IPX/SPX, Arpnet, IBM token ring, etc). One of the specifications that was listed for the development of tcp/ip was that the protocol had to be hardware independent. Remember, the entire idea was to connect different hardware platforms together. This is why tcp/ip can be used by every operating system and every hardware platform in existence today.

1.2. TCP/IP - the OSI Model

What the heck am I talking about when I say THE OSI model? OSI Reference model is the model that was developed to show the different independent sections of a protocol. The tcp/ip protocol pre-dates the OSI model, but it lends itself very well to using this model to describe it, because it is hardware independent. TCP/IP consists of only 4 layers, where the OSI model consists of 7 layers. But, each of the layers of tcp/ip corresponds to one or more layers of the OSI model. Below is a diagram of the layers of the OSI model and the corresponding tcp/ip layers.

Table 1-1. Layer Model

OSI REFERENCE MODEL	REAL WORLD EQUIVALENT TASKS	TCP/IP PROTOCOL
Application	Program to program transfer of information	Application
Presentation	Text formatting and display code conversion	
Session	Establishing, maintaining, and coordinating communication	
Transport	Accurate delivery, service quality	Transport
Network	Transport routes, message handling and transfer	Internet
Data Link	Coding, addressing, and transmitting information	Network Interface

OSI REFERENCE MODEL	REAL WORLD EQUIVALENT TASKS	TCP/IP PROTOCOL
Physical	The hardware connections (the NIC and the cables)	

**** All People Seem To Need Data Processing ****

**** Please Do Not Throw Sausage Pizza Away ****

These two somewhat silly sayings will help you remember the layers of the OSI model and what the first letters and order is. If you can get the letters in the right order, you can figure out the names of the layers. This is a common question on certification tests for all networking type subjects. Know it, live it, love it. Or not, just remember it. You will see it again!

Why is this silly diagram so important? Because tcp/ip is NOT one single protocol, but it is actually a suite of protocols, and each one of the parts that make up tcp/ip function at a specific layer of the model. This is important to understand what is actually happening when you are setting up a network using tcp/ip, or sending files to the Internet. Without a good understanding of the layers and what they do, it is more difficult to understand where to look when things go wrong with your network. Below is a table that explains in detail what each layer does for the network.

Table 1-2. Layer Model

OSI REFERENCE MODEL	Layer Definitions
Application - Layer 7	This is the layer where you perform your tasks, writing the email message, running your browser, opening an ftp session, receive and read your email, etc.

OSI REFERENCE MODEL	**Layer Definitions**
Presentation - Layer 6	The presentation layer works with the file system and the operating system. This is the layer that files get transferred from one format to another. This allows computers of different platforms to swap data. Without this layer, you could only network same platform computers. And if you remember, that would defeat one of the main purposes of developing the tcp/ip protocol suite.
Session - Layer 5	The other protocols of the tcp/ip suite work at layer 5 and above. This layer establishes and coordinates a session, which is simply a connection between two computers. You cannot transfer data between two computers without a session being established.
Transport - Layer 4	This is the layer of the model that the tcp protocol works. While layer 3 sends your data to it's destination, Layer 4 is the one that maintains the order of the packets. This layer makes sure the packets are received without error and that the packets are reassembled in the proper order. Without this layer your network would not be trustworthy for valuable data.

OSI REFERENCE MODEL	**Layer Definitions**
Network - Layer 3	This is the layer that the IP portion of the tcp/ip protocol fits into the model. This layer gets the packets from the data link layer and sends them to the correct network address. If there are multiple paths, this is the layer that determines what is the best possible route. Without this layer your data would never make it to the right destination.
Data Link - Layer 2	This is the layer that splits your data into packets readying them to be sent across the wires to their destination. The data link works to assure accurate data transmission. This is also the layer where the token or ethernet packets are handled.
Physical - Layer 1	The physical layer is pure hardware, i.e. Cables, network cards, hubs, etc. All the parts that the electrical signals are transported between the machines.

1.3. TCP/IP - The Math Behind It All

Everyone has heard the expressions tcp/ip address, or network address, and subnet mask used in reference to configuring your computer to either get on the Internet, or connect to your network at work. What does this mean? I knew what numbers to put into a machine to make it work long before I knew why those numbers worked. To ever really understand sending data and packets on the Internet, and how to properly set up your own network, it is important to understand not only what the valid numbers are, but how they are calculated, and physically what they mean. To understand what I mean when I write 216.227.108.198/24, or use a command in Linux like:

ifconfig eth0 192.168.7.7 netmask 255.255.255.0

Where in the world do those numbers come from, and what do they mean?

First, let us talk about binary math. Everyone has heard of it, but what does it mean and do? Computers work off of electricity (no joke!), and in the world of electrons and electricity there are only two states, charged and uncharged. This corresponds to either a zero (0), uncharged, or a one (1), charged. So computer chips naturally think and work in binary terms. This is why all the numerical representations of address must be expressed in binary form. Below is a table that shows some simple counting and how the numbers correspond to their binary equivalent.

Table 1-3. Binary to decimal translations

00000001	= 1	10000000	= 128
00000010	= 2	11000000	= 192
00000011	= 3	11100000	= 224
00000100	= 4	11110000	= 240
00000101	= 5	11111000	= 248
00000110	= 6	11111100	= 252
00000111	= 7	11111110	= 254
00001000	= 8	11111111	= 255
00001001	= 9	01111111	= 127
00001010	= 10		

Now why did I use 8 numbers to represent each of these numbers? because IP addresses are represented by 4 octets(ie. 8 bits or 1 byte), which corresponds to 8 binary digits in each octet for a total of 32 bits. So an ip address is considered a 32 bit number. Rather than us to always write out, or type in a bunch of binary numbers into the computer to set the addresses and subnet masks, we use the decimal equivalent. So if you are trying to figure out what the decimal equivalent of a binary number, the equation to use to accomplish this is:

```
                          (n-1)
        summation of {m*2      }

So a number of 11001001 in bi-
nary equals    2^7 + 2^6 + 2^3 + 2^0 =
                                        =  128 + 64 + 8 + 1
                                        =  201
```

This may seem really strange, but if you practice it a little, it becomes quite easy to calculate.

1.3.1. The mathematical AND operator

Now what am I talking about when I say AND mathematically? Everyone knows what add means, and what subtract means, but what is a mathematical and and a mathematical or mean? AND is the command that mathematically TCP/IP uses to mask out bits of an address. Look at the following table to see what anding two numbers in binary means.

```
1 AND 1 = 1
1 AND 0 = 0
```

so 1 AND any number equals that number.

```
0 AND 1 = 0
0 AND 0 = 0
```

so 0 and any number equals 0. Therefore any number you put a subnet mask of 255 against (255=11111111) you get that number. Any type you use a 0 as a subnet mask, you get a zero. For instance:

```
ip address of   192.168.13.0
subnet mask of  255.255.0.0
```

What is the network address? We AND these two numbers together and get 192.168.0.0 is the network's address. Write these numbers down in binary and look at them and you will see how the ones and zeros mask portions of the address.

1.4. Subnetting and subnet masks

Now that we have spent a lot of time working out way through the math behind a subnet mask and how to calculate ip addresses, why do we need a subnet mask? What is it's purpose? A subnet mask can be used to break a network into smaller sections, or subnets. Why would you want to do this? Since there are limited number of ip addresses, you're Internet provider will likely provide you a single address range according to your estimated needs. So if your network needs to be broken up into smaller subnets to either reduce traffic, or for security reasons, then you must break it into subnets.

What are 4 of the major reasons for subnetting or segmenting your network?

1. To divide a large network into smaller segments to reduce traffic and speed up the sections of your network.
2. To connect networks across geographical areas.
3. To connect different topologies such as Ethernet, Token Ring, and FDDI together via routers.
4. To avoid physical limitations such as maximum cable lengths or exceeding the maximum number of computers on a segment.

In the next two sections you will see examples of the different sized subnets you may use.

1.5. TCP/IP Address Classes & Reserved IP Space

Since IP addresses consist of 32 bit numbers, that means there are 2^{32} available IP addresses on the Internet. 2^{32} = 4,294,967,296 addresses between 0.0.0.0 and 255.255.255.255. The powers that allocate IP addresses arbitrarily broke up IP space into the network classes A,B, & C. They allocate these subnets to different organizations according to an organization’s estimated needs. Those organizations then dole out portions of their IP space to subscribers or even other smaller Internet Service Providers.

Here are the traditional classful network sizes and boundaries[2]:

Table 1-4. IP address classes

Class	address range	network/host parts	# of networks	# of hosts
A	1.0.0.0-126.255.255.255	n.h.h.h	126	16,777,214
B	128.0.0.0-191.255.255.255	n.n.h.h	16,384	65,534
C	192.0.0.0-223.255.255.255	n.n.n.h	2,097,152	254

You have undoubtedly noticed a couple of networks missing from this. The class A sized network 127.0.0.0 is reserved for localhost and loopback testing. Addresses 224.0.0.0-up are class D multicast and class E experimental addresses. We won’t be discussing multicast here, of course.

There are also specific networks described as *private* IP space. These networks are reserved from the public IP classes described above. Private IP space can be used by networks not connected to the Internet directly. There is no fear over IP address overlap because these private blocks are not intended to be routed across the internet[3].

- 10.0.0.0/8 is a Class A network ID allowing the range of usable host IP addresses from 10.0.0.1 to 10.255.255.254
- 172.16.0.0/12 is a network ID that consists of 16 Class B network blocks allowing the range of usable host IP addresses from 172.16.0.1 to 172.31.255.254
- 192.168.0.0/16 is a network ID that consists of 256 class C network blocks allowing the range of usable host IP addresses from 192.168.0.1 to 192.168.255.254

You've probably already noticed our use of addresses from these private IP blocks in our examples. These are the addresses you'll be using in your private home networks while you probably only get 1 public IP address from your Internet Service Provider.

You probably also noticed the odd notation 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. What is /8, /12, and /16? The next section explains that we can subnet in many other ways besides breaking on traditional 8-bit boundaries....

1.6. CIDR notation. Breaking out of Classful subnetting

You may have already noticed that IP address classes are pretty limited in terms of the size of networks you can allocate. When TCP/IP was created, it was simple for humans to understand the address classes because the broke cleanly on octet boundries (right at the dots in an IP address). Furthermore, no one was particularly concerned about the potential inefficiency and wastage of IPs when the supply of IPs seemed inexhaustable.

They're not inexhaustible, of course, and something had to be done. The breaking of IP space on 8-bit boundaries was a convenience mechanism but it's not necessary to break your subnet and host parts on 8-bit boundaries. CIDR, Classless Internet Domain Routing, was invented to solve the related problems of IP address shortage.[4]

A traditional Class C network division looks like this using 24 bits for the network part and the remaining 8 bits to represent the hosts:

```
   255     255      255       0
11111111.11111111.11111111.00000000
\-------network----------/ \-host-/
```

It’s very easy to see how the network breaks on the third octet.

But it’s not necessary to do that. Here, we subdivide our traditional class C expanding the netmask by one bit beyond the octet boundary. So we’re using 25 bits for network part now.

```
  255      255      255     128
11111111.11111111.11111111.10000000
\-------network----------/ \-host-/
```

So this changes our netmasks and we get masks that look like 255.255.255.128 for example. Instead of using 24 (8*3) bits for our network part, we’re using 25 bits -- hence the change in the last digit of the mask. This network essentially divides a class C network of 256 total addresses into two subnetworks of 128 addresses each. Of course you then lose 4 addresses, 2 in each subnet, for the network and broadcast addresses. your subnets then look like this:

```
   192.168.0.0 - 192.168.0.127
      192.168.0.0 is network address, 192.168.0.127 is broad-
cast address
   192.168.0.128 - 192.168.0.255
      192.168.0.128 is network address, 192.168.0.255 is broad-
cast address
```

CIDR also gives us a new form of notation referred to alternately as "cidr notation","prefix notation", or "slash notation". Using this method, we note the network prefix part of an address by adding a "/" followed by the number of bits used for the network part. Thus a Class C address can be described in prefix notation as 192.168.0.1/24 . This is a convenient shorthand for describing IPs within their network contexts. Applications that accept this notation can easily calculate the network information from the prefix address. In our sample above we have 192.168.0.0/25 and 192.168.0.128/25 for our networks. You will sometimes here network admins referring to networks as a "slash 24" instead of Class C and the like.

We don't have to divide up the common /24 (aka Class C). We can divide to fit our needs. For example a /23 gives us a network of 512 addresses. It can be described as a further subnet of the traditional Class B (/16) network or as a *supernet* combining 2 traditional Class C sized networks.[5]

CIDR can be a bit hard on humans to calculate. With traditional classful networks, just looking at the .'s helped to show you where the networks ended and the hosts began. There are numerous programs for Linux and other OS's that do subnet calculation for you. There are two commandline tools for *nix, cidr (http://home.netcom.com/~naym/cidr.html) and ipsc (http://ipsc.sourceforge.net/software.html). There are also graphical tools such as gipsc (http://ipsc.sourceforge.net/software.html) which is a GNOME IP subnet calculator. I even have an IP calculator on my palm pilot.

Most linux literature I have read on TCP/IP avoids classless networking, but it has actually been in common use on the internet for quite some time. You may have already been familiar with traditional Classful networks and been curious about those "strange" looking subnet masks. Since it's become so common in the real world, we figured you ought to know about it.

You should also know about it because it's the source of some common mistakes. Network providers are now allocation smaller subnets out of what was traditionally considered class A space. For example, most of the Cable providers have allocations from 24.0.0.0/8 for their modem pools. This creates a problem sometimes when you don't specify a mask with certain applications; may applications will assume the traditional classful subnet mask if none is given. If you were to set up your network interface using **ifconfig (1)**[6]on a Linux box like: **ifconfig eth0 24.1.2.3** because your cable provider gave you that as your static address, ifconfig would automatically assume a mask of 255.0.0.0. This would likely cause you to not be able to route or at least make millions of hosts on the Internet in the 24.0.0.0/8 network invisible to your machine. Thus it's critical to specify the appropriate netmask and not allow software to apply default Classes.

Just to give you some more examples to pore over, below are examples of subnets and how they break ip address ranges into sections.

Table 1-5. Subnetting examples table

Number of subnets required for a Class A Network	**Number of hosts**	**Number of bits used in the subnet mask**	**Subnet Mask**
2	4194302	2	255.192.0.0
6	2097150	3	255.224.0.0
14	1048574	4	255.240.0.0
30	524286	5	255.248.0.0
62	262142	6	255.252.0.0
126	131070	7	255.254.0.0
254	65534	8	255.255.0.0
Number of subnets required for a Class B Network	**Number of hosts**	**Number of bits used in the subnet mask**	**Subnet Mask**
2	16382	2	255.255.192.0
6	8190	3	255.255.224.0
14	4094	4	255.255.240.0
30	2046	5	255.255.248.0

Number of subnets required for a Class B Network	**Number of hosts**	**Number of bits used in the subnet mask**	**Subnet Mask**
62	1022	6	255.255.252.0
126	510	7	255.255.254.0
254	254	8	255.255.255.0

So as you can see, using a non-standard subnet mask, you can break the ip addresses into subnets to fit more specific needs.

We could spend many many pages talking about subnet masking, and spend many many hours delving into the deep dark recesses of this subject. But, this is intended as a beginners course to help you understand the basics of subnets, not turn you into a certified network engineer. Get yourself a good book on tcp/ip, and believe it or not, there are a couple of the Microsoft course books for mcse tcp/ip test that are pretty good. The TCP/IP for dummies book, both the msce dummies book and the standard one are both very good for someone to start with.

1.7. Packets & Ports (You want me to put what where?)

Okay. So now you've got your network size and you can route data between hosts via IP addresses. Now we need to move up the layer model from network host-to-host transport into the Session and Application layers. Now that we're moving data around, how does the computer know what to do with that data?

You've probably heard the term "packet" before and maybe you know that TCP/IP moves data around in the form of packets. So what is a packet and how does data get to be a packet?

TCP/IP is said to be a packet-switched networking method. These easiest way to understand this is to contrast it against the more traditional circuit-switched telephone network. In the phone network you pick up the phone and dial a number. when the phone on the other end is taken off-hook that completes a circuit -- a single point-to-point connection is established. If there is a break anywhere in that circuit, the call is lost. (I know this is an oversimplification).

Packet-switched networks, however, first break the data into small chunks called packets. A header is attached to the packet containing routing information and the individual packets are sent out onto the network. By breaking up the data into packets, changes in routing can occur dynamically. Packets arrive on the other end of a communication and the data is reassembled by the receiving host computer. Packets lost during transmission will be retransmitted by the sender, possibly taking a different route to the receiver.

TCP, Transmission Control Protocol, is responsible for keeping track of packet sequences in both the sending and receiving hosts. In other words, TCP maintains a connection session for duration of a network transmission. This can consist of receiving an email, receiving a web page, etc. TCP is also responsible for handing those packets off to the appropriate application.

All the information regarding packet routing and session information is contained in the packet's header. Here's the basic construction of a TCP/IP Packet:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |      TOS      |         Total Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|    Fragment Offset      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      TTL      |   Protocol    |       Header Checksum         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Source Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Destination Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Options....                               (Padding) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Data...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

As you can see from the diagram above, the 32-bit header contains all the information need to get the packet from it's source to it's destination as well as details concerning what to do with the packet once it gets there. The numbers at the top are the actual bits of the header so each row makes up a single 32 bit "word". The contents of each portion of the header are as follows: [7]

- Version: Specifies the IP version of the packet. The current version of IP is version 4.
- Internet Header Length (IHL): Indicates the length of the datagram header in 32 bit (4 octet) words. A minimum-length header is 20 octets.
- Type of Service (TOS): Allows an originating host to request different classes of service for packets it transmits. It's not generally supported today in IPv4.
- Total Length: Indicates the length (in bytes, or octets) of the entire packet, including both header and data. Given the size of this field, the maximum size of an IP packet is 64 KB, or 65,535 bytes. In practice, packet sizes are limited to the maximum transmission unit (MTU).
- Identification: Used when a packet is fragmented into smaller pieces while traversing the Internet, this identifier is assigned by the transmitting host so that different fragments arriving at the destination can be associated with each other for reassembly.
- Flags: Also used for fragmentation and reassembly. The first bit is called the More Fragments (MF) bit, and is used to indicate the last fragment of a packet so that the receiver knows that the packet can be reassembled. The second bit is the Don't Fragment (DF) bit, which suppresses fragmentation. The third bit is unused (and always set to 0).
- Fragment Offset: Indicates the position of this fragment in the original packet. In the first packet of a fragment stream, the offset will be 0; in subsequent fragments, this

field will indicates the offset in increments of 8 bytes.

- Time-to-Live (TTL): A value from 0 to 255, indicating the number of hops that this packet is allowed to take before discarded within the network. Every router that sees this packet will decrement the TTL value by one; if it gets to 0, the packet will be discarded.
- Protocol: Indicates the higher layer protocol contents of the data carried in the packet; options include ICMP (1), TCP (6), UDP (17), or OSPF (89).
- Header Checksum: Carries information to ensure that the received IP header is error-free. Remember that IP provides an unreliable service and, therefore, this field only checks the header rather than the entire packet.
- Source Address: IP address of the host sending the packet.
- Destination Address: IP address of the host intended to receive the packet.
- Options: A set of options which may be applied to any given packet, such as sender-specified source routing or security indication. The option list may use up to 40 bytes (10 words), and will be padded to a word boundary

The rules of TCP/IP also define a standard set of ports in use for differing types of connections. You may already know that webserver connections come into a machine on port 80. Here is a list of definitions of the most common ports.

Table 1-6. Common ports

Port #	Service	Description
20	ftp-data	Data port for ftp connections
21	ftp	actual ftpd service runs on this port
22	ssh	Secure shell
23	telnet	Telnet

Port #	Service	Description
25	smtp	Mail servers (ie. sendmail) run on this port
53	domain	Name server (ie. bind) for DNS
80	www	Web server
110	pop3	POP3 mail retrieval daemons
119	nntp	USENET news
137	netbios-ns	NETBIOS (windows file sharing) name service
138	netbios-dgm	NETBIOS (windows file sharing) Datagram service
139	netbios-ssn	NETBIOS (windows file sharing) Session service
143	imap2	IMAP mail retrieval
443	https	secure (SSL) web server

Actually your Linux box provides even more information than the above list. The file `/etc/services` is a more comprehensive list of available services and the ports they utilize. And of course you can turn to RFC 1700 (http://www.rfc-editor.org/rfc/rfc1700.txt) and later RFC's that update it for the canonical list of ports.

As a Linux user you should also be aware of the concept of *reserved* or *privileged* ports. Ports from 1-1024 fall into this category. All this means is that services running on those ports must be run by root. Regular users can open ports above 1024 for their own use.

Your machine temporarily opens up ports all the time on your behalf. When you look at a web page on port 80 of a remote machine, your machine also opens up a port above 1024 temporarily. The # of this port is in your request packet for the web page. When the packets consisting of the page are returned to you, they are returned to the temporary port opened when you initially sent the request (ie. clicked the link). After all, the page must come back to you via standard TCP/IP methods.

1.8. Network Troubleshooting

Ok, now we have talked a lot about binary math, about how to calculate subnets, now to determine what is the valid ip ranges to use for your network. You were diligent and careful, calculated all your addresses carefully, set up yourself a router to the Internet, configured everything, but it still doesn’t work. What is wrong? How do you figure it out?

One of the common mistakes I see with young technicians is the "guessing game" approach. They start making changes to things hoping to guess what is causing the problem. They do NOT take the simple approach of slicing the problem down and eliminating sections of the network, or possible items. Most times by the time they give up, they have made so many incorrect changes to the network, that it is impossible to determine what was the original problem.

Being an engineer, using the division method comes natural to me. You first start doing something to narrow down your search on the network. What are the tools available for you to use under Linux (or any other operating system for that matter)?

ifconfig (or ipconfig on a windows box) - This little command will quickly tell you what you have on your machine, what ip addresses, what interfaces, etc that are up and working. The only caveat to it all is on a windows 95 machine, the command is winipcfg, and it pops up a little gui box with the same information in it. On NT, ipconfig /all will show you all the settings. And ifconfig on Linux/unix. All operating systems that support tcp/ip have some version of this command available. All BSD/Linux ones use ifconfig as far as I know. Below is an example ifconfig from my suse Linux box:

```
bash-2.03# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:10:5A:A0:8D:37
          inet addr:192.168.3.12  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::10:5aa0:8d37/10 Scope:Link
          inet6 addr: fe80::210:5aff:fea0:8d37/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:532303 errors:0 dropped:0 over-
runs:0 frame:0
          TX packets:250499 errors:0 dropped:0 over-
runs:0 carrier:0
          collisions:75 txqueuelen:100
          Interrupt:11 Base address:0xe400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:689 errors:0 dropped:0 overruns:0 frame:0
          TX packets:689 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

bash-2.03#
```

Not only does it show me the ip address and subnet mask, but it shows me the MAC address of the network cards, the type of interface (eth0, eth1, lo, etc) but also the address and irq of each device. This is very helpful in troubleshooting hardware problems when setting up your network. Try this command on your own machine at home and see what it tells you. This is one excellent place to start understanding your own machine and network.

PING - ping is one of the MOST useful tools you could ever have in your bag of tricks. As far as I know, every networkable operating system ever written has a ping command. What does a ping command do? It basically sends a message to the address you provide and says "hey, are you there?" What is the format for a ping command under Linux?

```
bash-2.03#  ping 192.168.7.7
```

or ping *ip address*, where ip address is the address of the machine you want to check to make sure you can communicate with. Ping is one of the commands I use the most when troubleshooting network equipment and Internet connections.

PROBLEM: you just configured a computer to access the Internet using a cable modem. You set everything up, got your ip address from the cable modem dhcp, and your machine seems happy. BUT, when you open your netscape navigator and type in the url of http://www.nolug.org, it cannot find that address, and gives you either a 404 error, or host not found. What is your steps to determine what is going on? Step 1. Check your OWN machine. Ping yourself. Ping not only will work on other machines, but you can ping yourself. If you ping yourself and you don't get a response, then something is wrong with your tcp/ip setup, so don't look any further. What does the response from a ping look like?

```
bash-2.03# ping 192.168.3.12
PING 192.168.3.12 (192.168.3.12): 56 data bytes
64 bytes from 192.168.3.12: icmp_seq=0 ttl=255 time=0.238 ms
64 bytes from 192.168.3.12: icmp_seq=1 ttl=255 time=0.157 ms
64 bytes from 192.168.3.12: icmp_seq=2 ttl=255 time=0.166 ms
64 bytes from 192.168.3.12: icmp_seq=3 ttl=255 time=0.130 ms
64 bytes from 192.168.3.12: icmp_seq=4 ttl=255 time=0.158 ms
64 bytes from 192.168.3.12: icmp_seq=5 ttl=255 time=0.139 ms
64 bytes from 192.168.3.12: icmp_seq=6 ttl=255 time=0.129 ms
64 bytes from 192.168.3.12: icmp_seq=7 ttl=255 time=0.150 ms
--- 192.168.3.12 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0.129/0.158/0.238 ms
bash-2.03#
```

This is where I pinged myself on my machine at home. A ping command under Linux sends pings repeatedly until you hit the ctrl^C to kill the sending. Under windows, the ping is sent 4 times and quits on it's own. What does it look like if it doesn't ping correctly? Look at the following example of me pinging something I know isn't there.

```
bash-2.03# ping 192.168.3.254
PING 192.168.3.254 (192.168.3.254): 56 data bytes
--- 192.168.3.254 ping statistics ---
8 packets transmitted, 0 packets received, 100% packet loss
bash-2.03#
```

I ran the ping command and the first line of PING ***** showed up on the screen. Then nothing happened for several seconds so I hit the ctrl^c. Then it told me that it had sent 8 packets and none were returned. So I know that that machine is either not on, or the communication link between my machine and it is not correct.

You can also ping a domain name, i.e. www.insecure.org , or any other domain name. Below is where I pinged bellsouth by domain name. Notice I can find out the ip address of someone's domain by pinging the name. This is very useful in checking to see if your machine can translate the dns names correctly to an ip address. Remember this trick if you are having trouble connecting to a website. Ping it's name and see what happens.

```
bash-2.03# ping www.bellsouth.net
PING www.bellsouth.net (205.152.0.46): 56 data bytes
64 bytes from 205.152.0.46: icmp_seq=0 ttl=238 time=71.949 ms
64 bytes from 205.152.0.46: icmp_seq=1 ttl=238 time=70.421 ms
64 bytes from 205.152.0.46: icmp_seq=2 ttl=238 time=70.759 ms
64 bytes from 205.152.0.46: icmp_seq=3 ttl=238 time=72.407 ms
64 bytes from 205.152.0.46: icmp_seq=4 ttl=238 time=70.705 ms
64 bytes from 205.152.0.46: icmp_seq=5 ttl=238 time=71.554 ms
--- www.bellsouth.net ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 70.421/71.299/72.407 ms
bash-2.03#
```

Now if you get a response similar to the first one above on your machine, then your tcp/ip is installed and working correctly. The next step is to ping the next device in your network.

For the sake of argument, let's use my telocity adsl modem as an example. I just pinged myself, so I know tcp/ip is working on my machine, and now I need to check the next computer down the line. My next connection is my FREESCO router. It has an INSIDE ip address of 192.168.3.1. So I ping 192.168.3.1 and get the following response:

```
bash-2.03# ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
```

```
64 bytes from 192.168.3.1: icmp_seq=0 ttl=64 time=0.683 ms
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=0.583 ms
64 bytes from 192.168.3.1: icmp_seq=2 ttl=64 time=0.546 ms
64 bytes from 192.168.3.1: icmp_seq=3 ttl=64 time=0.603 ms
--- 192.168.3.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.546/0.603/0.683 ms
bash-2.03#
```

This tells me I can communicate with the inside of my router, so I have connection to the router. Next step is to ping the outside interface of my router. I do that and get a good response from it. Then the next step is to ping the adsl modem itself. I ping it's ip address and get a good response. Ok, so I am good to the modem. I look at the modem and the lights are all on showing I have connectivity, so I now need to ping something outside my network. A good place to start is your isp's dns servers. Telocity uses 216.227.36.96 as one of their dns servers. So I ping that server, and get a good response. So the ip portion of the network is working fine, and I am able to communicate to the Internet. BUT, I still can't open a webpage in my netscape browser!!! If my ip addresses are right, what is the problem? There are many different sections of networking, and different services that must all work correctly if you want your network to work correctly. TCP/IP is the basis and the base from which all unix/Linux and Internet connectivity originates. But being able to ping them isn't enough. The problem that I just describes is a classic example of not having your dns servers set correctly on the machine I am working. I can ping everything in the world, but my browser cannot find a dns server to tell it what is the ip address of www.nolug.org. Many people have this problem. They think there is something wrong with their network, something isn't working, or something is broke with tcp/ip and start changing things, when all they need to do is to go add their isp's dns server to their machine. It is a good idea to have a list of common local dns servers.

```
        216.227.108.197          Barry's dns server and web-
server running NOLUG.org
        205.152.128.20           Bellsouth's primary dns server
        205.152.0.5              Bellsouth's secondary dns server
        216.227.36.96            Telocity's dns server
```

And there are thousands of other dns servers out there that can be used to have your machine understand url names.

traceroute - Let us say you have your ip addresses correct, and your pinging different machines outside your network, but there seems to be one address or network that is unreachable. How can you find out where the problem lies? Try a traceroute to the offending address to see where your packets are going. Below is a traceroute I did to the bellsouth dns server from my machine. Notice that some hosts do not return a name, just * * * instead. There is nothing wrong with this. Using traceroute you can find out all kinds of things about the connections between you and another system.

```
bash-2.03# traceroute 205.152.0.5
traceroute to 205.152.0.5 (205.152.0.5), 30 hops max, 40 byte packets
 1  192.168.3.1 (192.168.3.1)  1 ms  1 ms  0 ms
 2  dsl-216-227-108-
198.telocity.com (216.227.108.198)  2 ms  1 ms  1 ms
 3  route-64-34-214-
1.telocity.com (64.34.214.1)  20 ms  21 ms  23 ms
 4  fe1-2-
core1.dfw.tlct.net (216.227.96.65)  25 ms  22 ms  22 ms
 5  209.246.152.61 (209.246.152.61)  22 ms  22 ms  23 ms
 6  gigaethernet6-
0.core1.Dallas1.Level3.net (209.244.15.37)  21 ms  25 ms  22 ms
 7  so-5-0-
0.mp2.Dallas1.level3.net (209.247.10.105)  22 ms  21 ms  23 ms
 8  209.247.10.110 (209.247.10.110)  22 ms  22 ms  21 ms
 9  209.245.240.138 (209.245.240.138)  24 ms  23 ms  23 ms
10  140.at-6-0-
0.XR1.DFW9.ALTER.NET (152.63.98.126)  25 ms  23 ms  23 ms
11  185.at-2-0-
0.TR1.DFW9.ALTER.NET (152.63.98.34)  23 ms  24 ms  23 ms
12  128.at-5-1-
0.TR1.ATL5.ALTER.NET (152.63.0.125)  66 ms  65 ms  78 ms
13  297.ATM5-
0.XR1.ATL1.ALTER.NET (152.63.81.29)  65 ms  67 ms  66 ms
14  195.ATM4-
0.GW6.ATL3.ALTER.NET (146.188.233.217)  67 ms  68 ms  68 ms
```

```
15  bs-stonemountain-
gw.customer.alter.net (157.130.72.54)  68 ms  70 ms  70 ms
16  205.152.37.204 (205.152.37.204)  68 ms  68 ms  68 ms
17  205.152.3.46 (205.152.3.46)  70 ms  69 ms  72 ms
18  * * *
19  * * *
20  * * *
21  * * *
22  * * ns.bellsouth.net (205.152.0.5)  71 ms
bash-2.03#
```

Notice that there are connections in Dallas-Fort Worth (DFW9.ALTER.NET) in my path to the local bellsouth dns server. Just because a machine is physically close to you doesn't always mean that the path to it is direct. This all depends on the connections that your isp has to other networks. For telocity, my packets go to the Dallas area to get to the backbone of the internet, then are transmitted to the correct subnet through Bellsouth's backbone connection. This seems stupid and wasteful, but it is the best way for redundancy to occur and prevent network outages from affecting too many people, and to allow multiple paths to the destinations.

nmap - there is a lot written about nmap, but what exactly does it do? nmap very simply goes to an address and scans to see what ports are open on the machine. We talked about the common ports earlier, and their uses. If you are able to ping a certain ip address successfully, but cannot connect to the mail server, or get ssh to connect to it correctly, then run nmap and see if the port you need is open. See the following example of nmapping an ip address:

```
bash-2.03# nmap 216.227.110.205
Starting nmap V. 2.3BETA14 by fyo-
dor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on dsl-216-227-110-
205.telocity.com (216.227.110.205):
Port    State       Protocol  Service
9       open        tcp       discard
21      open        tcp       ftp
22      open        tcp       ssh
25      open        tcp       smtp
```

```
53      open        tcp       domain
80      open        tcp       http
113     open        tcp       auth
443     open        tcp       https
993     open        tcp       imaps
995     open        tcp       pop3s

Nmap run completed --
1 IP address (1 host up) scanned in 15 seconds
bash-2.03#
```

So you see nmap gives you the details of what ports are open. If you are trying to ssh into a machine, and do not get the port 22 open, then either the machine or the firewall is NOT letting you through. Investigate further. Another example of using nmap on the bellsouth domain:

```
bash-2.03# nmap www.bellsouth.net

Starting nmap V. 2.3BETA14 by fyo-
dor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on services.bellsouth.net (205.152.0.46):
Port    State       Protocol  Service
80      open        tcp       http
179     filtered    tcp       bgp

Nmap run completed --
1 IP address (1 host up) scanned in 13 seconds
bash-2.03#
```

Now don’t just go using nmap to hit everyone in sight. If a website has an Admin like Scott, then that person will get real mad at you nmapping them over and over. Nmap is used by some people to find ports to hack into someone else’s machine. One security issue is to set up something to log who is scanning you and see what they are doing. If you are unsure of how to do this, ask someone with experience on how to do this. That is an entire topic of it’s own to explain the intimate details of securing your firewall, or your machine.

Route - running a simple route command on your machine is a quick way to see how your routing table for your machine is set up. What the routing table shows you is where it wants to send any packets you send to the network. Below is the routing table of my suse box at home:

```
bash-2.03# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Met-
ric Ref    Use Iface
192.168.3.0     *               255.255.255.0   U     0      0      0   eth0
loopback        *               255.0.0.0       U     0      0      0   lo
default         192.168.3.1     0.0.0.0         UG    0      0      0   eth0
bash-2.03#
```

Notice my default gateway is set to 192.168.3.1 which is my FREESCO Router. Any packet addressed to any network other than my 192.168.3.X internal network is sent to the router to be sent on it’s way. You must have a default route set for your machine to understand where to send packets. You also notice the first entry of 192.168.3.0. This is the standard notation for everything on my 192.168.3.X internal network. Your routing table can have many entries in it if you have a complicated internal network.

Arp - is a command to show what hosts are in your cache on your machine. Sometimes useful, but seems to be more useful to debug the crazy caching of windows. Below is the help portion of arp command from linux.

```
bash-2.03# arp --help
Usage:
  arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]             <-
Display ARP cache
  arp [-v]          [-i <if>] -d  <hostname> [pub][nopub]    <-
Delete ARP entry
  arp [-vnD] [<HW>] [-i <if>] -f  [<filename>]              <-
Add entry from file
  arp [-v]   [<HW>] [-i <if>] -
s  <hostname> <hwaddr> [temp][nopub] <-Add entry
  arp [-v]   [<HW>] [-i <if>] -
s  <hostname> <hwaddr> [netmask <nm>] pub  <-”-
```

```
  arp [-v]   [<HW>] [-i <if>] -
Ds <hostname> <if> [netmask <nm>] pub      <-”-

        -a                      display (all) hosts in alterna-
tive (BSD) style
        -s, --set               set a new ARP entry
        -d, --delete            delete a specified entry
        -v, --verbose           be verbose
        -n, --numeric           dont resolve names
        -i, --
device             specify network interface (e.g. eth0)
        -D, --use-device        read <hwaddr> from given device
        -A, -p, --protocol      specify protocol family
        -f, --file              read new en-
tries from file or from /etc/ethers

  <HW>=Use ’-
H <hw>’ to specify hardware address type. Default: ether
  List of possible hardware types (which support ARP):
    ether (Ethernet) tr (16/4 Mbps Token Ring) tr (16/4 Mbps To-
ken Ring (New))
    ax25 (AMPR AX.25) netrom (AMPR NET/ROM) arcnet (ARCnet)
    dlci (Frame Relay DLCI) irda (IrLAP)
bash-2.03#
```

Below is a copy of running the arp command on my machine at home. Not very exciting.

```
bash-2.03# arp
Address                 HWtype  HWad-
dress           Flags Mask            Iface
192.168.3.1             ether   00:60:97:6A:D9:22   C                     et
bash-2.03#
```

Notes

Wide Area Network. As opposed to your LAN, Local Area Network

The original IP address classes are described in RFC 1020 (http://www.rfc-editor.org/rfc/rfc1020.txt) published way back in 1987. RFC's, *Requests For Comments* are the documents that define the internet standards we live by. There are thousands of them and the body of them represents both the current state of internetworking as well as the evolution of the net over time. The very name indicates the open and voluntary nature of the internet. Visit http://www.rfc-editor.org to learn more about how they work.

These blocks are officially defined in RFC 1918 (http://www.rfc-editor.org/rfc/rfc1918.txt).

It was also created to alleviate the growing amount of individual routes by creating aggregate 'supernets' but we won't be discussing that topic here.

That also means that the private space designated as 192.168.0.0/16 in the previous section need not be broken into 256 /24 (Class C) blocks. It can be used simply as one whole network (supernet) of /16 size or split into subnets of varying sizes according to the user's needs.

see the **ifconfig** man page and it will tell you this

For a more detailed treatment (and the original source for this diagram and list) see this link (http://www.mtsac.edu/~adegtyar/tcpip.html#IP).

Chapter 2. Building Your Home Network

2.1. Acquiring Equipment(dumpster diving at it's best)

section unedited and in progress.

One of the biggest challenges that face a geek is where to get a supply of parts and equipment to build all the toys he/she wants/needs to learn more and have a system at home to be proud of. What are some of the local/internet resources available to the geek for acquiring equipment?

1. This NOLUG, and any other LUG/computer group. Let's face it, other geeks are a good source of parts. Almost everyone has some kind of parts stashed back in a closet or in a drawer. And we like other geeks and always want to help them out.
2. Thrift stores - one of my favorite!! Never miss a chance to go to a GoodWill, Salvation Army store, or any other type thrift store. I bought a working HP Laserjet IIIP for $24.95 at a junk store, and i bought a 20 ft Sun 13W3 Monitor cable and Sun4 mouse for $3.00 at GoodWill, and a 3com network card for a quarter ($0.25) at a Salvation Army Store. If you take the time to dig through the junk boxes and the equipment that has been donated by either people or businesses to these charitable organizations, you can find bargains. Now, you have to be persistant and look hard. Some of the computer equipment is priced ridiculously high. And it is a hit or miss kind of proposition. BUT, for the persistant geek, there are treasures to be had for pocket change.
3. Yard sales - there are some neat computer toys to be had a yard sales. Older equipment in mint condition that has been in someone's closet for years untouched. If you spend a little time and look at yard sales, you can find things that are very usable.

4. EBay (mine and Hunter's favorite toystore!). Don't sell the online auctions short. Yes there are things that are sold at way above their retail value, but network parts and unix/vax machines and sun stations go sometimes for ridiculously low prices. Like all methods of scrounging parts, it just takes time and persistance.
5. Where you work, schools, etc. As someone who works in the computer industry doing network consulting, there are tons of perfectly good equipment that is sitting in storage rooms unused at all kinds of businesses. Some of these businesses will give you the stuff if you haul it off. Especially if you tell them it is for the NOLUG organization, and will be used for educational purposes. A lot of times they just want the space cleared up and rid of the old equipment.
6. Dumpster diving itself! Now it sounds strange, but keeping up with when schools and businesses throw away equipment is a good way to acquire stuff. Some businesses and especially schools had rather throw stuff away rather than give it away. Doesn't make sense, but it is true. So, if you know someone at a school, or in an IT department at a business, ask them to keep an eye out for equipment being thrown away.
7. The newspaper. There are sometimes some good deals in the local newspaper. Not often, but a few.
8. Closeout/junk tables at computer stores. Treasures abound in these places. Especially in bigger cities like Houston.

So, in a nutshell, finding good CHEAP sources of equipment to feed your geeky habits just takes time and patience. Now, over time you can end up with a collection of stuff to use, but it won't happen overnight.

2.2. Network Cards

One of the most important pieces of equipment you need to acquire for your home network is network cards. Now, there is no reason to go spend big bucks on a name brand, top of the line network card, when there are some name brands that have bargain prices on them. If you are running a windows box, any network card will go in there and have drivers for it. But, linux is a little different. Now, Linux will run a LOT of cards, but the easiest way to make sure you have a card that will easily be recognized by Linux is to stick with either a 3com, or a NE2000 compatible. I have had really good experiences with the D-Link products. Almost every card they make is a NE2000 compatible. Even their PCI network cards. And they usually run under $20 at CompUSA. So if you can't find any cards anywhere else, there are cheap ones that can be had at the store. Just do yourself a favor and research the cards before you grab a bargain card to make sure it will easily run under linux. For most networks, 10baseT cards are sufficient to run a home network.

There are different types of network cards. Some have jumpers on them (older 10baseT cards such as Novell Anthem, OLD 3coms, etc), but most cards today whether ISA or PCI are autoconfiguring, and have a driver disk that comes with them. You can also download this config disk from the manufacturer's website. You will also need a DOS disk around to boot from. Configuring most cards is a simple exercise. Put ONE and only one card in the computer, boot to a dos disk, then run the config utility from the manufacturer's disk. Turn off ANY plug and play features, and set the i/o address and irq to something other than default, yet doesn't conflict with any other device. Write the config to the card, then write down the i/o address and irq that you just used. Then power down the computer.

Now do yourself a BIG favor. Write the information you just wrote down about those cards on a sticky and tape it securely to the front cover of your router. Just in case you need to reconfigure later, or your floppy gets corrupted or you upgrade to a new version of freesco, you will need this. Also, it is also a good idea to take a sharpy or a piece of masking tape and write on the outside of the metal tab on the network card what address each one is as you configure it. That way once they are in and you don't remember which is which, you can plug the cables in right.

Now, remove the one you just configured and install the other network card. Repeat the same process only use a different i/o address and irq for the second card. This time when you shut down, put the first card back in so both are in the computer. Now you are ready to move on to configuring a router.

2.3. Hubs & Switches

Once you have gotten your NICs for your computers, then you need to determine what you want to use to connect all of them together. Now there are switches and hubs. Both of them will work, but for a home network, you do NOT want the expense of a switch. Not unless you get a bargain on one on eBay. A simple hub is sufficient for a home network. You can pick up a 10baseT hub very cheaply on eBay, or at the store, or online mail order. Unless you just want the expense and speed, a cheap 10baseT network will be plenty for you to share the internet and network your house. Do yourself a favor when looking for a hub. Figure out the maximum number of computers you think you will ever want to have running at your house, and then add a couple to that number. What good is a network without a couple of free hub ports open for your friends to come over and plug into your network?

Now switches are excellent pieces of equipment. They allow you to segment your network and do all kinds of fancy network management activities like monitoring traffic, monitoring ports for transmission errrors, etc. But, they are sometimes difficult to install and configure, depending on the brand and the style. For a home network, unless you are a network engineer, or have a definite reason for a switch, get yourself a hub. They are very easy to set up. Just plug in the power, then plug in your cat5 patch cables from your machines, and you are set up and running.

2.4. Cables & Connectors

So you’ve got your cards, now you need your cables & connections to start linking your LAN together. This will partially be determined by the connectors on your cards. Their are 3 basic types of connectors on modern cards.

The first is the BNC-style connector. This is for thinnet ethernet which is is ethernet transported over coax (also known as 10base-2). The cables are very similar to those used for your Cable TV though the end connectors are a bit different. Thinnet connects in a single long chain from computer to computer. Each card has a T shaped connector attached with the coax entering and leaving each leg of the T. A 50-ohm terminator must be installed on each end of the cable. Thinnet segments should not be longer than 600 feet.

If you have older equipment, thinnet may be your best option. Since all hosts are linked in a chain, you don't need a hub either which may save you money. Still 10base2 is on it's way out.

The most popular modern option for cabling is CAT5 which can carry both 10baseT and 100baseTX. CAT5 cable is also referred to as UTP, Unshielded Twisted Pair. CAT5 cable consists of an outer jacket holding 4 pairs (8 individual) of copper wire. It is terminated by modular RJ45 plugs which are similar to telephone plugs (known as RJ11). The Twisted Pair cabling is very similar to telephone premise wiring. CAT5 segments should not be run longer than 650 feet; the more segments connect to your hub equipment, the less length you should extend your segments without signal-boosting equipment (ie ethernet repeaters).

UTP cabling does require a hub, allowing you to connect hosts in a "star" pattern off the hub. UTP is fairly inexpensive to buy premade patch cables for. If you make a lot of cables, it's worth your while to buy the cable in bulk and tip it yourself.

To terminate CAT5, you really need a modular plug crimping tool and a CAT5 stripper. Both of these items as well as cable and tips can be purchased in a number of locations including Home Depot. 10baseT and 100baseTX uses wires 1,2,3 & 6 inside the cable jacket to provide communications. This means that technically you can use the other 2 pair in the cable for an additional connection but this is generally not a good idea.

The colors of each of the wires help in setting up the cable and there is a standard method to line up the wires in a connector to allow transmission. The wire color order for a standard RJ45 tip is orange-white,orange,green-white,blue,blue-white,green,brown-white,brown. If you've never tipped CAT5, describing it in print isn't going to help you much, so find a friend and have him show you how it's done.

Lastly your card may have a 15pin AUI port. This port typically requires the connection of an additional device, an ethernet transceiver, to operate. There's really no reason why you'd need to fool with AUI+transceiver in a home network configuration.

2.5. DHCP - Assigning IP addresses

DHCP, Dynamic Host Control Protocol, is a method for assigning IP addresses to hosts on a network. A DHCP server running on a network manages a pool of available IP addresses and doles them out to hosts that request them. This way, machines can be added to a network dynamically so if your buddy comes over with his shiny new laptop, he can hop on your LAN with ease. Otherwise, you must statically assign an IP address to each machine on your network. This probably easy to manage in a home network, but it can get annoying quickly

Every NIC card has a unique MAC (Media Access Control) ID number. When you plug a host into the network running a DHCP client, it sends a packet out on the network broadcast (255.255.255.255). In that packet, is a request for an IP and the NIC card's MAC ID. The DHCP server is configured to listen for these requests. When it receives one, it checks the pool of available IP addresses. It then *leases* an IP to the MAC ID, returning another broadcast packet out on the network. The DHCP client receives the acknowledgement and the packet contains data telling the host it's IP and generally your other routing information. The DHCP reply packets will usually handle the job of telling a host it's default route, it's DNS servers, and many other configurable pieces of information as well.

Lets take a look at a sample dhcpd.conf file:

```
# $OpenBSD: dhcpd.conf,v 1.1 1998/08/19 04:25:45 form Exp $
#
# DHCP server options.
# See dhcpd.conf(5) and dhcpd(8) for more information.
#

# Network: 192.168.1.0/255.255.255.0
# Domain name: my.domain
# Name servers: 192.168.1.3 and 192.168.1.5
# Default router: 192.168.1.1
# Addresses: 192.168.1.32 - 192.168.1.127
#
shared-network LOCAL-NET {
option  domain-name "my.domain";
option  domain-name-servers 192.168.1.3, 192.168.1.5;

subnet 192.168.1.0 netmask 255.255.255.0 {
option routers 192.168.1.1;

range 192.168.1.32 192.168.1.127;
}
}
```

This is pretty straightforward. It sets up a pool of dynamically assignable IPs from 192.168.1.32 to 192.168.1.127 inside of network 192.168.1.0/24 . Furthermore, it tells those machines that their default route is at 192.168.1.1, they have a domain name of "my.domain" and name servers at 192.168.1.3 and 192.168.1.5. The sample file reserves the first 30 IPs in 192.168.1.0/24 for static IP machines. You can also add entries to assign specific IPs to specific MAC addresses thus assuring certain machines get the same IP every time.

DHCP is infinitely configurable but the example above -- with appropriate entries for your local network should be sufficient for home network. The only caveat with DHCP is that you must have your network carefully segmented. You should not have two DHCP servers attempting to assign IPs on the same network segment. So if you have a DSL connection to the Internet, you need to have two IPs in one Linux box that connects to both the DSL router and the other to a hub connected to your internal LAN. The reason is that your Internet provider will also be attempting to assign IPs via DHCP and you will have problems if your network is not segmented this way.

2.6. NAT/IP Masquerade

NAT (Network Address Translation) and IP Masquerade or two terms for the same thing. IP Masquerade is the name of the Linux implementation of NAT so the terms are essentially interchangeable. Since NAT is a standard term and is used in many machines besides Linux boxes, I will use the term NAT throughout. If you read "IP Masquerade" in a document, just think NAT.

NAT solves a common problem. Your Internet provider, whether dialup, cable or DSL, generally only provides you with one routeable IP address. If you can get more, it will usually cost $bigbucks. They're not just being stingy, "real" IPs [1] are becoming scarce. Your ISP has to justify their IP usage to their upstream providers. NAT is one way to help to with scarce IP resources.

NAT allows your Linux box to share your single routeable IP with all the hosts on your private LAN. The NAT-enabled PC has two NICs in it. One is connected to your internal LAN via a hub, the other is directly cabled to a cable modem, DSL modem, or (you poor soul) a dialup modem. The NIC connected to your LAN has an IP address in your private IP space. In the example dhcpd.conf in the previous chapter, the default router for the network is 192.168.1.1; this would be the IP for this network card in your NAT box.

The IP for your other card (or dialup modem, you poor soul), can either be assigned by your ISP or may be statically assigned. It is your "external" routeable IP.

When a host on your LAN wants to send something out to the Internet, it sends a packet to it's default router, which is your Linux box's internal NIC on the LAN. NAT takes this packet and adds additional header information to translate (aka masquerade) it to your external IP address. It then sends it out to your ISPs default router for its destination. When a reply is made to the packet, the NAT box receives the packet intended for the machine on your internal LAN, removes the extra header information, and forwards the packet to the appropriate host on the internal LAN. The NAT software maintains a table of opened connections to track packets.

The easiest way to understand NAT is to think of a letter wrapped inside of two envelopes. The outer envelope contains the address of a building you are sending your letter; the internal envelope contains the floor and office number of the person you are sending the letter to.

By it's very nature, NAT provides a bit of extra security by obscuring your hosts on an internal network. By not having them all running on public IPs, they are less susceptible to attack. But you should not rely on NAT as your sole security mechanism, the host running NAT should be running firewalling software as well to protect it from attack. If someone breaks into it, after all, they have full access to all the machines on your LAN. NAT in Linux relies on ipchains to do its magic so you're partly there anyway.

Lastly, not everything works through NAT. FTP often doesn't work "out of the box". You may need to tell your FTP client programs to run in "passive" mode for their connections. The symptom of this problem is connecting to an FTP server and the connection just hangs without generating any output. If you ICQ, you need to send all your messages through the server. NAT just adds a little layer of complication. If you remember that your machines on the local network have no direct knowledge of the world beyond their default router, you should be able to troubleshoot most NAT-related networking issues. With NAT bing such a popular standard these days, it's rare to find an application that will not work at all with it.

2.7. Cable/DSL

Now that we have talked about all the goodies you need to find to set up a home network, you need to decide what isp you are going to use to connect this pile of non-homogeneous collection of equipment to the internet. None of us could live without our internet fixes. Now, if you have very limited funds, or are still living at home, etc, you may be stuck with a dial up modem. BUT, if you are looking at a second phone line for your computers, then a dial up isp on top of that, you are looking in the $45 a month bill for the two, PLUS the cost of installing a second phone line which isn't cheap. In my opinion the best way to go is either cable modem or dsl. For under $50 a month, and most of them are waiving any setup fees, and some, like telocity don't even require an annual contract. If you can get either in your area, do it. Now there are always big debates about which is better, dsl or cable. If you can get both, go with whichever is the fastest and cheapest. Either is better than dialup!!!

Now with that said, what do you need to know to connect a linux box, or a freesco router to the internet using cable/dsl? First you need to know what the ip address scheme they are assigning you. Most isps give you a paper that has all the mac addresses, ip addresses, and subnet masks, and their dns server names and addresses, along with the name of the mail and new servers they provide. Do NOT lose this piece of paper. All of that information is what you need to give your linux box for it to find it's way onto the internet.

For this section I am going to talk about connecting a normal linux machine to the internet and not a freesco router. I will also assume you have your linux installed and running. Put your network card in the computer if it is not already there. Boot your machine and it should find the card. Now depending on the distro you are using, SuSE uses yast or yast2 to add new hardware, other distros use different utilities. Consult their websites to determine how to configure the driver for your system.

Now that the card is configured to be recognized, it needs to be assigned an address. Do this by either dhcp from another machine running dhcp across your cable/dsl modem, or if you have a static ip address use it. On most linux systems there is a neat utility called pump. I don’t remember what pump stands for, but it will dynamically configure your network settings from a dhcp server. As root do **pump -i eth0** . After a pause it should silently return you to the command prompt. This means it worked and your card is configured and the address is assigned. When in doubt about the correct configuration for connecting, and if your first tries don’t succeed, then ask the nolug list for help.

2.8. Building a home Linux router (freesco)

1. take your old doorstop type computer (ie 386, 486, low end pentium) and take the cover off. Remove the hard drive, and sound cards, any other extraneous devices that are in the machine. Leave a single 1.44 meg floppy drive in the machine, and whatever video card you have in there. Obviously leave the motherboard and memory in there :-)
2. next, determine how much memory is in the computer. (usually powering on the computer will quickly tell you what is in it). I recommend having at least 16 meg in the computer. FreeSCO runs off of a ramdisk of memory, and you can add additional features if you have more than 16 Meg of memory.
3. One additional thing you can do. If you can find an old ega/cga monitor and video card, replace the one in the router with them. Those old monitors are junk as far as most people are concerned, so you can get them for free. I have several of the old

monitors and have given several to members of NOLUG for their use.

4. Ok, so now we have a lean mean FreeSCO machine!! NO extra bull in the way, just what is needed. Now we need to move on to the bios setup on the machine. If you have a really old 386 or 386sx, then there may not be any settings to change in the bios, nor a way to get to the bios. Most 486 and all low end pentiums have a way to get into the hardware settings, or bios settings. On different computers it is a different key combination to get in. If your computer has Pheniox bios, more than likely you need to hit F2 while it is booting to get into the bios. Most, if not all Award bios use the del key to get into the bios. Compaq computers (just to be obnoxious and hard to deal with) use F10. There were some older bios that required you to hit cntrl-shift-esc, or cntrl-enter, or some other weird combination of keys. If your machine doesn't show a hit **** to enter setup at boot time, then try to look up the motherboard on the web and see what the key combination is set to on that variety of bios. Ok, so now we are in the bios. What now??? look around the bios and find any settings that are turned on and determine if you need them (at this point you may want to ask for help from someone if you are not familiar with bios terms). Things that you should turn off completely: that annoying power management crap. Remove any reference to a hard drive and set all the hard drive settings to not installed. If there are any strange devices like built in modems on an old packard bell, turn them off. I usually turn off all the serial ports and parallel ports too. Turn off any embedded device you don't need, which on a freesco router for your home, you won't need any of them. Make sure your 1.44 meg floppy is set to be Drive A, or the boot drive. Also, if your computer has the ability to select the boot order, make sure the floppy drive is the first boot device, since it will be the only bootable device in the computer.

5. Now we have our LMFM with all the junk stripped, the bios set to not operate any extra junk, and we are ready to start on the network cards. As I talked about above, we go through the setup routine of one card at a time if they are ISA cards. If they are PCI network cards, 3com, or dlink novell compatables, then we just shove them in and start configuring the router. But for the older cards make sure they are configured correctly and NOT conflicting with each other.

6. Now comes the meat of the subject. Our machine is physically together and ready to rumble on your cable/dsl. Go to www.freesco.org or www.freesco.com and

download the 1.44 meg boot image. Once this is downloaded to your drive, then you need to write it to a floppy. If you are on a windows box, download rawrite, or rawrite2 to your machine and use it from a command prompt. If you are on a linux machine then use dd to make the disk. The format for the commands are as follows:

```
dd if=1.44imagename.ext  of=/dev/fd0
```

this will write you a floppy disk in your first floppy drive on your linux machine.

```
Rawrite [enter]
name of image file?:   1.44imagename.ext [enter]
destination drive?:   a: [enter]
```

this will write you a floppy in your drive a: of your dos/windows machine

7. Now that you have your bootable freesco disk in hand, we are ready to play and configure the router. Put your floppy in your newly built router and turn it on. When it gets to the boot prompt, type setup and hit enter. This runs the setup routine to configure the router for the first time. Now it will pop up a menu that shows you some different things. Just hit enter and go to the next screen. You will see a listing of all the difffent types of configs you can set up a freesco to do. We will be concerned mainly with (e) an ethernet router between your network and either a cable or dsl modem. Hit e and then it will go through many questions about your setup. Remember the sticky or tape I told you to put on the outside of the case of your router??? One of the first questions you will be asked is what is the I/O address of your first network card, or eth0. Chose the card with the lowest I/O number for the first interface. Put in the I/O, then it will ask for the IRQ. Next it will ask you for the second i/o and irq of the second interface, eth1. Enter these numbers, then the next question is do we want to use DHCP to configure the first interface?? Now this answer depends on your service provider. If you have a static ip address assigned to you by your isp, then just enter the ip address they gave you into the config. Now if you are running telocity, or one of the service providers that give you a modem that does dhcp for you, then you tell it yes and let it configure itself. With Telocity, you have a static ip address, but internally, the telocity gateway modem does DHCP. What it does is it has ONE ip address to give, and always gives that same ip address to your router.

8. Next it will ask you about what ip address to use for the eth1 or second network card. Now the answer to this question depends on the tcp/ip class from the last session. Hopefully you have chosen a valid ip addressing scheme for your internal network of something like 192.168.1.0/24. For this example, let us assume you did this. The normal numbering convention is to make your router, or default gateway of your network be dot-1, or in this example, 192.168.1.1/24. So put in this address into the freesco as 192.168.1.1, then hit enter. The next question will be what subnet mask. We are using standard class C addressing, /24, so you enter 255.225.255.0 then hit enter.
9. Next you go through the options and determine what you want on and off. A couple of suggestions. Turn on the extra modules if you have 16 meg or more of memory. Also, do NOT turn on the telnet server, or any of the other services, except the http control interface. Turn this on. It is very useful. I use it to reboot my router, or reconfigure my firewall from work. Very nice interface. I would not use your router as a print server unless you have a pentium box using as a router. But read the documentation on the other services and make your determination as to what you want to turn on. Remember, there is a downside to almost all of the service you turn on inside freesco. I think you are better off not using the ones in FreeSCO, and using services from behind the firewall on other machines.
10. One other service you might want to turn on for the router is dhcp server. Make the freesco give out the ip addresses to your other machines. This allows you to plug and unplug machines to your network and not have to manually keep up with ip addresses. Just heed their warning. DON’T reserve a ton of ip addresses for dhcp. This eats memory and will slow down your router. If you have 4 computers on your network, then reserve 6 or 8 addresses for the machines. Always give yourself extra for future stuff or friends visiting. Just make sure you have this service turned on secure, or locally, not world-wide. This is true of ALL services except the control panel on port 82.
11. Exporting services. Now this is probably the easiest thing to configure, and the most important for most people. This is where you set it up to allow you to ssh into your workstation from outside your firewall, set up a mail server, or run a webserver. When it asks you do you want to export services, hit yes. Then hit enter to get into the editor. It is a lot like an old dos editor, very simple and functional.

Now they have an example of what the syntax of the lines needs to be. It is very simple line like this: tcp,53,192.168.1.15/53. Now what this shows is this. Open a tcp port on the firewall of port 53, and direct any hits on that port to the machine with the address of 192.168.1.15 and hit port 53 on that machine. Now you can see how you could have directed port 53 on the firewall to port 64000 on the .15 machine. This is called redirection. You must refer back to the class notes from tcp/ip to determine what ports you need opened for each service on your firewall. It all depends on what you are wanting available to the world from within your network. My listing of exported ports is 15 lines long or more. But I am running a dns server (53) mail server (110,25) webserver (80) vncserver(5901, 5800), ftp server, etc. But be careful with what you open up. Each opened port on your firewall is a potential area for someone to hack your system.

12. The rest of the questions and services all depend on your particular needs. If you run into trouble with your configuration, email the mailing list and myself, scott, or konrad or one of us would be more than willing to help you get it going right.
13. Now it is all configured and ready to go, exit out and save the configuration. It will right to your floppy the configuration. Now, reboot the machine, even though it will try to finish the boot process. Now when it boots you should see it doing some things, let it run then it will clear the screen and start initializing hardware. This is where you want to watch it closely. You should see it initialize the loopback interface, l0, then it will get to a point of getting an ip address for eth0 if you are running dhcp. Now sometimes this take 15 or 20 seconds. Hit the cntrl-shift-F3 to see the logfile if it freezes for more that 30 or 40 seconds. In the logfile, it will say no DHCPOFFERS on the last line if it was unable to get an ip address. Then you have something wrong with the configuration. Go back and check everything, especially the card settings. Also, if everything looks right, reverse the cards and use the second card as the first interface and vice versa. Sometimes for some unexplainable reason this makes a difference.
14. Now that you have it all configured, and booted, then you should be able to access your network through the router. Just go to each of your other machines and set the default gateway on all of them to be the inside interface of your router, i.e. 192.168.1.1 in our example.

2.9. Dialup to the Internet (you poor soul)

Sometimes you're stuck with dialup access. Or maybe you just want to have a backup link for your cable or DSL connection. Either way, you need to be able to set up a modem under Linux.

Luckily, it's much easier than it used to be to get your modem connection running. The first question thing you need to figure out is what kind of modem you have. If it's an external modem, you're good to go. If it's an internal, there's some things you need to figure out.

The first issue is Winmodems. If your computer has one of the aberrations of nature, you will not be able to use it in Linux. A Winmodem offloads most of the modem functionality that would run in hardware with a normal modem onto the processor of your PC. They are completely proprietary. If you have a PCI modem,chances are it's a winmodem. If it's an ISA modem, you may be lucky enough to have jumpers on the card to set up your COM port and IO address. If it's a PnP ISA card, you will have to use isapnp to set up these parameters. Most modern Linux distributions will handle this "out of the box". So unless you've got something really old and esoteric or you've got a PCI Winmodem, you're good to go. And if you're not, finding an external modem cheap shouldn't be too hard.

The easiest way to check for a modem is via your trusty command line. Enter the command **dmesg | grep tty** and you'll see something like this:

```
ttyS00 at 0x03f8 (irq = 4) is a 16550A
ttyS01 at 0x02f8 (irq = 3) is a 16550A
tty03 at 0x03e8 (irq = 3) is a 16550A
```

In Linux, our COM ports have a different naming convention. COM1 in windows is /dev/ttyS00 in Linux. COM2 is /dev/ttyS01, etc. etc. In the listing above, I have three such ports listed. On the back of my box is two serial ports, so the third port is my modem. woohoo.

The more you know about your modem when you start, the better off you'll be. The nice thing about Linux these days is that most distributions can set up a dialup to an ISP as easily as windows. KPPP, gnome's PPP setup, Redhat netconfig, and debian's pppconfig all mimic Windows' Internet connection wizard. They will attempt to detect your modem and usually succeed. They will then ask for your username, password, and phone number. And that's about it. At one time, you had to worry about scripting your connection to match prompts from your ISP's servers. Now most ISP's use PAP authentication so this step is standardized. If you're in doubt, try PAP first and use your distribution's PPP setup utility's debug window to watch the connection progress.

In an attempt to be more versatile some of the Linux utilities have lots of additional options for setting static IPs, scripting connections, etc. Don't try these options unless you're absolutely sure you need them. Do everything dynamically and allow your ISP to assign your address, set your default route, and even set up your nameservers. Unless your ISP is really, really dumb, it will work as easily under Linux as it does with Windows. Be happy because this used to be much, much harder to do.

Notes

as opposed to your reserved IP space in your internal 192.168.1.0/24