How To Set Up A Linux Network

Before we get into setting up Linux networking on a Debian system, we'll cover the basics of how to set up a network with both Windows and Linux systems and how to make it a "private" network. Here the term "private" may not mean what you think it does. It has to do with the IP addresses you use on your home or business network. You'll then understand the value of having a proxy/NAT server or a firewall system which also performs the proxy/NAT function on your network.

Once we cover the "whys" and "whats" we'll get into the "hows". You'll see how easy it is to set up a home or small-business network including what hardware is needed. We'll briefly mention what you need to look at on Windows PCs and present more in-depth information on which files are used on a Debian system to set up networking. The **Network Configuration Files** section shows what files are involved in setting up your Debian system to work on a local network and how they need to be configured to enable the various functions involved in networking including being able to connect to the Internet.

> **Note:** Even if you're not familier with TCP/IP networks, try giving the material on this page a shot. It's presented in an introductory manner. Don't be concerned if you don't understand all of the material on OSI layers, subnetting, etc. presented on this page. Understanding this material is not necessary when setting up a network or using the subsequent guide pages on this site. It is merely presented for those who wish more in-depth information.
>
> Even if you don't have a network you can still play around with the material present on this page and on the Proxy/NAT and Firewall pages. See the **No-Network Network** section below on how to do this.

What's particularly appealing about Linux for small businesses and non-profit organizations is that you can set up both internal (file, print, database) servers, external Internet (Web, e-mail, ftp) servers, firewalls, and routers (yes, you can set up a Linux system to be a router too) for very little cost. The operating system and server applications are free and, given that Debian will run on older hardware, the hardware costs can be minimal. These attributes also make it a great toy for those wishing to learn more about networking. Pick up one CD set and you can set up all the Linux servers, firewalls, and routers you want and experiment your brains out.

"Private" Networks

Theoretically, every system on a network needs a unique identifier (a unique address). As such, every system that accesses the Internet would need a unique IP address because TCP/IP is the protocol of the Internet. However, when the Internet exploded in the mid-90s it became clear that there simply were not enough addresses available in the TCP/IP address space for every computer in every office of every Internet-connected organization. That doesn't even take into account those who wanted to access the Internet from home.

The solution was to create "private" address ranges to be used in conjunction with "address translation". Lets look at the first piece first.

Three blocks of IP addresses were set aside as private, meaning that all of the routers on the Internet would be configured to **not** route them. That's why private addresses are also referred to as "non-routable" addresses. The benefit? If packets from systems with private addresses weren't routed between Internet-connected networks, then a whole bunch of networks could use the same private addresses because they'd never "see" each others addresses. In other words, these same addresses could be used by any number of computers around the world because if they weren't routed, it would never be "discovered" that they weren't unique.

So if they're not routed, how do you get on the Internet if your computer has a private address assigned to it? That's where the second piece, address translation, comes in. Normally, in order for all the computers in a company to have Internet access they would all have to be assigned routable ("public") IP addresses that could pass through the Internet. Since there aren't enough addresses for this, companies instead assign all of the hundreds of computers in their organization private addresses and they all share a single "public" address to access resources on the Internet. This sharing is accomplished by configuring privately-addressed systems to use a special server, called a **proxy server**, to access the Internet.

A proxy server has two NICs (Network Interface Cards) because it's connected to two different networks. One NIC is connected to the Internet and is assigned a single "public" (routable) IP address. (This NIC is referred to as the "external interface".) The other NIC is connected to the company's internal network. It is assigned a private IP address so that it can communicate with all of the other privately-addressed computers in the company. (This NIC is referred to as the "internal interface".) The proxy server acts as a "gateway" onto the Internet. (Because of the gateway behavior, a proxy server should also have firewalling capabilities to protect the internal network.) However, in addition to acting as a gateway, it acts as an address translator.

Web2PDF
converted by Web2PDFConvert.com

The private IP addresses assigned to the systems on your internal nework are chosen by you from one of the three private address ranges listed below.

Public IP addresses are only available from an ISP. In most cases, such as with a dial-up, DSL, or cable modem, your ISP automatically assigns a single public address to your modem using PPP, bootp, or DHCP. This assigned address can change from time to time ("dynamic"). It requires no configuration on your part. Business customers typically obtain multiple public addresses from their ISP. These addresses do not change ("static"). Static addresses are needed for Internet servers that are referenced by DNS records such as Web servers, mail servers, etc. that are contacted using a domain name.

When a computer on the internal network with a private address wants to request information from a Web site, it actually sends the request to the internal interface of the proxy server. The proxy server, with it's public routable address on the external NIC, is the one that actually sends the request to the Internet Web server. The Web server sends the response back to the proxy server's external NIC, and the proxy server then forwards the response on to the computer on the internal network that made the initial request. The proxy server keeps track of which internal computers make which requests.

The advantage? Hundreds of computers in a company can access the Internet and only take up a single public Internet address (that of the proxy server's external NIC). Another advantage is security. If your computer's address can't be routed over the Internet, it would be hard for someone to get at your computer *from* the Internet. (There are ways though.)

The translating of a private address to a public address (outbound request) and back again (inbound response) is most commonly known as **NAT** (Network Address Translation). In the Linux community it's also often referred to as "masquerading" because the proxy server hides the true identity of the internal computer that made the initial Internet request.

The internationally-established private IP address ranges that can be assigned to internal network computers are as follows:

- **10.0.0.1** through **10.255.255.254**
  - 16,777,214 addresses
  - 16,777,214 computers on 1 network (10.x.x.x)
  - Uses a subnet mask of **255.0.0.0**
  - First octet must be the same on all computers
  - A Class **A** address range
- **172.16.0.1** through **172.31.255.254**
  - 1,048,574 addresses
  - 65,534 computers on each of 16 possible networks (172.16.x.x to 172.31.x.x)
  - Uses a subnet mask of **255.255.0.0**
  - First two octets must be the same on all computers
  - A Class **B** address range
- **192.168.0.1** through **192.168.255.254**
  - 65,534 addresses
  - 254 computers on each of 256 possible networks (192.168.0 to 255.x)
  - Uses a subnet mask of **255.255.255.0**
  - First three octets must be the same on all computers
  - A Class **C** address range

Clearly, with 254 possible addresses on each of the 192.168.x.x private address ranges, using one of these ranges is plenty for most small businesses and those who want to play around with a network at home. (This is why, you may recall, the default IP address in the "Network Setup" part of the Debian installation was 192.168.1.1.) The 10.x.x.x address space is often used by very large organizations with many dispersed locations. They will "subnet" this large private address space so that one location will have an address range of 10.3.x.x, another will have 10.4.x.x, and so on, with each location having the ability to have up to 65,534 computers. Each location may even further subnet their address space for different departments or facilities. For example, in the location that has the 10.3.x.x address space, the engineering department will have the 10.3.2.x space, the accounting will have the 10.3.3.x address space, etc. with each department being able to have up to 254 computers. (You'll see where these numbers come from later.)

Each of the numbers separated by periods in an IP address is referred to as "octet" because the value of the number (0 to 255) is derived from eight binary bits. An IP address actually consists of two parts. The first part of an IP address identifies the **N**etwork that a computer is on, and the other part identifies the individual **C**omputer on that network.

There's an often-used analogy comparing an IP address to a telephone number. The network part of the IP address is analogous to the area code, and the computer part of the IP address is like the individual's phone number. All the people on phone company network (in one area code) have the same area code number, and no two of them have the same phone number. Conversly, two people can have the same phone number in different parts of the country because they're not in the same area code (not on the same network). It's when you put the area code and an individual's phone number together that the number becomes one that is

converted by Web2PDFConvert.com

unique to the entire country (internetwork). In this context, we call the area code the **prefix**. With IP addresses, the network part of an IP address is the prefix.

> **Note:** If you get into routers you'll learn about another instance where this telphone number anology holds true. The long distance carriers don't care what the individual's phone number is. All they look at is the area code and route the call to the baby bell's appropriate switching center. Likewise, most network routers don't care about the computer part of an IP address. They only look at the network part of the address because routers route packets between networks, not PCs. The only router that cares about the computer part of the address is the "final hop" router (the router which is connected to the network segment that destination computer is connected to).

With national phone numbers the prefix is always three digits. Unlike national phone numbers, the prefix of an IP address can vary in length. That's where a **subnet mask** comes in. It determines how much of the IP address is the prefix (the network part). If you have a subnet mask of 255.255.0.0 it means that the first two numbers (octets) in an IP address are the network part (prefix) of the address and the last two numbers (octets) are used to identify the individual computers on that network.

```
                213-555-1212
                172.16.3.187
                  N.N.C.C
                255.255.0.0
   1111 1111.1111 1111.0000 0000.0000 0000
```

Note that a '1' in a subnet mask indicates a **N**etwork bit and a '0' indicates a **C**omputer bit. Because of the way an IP address is split into two parts (network part followed by the computer part), a subnet mask will always be a series of consecutive 1s followed by a series of consecutive 0s. In other words, you'll never see a subnet mask where 1s and 0s are interspersed (ex: 11100110).

Here's a few points you need to be aware of when you assign IP addresses to systems on a network:

- All systems on an internal network must have the same subnet mask
- The **network** part of the IP address must be the same for all computers on the internal network
- The **computer** portion of the IP address must be different for every computer on the internal network

That second point is important. With a Class C network the subnet mask of 255.255.255.0 indicates the first *three* octets are the "network" part of the IP address. That means that the first three numbers (octets) of the IP address must be the same on all of the computers on this internal network. That leaves only the last octet available to identify individual computers on the network (8 bits equals about 254 computers that can be uniquely identified, a value between 1 and 254 for the last octet).

> **Note:** People are often confused by the 192.168.x.x private address range because of it has two 'x's. The first 'x' means you can pick **one and only one** number from the range of available values (0 through 255) for your network. Because this is a Class C address which uses a subnet mask of 255.255.255.0, the number you choose for the first 'x' must be the same on all systems on your network.
>
> You can set up a network with 192.168.1.x addresses and another network with 192.168.2.x addresses but because these are Class C address ranges they are two totally separate networks. You would need to set up a router to inter-connect the two networks in order for the machines on both networks to talk to each other. If you think that, given the number of servers, workstations, network printers, switches, etc. you may have in the forseeable future may grow to more than 254, you may want to consider using the Class B private address space on your network.

The range of values for the last octet (to assign to computers) on a Class C network is 1 to 254. So you can only have 254 computers on a network that has a subnet mask of 255.255.255.0 (a Class C network). A '0' in the last octet is reserved as the "wire" address for the network and 255 in the last octet is the broadcast address for the network. No system can be assigned either of these numbers. The following addresses can't be assigned to computers when using the private IP address ranges:

"Wire" or "Network" Addresses:

10.0.0.0
172.16-31.0.0
192.168.0-255.0

Broadcast Addresses:

10.255.255.255
172.16-31.255.255
192.168.0-255.255

Because you can't use these two (wire and broadcast) addresses to assign to computers, the number of addresses that you can assign to computers can be calculated using the equation:

converted by Web2PDFConvert.com

$2^c - 2$ = number of available computer addresses

where **c** is the number of **c**omputer bits in your subnet mask. Now you can see how we came up with the number of computers you could have on each network when we gave the three private address ranges earlier. With a class B address you have 16 computer bits in the subnet mask:

213-555-1212

```
    172.16.3.187
       N.N.C.C
    255.255.0.0
1111 1111.1111 1111.0000 0000.0000 0000
```

so using our equation we end up with

2^{16} = 65,536 - 2 = 65,534 available computer addresses

On a class C network we have 8 computer bits in the subnet mask so:

2^8 = 256 - 2 = 254 available computer addresses

Earlier we said that private address ranges and NAT help conserve IP addresses but the use of subnet masks can also. For a more thorough explanation on IP address classes and the use of subnet masks check out the **Subnetting** section later in this page.

You may run into networks in businesses and schools where the computers **don't** have private addresses. Our local community college is one such case. That's because they jumped on the Internet bandwagon early when public addresses were being handed out for the asking and got themselves a Class B public address (which allows them to have 65,534 publically addressed computers). Every PC at all the campuses has a publically routable address. The upside is they don't need to use a proxy/NAT server in order for the students to be able to access the Internet. The downside is that their internal network is basically a part of the Internet. This is a huge security risk if no firewall is in place because every system is accessible from anywhere in the world (which is why they have a mega-bucks Cisco PIX firewall appliance).

If you're using a networked Windows system at the office and you want to see what your system's TCP/IP settings are, open up a DOS window and enter the command:

```
ipconfig
```

To view the TCP/IP settings configured for a NIC on a Linux system type in the following command at the shell prompt:

```
ifconfig
```

and look for the "eth0:" entry.

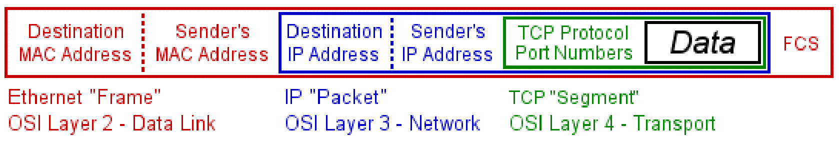
What Network Hardware Does

Now that we've covered the "logical" part of networking, lets take a look at the hardware. When you talk networking hardware you're talking NICs, hubs, switches, routers, cables, etc. However, hubs are quickly disappearing, being replaced by switches. When switches were first introduced, they were significantly more expensive than hubs on a "per port" cost basis, but their prices have dropped to the point where the cost difference isn't an issue.

Why are switches better than hubs? When two PCs want to talk to each other, switches set up a virtual direct connection between two. This means the bandwidth isn't shared with other systems. The two PCs get it all (100 Mbps). They accomplish this by using the MAC addresses of systems on the network.

Each system on a network actually has two addresses. A "logical" IP address that can be assigned and changed by you at will, and a "physical" hardware address that is permanently burned into a NIC. This permanent hardware address is often referred to as a MAC address or "BIA" - Burned-In Address. (Physical, hardware, MAC, and BIA all refer the same address.) Each "packet" sent over a network must include both the logical and physical addresses to identify the destination computer.

Multi-layer Encapsulation

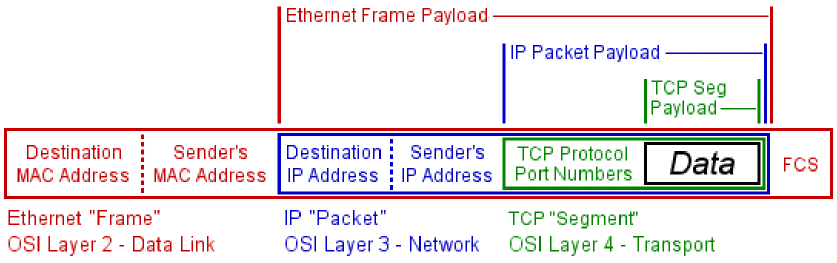


Web2PDF
converted by Web2PDFConvert.com

The above diagram shows how the MAC and IP addresses are used on a LAN. It also illustrates the theory behind how "encapsulation" is used to build a frame and how they apply to the OSI layers. Don't worry about the OSI layer stuff if you're not studying networking. It's presented in just about every network book ever written so I included the references here for those that may be reading up on networking and are encountering this material. (The "FCS" at the end of the frame stands for Frame Check Sequence which is a CRC checksum to make sure the frame didn't get corrupted in transit.) The above also shows why TCP/IP is independent of the networking technology used. When using WAN connections instead of an ethernet LAN, the TCP/IP packet would be encapsulated in a PPP (dial-up, ISDN, or leased-line), frame relay, or X.25 frame. If encapsulated in a frame relay frame, the Layer 2 addressing would use numbers called DLCIs (Data Link Connection Identifiers) instead of MAC addresses.

Technically, using the term "packet" when referring to what's sent across the wire isn't proper. A packet has a very specific meaning in the TCP/IP world. The term "frame" would be more appropriate but it's become rather common to mis-use the term "packet" in this way, such as in "packet sniffers", etc. And actually, nothing is "sent" across a network cable. The voltage on the network cables just fluctuates rapidly and, like the telegraph operators in the old west who knew Morse Code, every device attached to the same LAN cable segment senses and monitors this series of flucuations to see if they pertain to them (as indicated by the destination MAC address). If you're familier with the OSI model, these voltage fluctuations are what's referred to by "Layer 1". It's the NIC's job to convert a frame into a series of voltage fluctuations when sending and to use these voltage flucuations to rebuild a frame when receiving.

Encapsulation Payloads



The encapsulation scheme was devised because it offers flexibility. One layer's packet is the next lower layer's payload. Packets that travel over a LAN are encapsulated in an ethernet frame. If you're connecting to another network using a modem, your computer encapsulates the packets in a PPP frame. The Layer 2 framing is dependent on the connection technology you're using. When Cisco routers are used to connect remote sites they can use any number of different technologies with their related framing protocols including PPP, frame relay, HDLC, etc. Likewise, the Layer 3 protocol could be IPX on a Novell network rather than IP. In such a case the Layer 3 header would contain source and destination IPX addresses (and Novell's SPX protocol would take the place of TCP).

The sending computer takes your data (your e-mail message or Web site request) and does the series of encapsulations based on the type of network you're on. The receiving computer simply reverses the process and de-encapsulates the received frame to extract the data you sent.

One other point to note in the above diagram is that the destination IP address is not the only TCP/IP-related information specified about the destination computer. The TCP port number for the requested service is also specified. HTTP (Web) uses port 80. FTP (file transfer) uses port 21. You may have seen the term "socket" used in TCP/IP literature. A socket is simply referencing the IP address and port number together. For example, traffic destined for a Web server with an IP address of 172.16.5.20 would have a destination socket of 172.16.5.20:80. It's the use of different port numbers (and sockets) that allows you to still surf the Web *while* you're using FTP to download a file.

So how does the sending system find out what a destination computer's MAC address is so that it can put it in a frame? It uses ARP. ARP is an address resolution protocol kind of like DNS. Whereas a computer will send out a DNS query packet to a DNS server to resolve a domain name or system name to an IP address, it will also send out an ARP query packet to resolve an IP address to a MAC address. An ARP request is basically just a broadcast that says "*Whoever has IP address x.x.x.x, send me your MAC address*". We mention this because when comparing hubs to switches to routers you can see where these two different destination addresses come into play. (You'll see how all this ties together in the **How It All Works** section later in this page.)

Network Device	**Packet Handling**	**OSI Layer**
Hubs	Hubs don't look at anything. They serve as a central electrical tie point for all the systems on a network and are basically just repeaters. Any packets received on one port are flooded out all other ports. All systems on the network see all traffic and all share the available bandwidth. As a result, only one computer on the network segment can transmit packets at a time (shared bandwidth).	1 Electrical signals and connections

Web2PDF
converted by Web2PDFConvert.com

Switches	Switches also serve as a central electrical tie point. They don't care about IP addresses. They look at the MAC addresses in packets and build a lookup table that shows which MAC address is plugged into which switch port. This allows them to forward packets MAC-addressed for a specific system to that system only (the virtual direct connection). None of the other systems on the network segment see them. (If it doesn't have the destination MAC address in it's ARP table it floods the packet, either waiting for the destination system to respond so it can add an entry into its ARP table for it, or it lets the default gateway router deal with it.) As a result, multiple computers on the same network segment can transmit at the same time (bandwidth is not shared). Another advantage of switches is that the virtual direct connection they set up between two systems allows for full-duplex communications between the two systems. Full-duplex operation has *the effect* of doubling the bandwidth under certain circumstances because the both systems can send and receive packets to each other simultaneously (provided both NICs support full-duplex operation). There is a *disadvantage* to switches though. You can't hook a system running a "packet sniffer" to a switch to try and analyze your network's level of bandwidth utilization or capture traffic to/from a specific system because the sniffer system can't see all the traffic. It will only see traffic sent to it and any "broadcast" traffic (such as ARP requests) which is flooded out all switch ports. (Knowing the level of broadcast traffic on your network is helpful however.) You can only sniff traffic to/from other systems when they (and the sniffer system) are connected to hubs. However, Cisco switches have a feature called SPAN that lets you configure a port as a SPAN port to hook a sniffer to. You then specify one or more other ports on the switch which will have their traffic duplicated to the SPAN port. You would specify two ports to monitor the traffic between two systems or all other ports to simulate having your sniffer hooked to a hub. This could slow down a heavily utilized switch though as you would in essence be doubling the amount of traffic across the switch fabric because all unicast traffic is going to two ports instead of one.	2 Hardware addressing (MAC address)
Routers	Routers do switching too, building a lookup table of the MAC addresses of systems on the network segment that connects to each interface, but they also look at IP addresses. If the destination IP address of a packet isn't on the same network segment as the system that sent the packet, the router looks at a different table (routing table) to see which other interface it needs to send the packet out of to reach the destination network. That's why default gateway devices are routers. Often times a router will connect to two different types of networks. One interface will connect to an ethernet LAN while another connects to a frame relay WAN link. If the router receives a packet on the LAN interface that needs to be forwarded onto the frame relay link, it will strip off the ethernet frame and re-encapsulate the packet in a frame relay frame before sending it on its way.	3 Logical addressing (IP address)

Switches and routers aren't the only things that build ARP tables. Computers build them too, but the table entries are temporary (called the "ARP cache"), usually being removed after a couple minutes. On a Windows system, open up a DOS window. Linux systems just do the following at the shell prompt. Ping another system on your network. When the ping is finished (on Linux systems press Ctrl-C to stop the pinging), type in the following command to view the ARP table entry created as a result of the ping:

```
arp -a
```

Given that most broadband Internet connections are only in the 1 Megabit-per-second (Mbps) range, you probably wouldn't notice much of a performance difference between using a hub and a switch on a home network. In this case you could save some money by buying a used 8-port 10 or 100 Mbps hub on places like eBay rather than a new switch. However, on larger business LANs the performance difference could be significant.

Notice one very important point. An ARP is a broadcast packet. As such, only systems on the same LAN segment will see the broadcast and respond. In other words, ARP will only resolve the MAC addresses of destination systems on the same Layer 2 segment (which is comprised of systems interconected by hubs or switches).

How does your system know when to try and get the destination system's MAC address? Easy. It looks at the destination IP address. If the destination IP address isn't on the same subnet (doesn't have the same **network** portion of the IP address), then it knows sending an ARP request with the destination's IP address won't do any good. So instead, it sends out an ARP request with the IP address of the **default gateway**. Your system knows that to access any system on a different network (or subnet), it has to send the packet out the default gateway router so it ARPs to get the MAC address of that router's interface. (That's why you enter an IP address, subnet mask, and default gateway address in your system's TCP/IP configuration.)

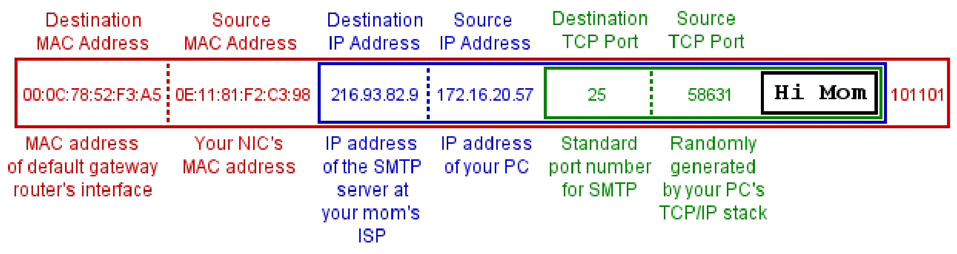
Let's take a look at a somewhat realistic frame. You're on your PC at work which is connected to an Ethernet LAN and you send your mom an e-mail. Remember, when *sending* information the encapsulation process starts at the top OSI layer (Application layer) and works it's way down through the transport, network, and link layers. (Received frames go up the OSI layers and get de-encapsulated at each layer.)

Web2PDF
converted by Web2PDFConvert.com

- Since e-mail messages are sent using the SMTP protocol, your e-mail message will get wrapped in a TCP segment with port 25 as the destination port.
- Then, since your mom's e-mail address contained the domain name of her ISP, your system has to send out a DNS request to get the IP address of the mail server for that domain name. Once it has that it will use it as the destination IP address as it wraps the TCP segment up in an IP packet.
- Because the source and destination IP addresses are for different networks, your system will send out an ARP request to get the MAC address for the default gateway router's interface. With that it can wrap the IP packet inside an ethernet frame, calculate the checksum value for the FCS (Frame Check Sequence), and pass it down to the hardware layer to be put on the wire.

Note that your system had to do two address resolution queries (DNS and ARP) to get the two destination addresses (IP and MAC). Only after it has both of them can it build the frame.

Outgoing E-mail Frame



At some point the packet will likely encounter your proxy server and it will change the source IP address of the packet to the public address of the external interface of the proxy server. The proxy server will use the randomly generated source port number (58631) to keep track of your PC's SMTP "conversation" with the mail server.

As your e-mail packet travels over the Internet the frame type will change to whatever medium (frame relay, ATM, etc.) is encountered along the way (the intermediate hop routers de-encapsulate and re-encapsulate with the proper frame type as necessary because routers can connect to different network mediums). But once the proxy server has changed the private IP address to a public one and sent the frame on its way, the IP addresses in the packet (and any information in the layers above it) never change as it winds its way through the Internet. Only the Layer 2 framing changes.

> **Note:** With point-to-point serial links like a dial-up connection between two modems (PPP Layer 2 protocol) or a leased line between two routers (HDLC Layer 2 protocol) there is no need for Layer 2 addressing. As an analogy, take the old George Carlin line "When there's only two people on an elevator and one of them farts everybody knows who did it". If there's only two devices connected to a link and one of them is sending they both know who needs to be receiving. The packet is still encapsulated in a frame appropriate for the medium, but there is no addressing information in the frame header.

Going back to the OSI model for those studying networking, you can see how the pieces of the above frame relate to the lower layers of the model. It also shows how all the different technologies relate to each other.

OSI Layer	**Function**	**Example Technologies**
1 Physical	Hardware interconnection of devices specifying cable types, connector types and pin-outs, and encoding/modulation methods. (*Electronic voltage levels.*)	10/100-Base-T (for Ethernet), Token Ring (hardware specs), LocalTalk (for AppleTalk), 232c and v.35 (for serial connections)
2 Link	Local network (same segment) addressing (*commonly MAC addresses*). Used by NICs, switches, and routers.	Ethernet (802.2/802.3), Token Ring (802.5), Frame Relay, ATM, Serial links using PPP, HDLC, etc.

Web2PDF
converted by Web2PDFConvert.com

3 Network	Internetwork (different subnet or network) addressing (*commonly IP addresses*). Used by routers.	IP, IPX (Novell), AppleTalk
4 Transport	Service identification (WWW, e-mail, FTP, etc.) commonly by *port numbers*. Used by the protocol stack.	TCP and UDP (using port numbers), SPX (Novell), AppleTalk

Note that a NIC is both a Layer 1 device (because it has a 10/100-Base-T connector) and a Layer 2 device (because it looks at the MAC addresses of the frames it sees). Same for switches because they have 10/100-Base-T connectors and they look at the MAC address to build their MAC address-to-port lookup table). Routers can have multiple interfaces of varying types (serial, 10/100-Base-T, Token Ring) so it is a Layer 1 device (different connector types), has to appropriately re-frame a packet for each of those network types and apply the appropriate Layer 2 address (so it is also a Layer 2 device) and it looks at the IP addess to see which interface the packet needs to go out of (so it's also a Layer 3 device).

Also note that the Layer 3 Network technologies (IP, IPX, etc.) are totally independent of the Layer 2 Link technologies which is why you can have a TCP/IP network span different network types or run Novell IPX/SPX on an Ethernet or Token Ring network.

There's one important point about Layer 4. Note that it's the server **applications** that open ports. A common security practice is to run a "port scanner" program on a workstation and point it at a server to see what ports are open on that server (to see which ports the server is listening on). If ports are open that shouldn't be, it's because some application or memory-resident service opened the port and is listening for service requests on it. To close ports you need to find out which application or service has them open and shut them down as well as keep them from starting when the server is booted. As you will see in the Internet Servers page, applications are often started at bootup through **inetd**.

The above was a simplified coverage of frames and packets. There's more than just addresses in frame (Layer 2), packet (Layer 3), and segment (Layer 4) headers. There are flags, status indicators, sequence numbers, etc. And we didn't show the information in the layers between the Transport layer (TCP ports) and the Application layer (the data). For example, NetBIOS and client/server technologies like SQL would use the "Session" layer above the Transport layer.

Time spent learning the OSI model is an investment that will likely pay off in big time savings when troubleshooting network problems. The layers allow you to narrow down and isolate the possible sources of problems. As a simple example, take a Web server that isn't responding to requests from browsers. If you can ping the server, which uses Layer 3 ICMP packets, you know the network configuration is OK but for some reason the Apache Web server software isn't listening on port 80 (Layer 4). Likewise if two Windows systems running NetBIOS over TCP/IP can't "see" each other in Network Neighborhood but they can ping each other you know the TCP/IP properties are OK and it's likely a Session layer problem. Session layer problems could be something with the Windows Networking configuration (ex: File and Print Sharing isn't enabled or the Server service isn't running - neither of which is needed for TCP/IP at layers 3 and 4 to operate properly).

Web2PDF
converted by Web2PDFConvert.com

The Two Parts of `Ping`

Be careful about your assumptions when using the `ping` command. It has two parts The ECHO (what you send) and the ECHO REPLY (what the remote system sends back).

It is easy to assume that if you don't get a response to a ping there is no connectivity between the two systems. This is not necessarily true. The ping ECHOs may very well be reaching the remote system. However, if the remote system isn't configured correctly, you may not receive the ECHO REPLYs.

The most common reason for this is that the remote system doesn't have a route in its routing table to the network your system (the ECHO sending system) is on. As a result, the remote system is trying to reply but either it doesn't know where to send the replies, or it is sending them but out of the wrong interface (which can happen if the replying system has multiple NICs, but remember that your system sees a connected modem as the `ppp0` interface, essentially another NIC).

Causes for this are incorrect values for the default gateway, subnet mask, or IP address on the remote system (i.e. an incorrect Layer 3 configuration). Also consider the fact that the incorrect values could be on the ECHO sending system. In this case, while there may be connectivity between the two systems, the ECHOs are never being sent or are being sent out of the wrong interface. Although Layer 3 may be configured incorrectly, there is still may be traffic coming from the system. Look at the port indicators on your hub or switch (for ethernet interfaces) when you ping to check for the existence of Layer 1 signals indicating ping traffic. Likewise, the indicators on an external modem can be used to see if ping attempts are being sent or received at the lower layers with dial-up connections.

If you're trying to ping a system on a remote network it's also possible that a router or firewall somewhere between the two networks is blocking one or both of the ping components or the router/firewall itself has an incomplete routing table.

Don't get hung up on the word "protocol" that you see mentioned a lot in networking. It's just a technical term for "a set of rules or commands". The PPP protocol dictates that if you're going to put a PPP frame on the wire the frame has to be comprised of these fields in this order with these lengths and each field has to contain this information. The SMTP protocol dictates that if you're going use the SMTP service on a mail server to send a mail message you have to use these commands with these parameters in this order to set up the connection, specify the sender, specify the recipient, etc. Defining a protocol is just a means of establishing a standard. If everyone builds their hardware or writes their software to follow the same rules then all the pieces work with each other.

Debian comes with two great "sniffer" packages that allow you to look at individual packets. **tcpdump** is a console program that will display packet traffic in a real-time manner, with the packets scrolling up the screen as they are received and displayed. **Wireshark** is a sophisticated GUI protocol analyzer that breaks the frames down into their individual OSI layers and displays the data contained in each of the frame, packet, and segment fields. I use a dual-boot notebook at work with Fluke's *expensive* (read 'thousands of dollars') "Protocol Expert" sniffer installed on the Windows 2000 partition and Wireshark installed on the Debian partition and the differences in the information provided by both is very small indeed. (A Windows version of Wireshark is also available.)

Where to learn more - *The best of our bookshelves:*

More info...

If you're interested in really learning how networks work and the TCP/IP protocol, Cisco's **First Year Companion Guide** is THE book for you. It's one of my all-time favorite networking books. It really makes the light bulbs come on. It's actually the first-year text book for the Cisco Networking Academy program but the first half of the book (first semester of the program which is Chapters 1 through 15 - in the book) deals entirely with the "basics" of networking. It has one of the most thorough presentations of the OSI model I have ever seen in any book, and understanding the functions of the various layers in the OSI model is understanding how networks work. TCP/IP, address classes, encapsulation, the functions of switches and routers, and how to subnet a network are all covered. They even get into security, network management, and home networking later in the book (along with some basics on router programming so you can see how that's accomplished). A strong foundation in the OSI model is essential if you're going to be good at network troubleshooting and this book will get you there. The book gets some bad reviews on Amazon due to some typos and misplaced diagrams. But the fact that you can easily identify a typo or misplaced diagram indicates you understand the material. (Be sure to get the newer 2nd Edition which fixes many of these mistakes.)

And as long as we're on the subject of cool Debian packages and verifying connectivity, one package that's probably already on your system as part of the default installation is the **mtr** utility. It's like the `traceroute` command, only better. If your Debian system has Internet connectivity and you have the correct DNS server settings in the `resolv.conf` file, type in the following at the shell prompt:

```
mtr www.debian.org
```

Web2PDF
converted by Web2PDFConvert.com

It's a great tool for real-time monitoring the effects of any changes you make to server, proxy, firewall, router, etc. configurations. Press Ctrl-C to quit the program.

Sometimes you'll make a change and then you don't get the results you expected. It may be because systems cache certain "reachability" information. We said earlier that **arp** maps IP addresses to MAC addresses (layer 2). Systems also maintain routing tables which relate an interface to the path needed to get to a given IP address (layer 3). Often times you will need to clear these caches to get the results you expect.

Function	Linux Command	Windows (DOS) Command
View ARP cache	`arp`	`arp -a`
Clear ARP cache	`arp -d` *`entry`*	`arp -d`
View routing table	`route`	`route print`
Clear routing table	`route del` *`entry`*	`route -f`
View DNS cache	`n/a`	`ipconfig /displaydns`
Clear DNS cache	`n/a`	`ipconfig /flushdns`

Note that there are no Linux versions of the DNS commands. Windows caches a lot more things to try and improve its performance so if you're using a Windows system to test DNS changes be sure to clear that cache also. Also note that you can't flush all entries from the arp cache or routing table with Linux. It has to be done one entry at a time.

What A Subnet Mask Really Does

So now that you know about ARP and MAC addresses lets go back to subnet masks for a minute. We know that a subnet mask identifies the network portion of an IP address, and that you have to enter a subnet mask on every system on an internal network and it has to be the same on all those systems. But how does a system make use of it? It has something to do with the **default gateway** address you also enter on most systems.

When a system wants to send a packet to another system, it takes the IP address of the destination system and compares it to the subnet mask. If the network portion of the address is the same it knows the destination system is on the same network (segment) it is. So it sends out an ARP request which in effect says "System with IP address *x.x.x.x* send me your MAC address." When it gets the reply it puts the destination system's MAC address in the frame header and puts it on the wire.

If the system compares the destination IP address to the subnet mask and finds that the destination system is on a different network it still sends out an ARP request, but this time the ARP request contains the IP address you entered for a **default gateway** (typically a router). When the router responds to the sending system, the sending system puts the routers MAC address in the frame header (Layer 2) and sends the packet to it and it lets the router worry about the Layer 3 stuff. As a result, you can see that the only Layer 3 function performed by end systems is to find out if a destination system is on the same network (subnet) or not using the subnet mask (and getting the end system's IP address in the first place). If it's not, it sends it to the default gateway and lets it worry about getting the packet to the different network. (But how does the sending system get the destination system's IP address in the first place? That's why you also have to enter DNS server addresses on every system. You'll see how they use these addresses on the DNS Services page.)

Given all of the above you can figure out what problems you'll have if you have an incorrect subnet mask on a system. Lets use the Class B network 172.18.0.0 as an example. Recall that the correct subnet mask for a Class B network is 255.255.0.0 so all systems on the same network segment have addresses that start with '172.18'.

If a system on this network has an IP address of 172.18.5.10 and an incorrect subnet mask of 255.0.0.0 it'll think that any system where the first octet is 172 is on the same segment it is. So while a packet destined for the system 172.16.2.12 is on a different segment, the system with the incorrect mask will think it's on the same segment and will never try to use the default gateway. It'll just sit there and ARP its brains out trying to get a response from 172.16.2.12 with its MAC address. Eventually it'll error out and say that the system is unreachable.

Going the other way, if the Class B system (172.18.5.10) has a Class C mask of 255.255.255.0, it'll think that only systems with addresses that start with '172.18.5' are on its local segment. So if it has a packet destined for 172.18.3.33 (which is on the same segment) it'll think it's on a different segment and send an ARP request with the IP address of the default gateway router and send the packet to it, even though it should have just sent an ARP request using the IP address of the destination system itself. In this instance, the router *may* be smart enough to just send the packet back onto the same segment to the destination system or it may just drop it. It depends on how the router configured. And if the router is smart enough you may not notice there's a problem. But having a "narrower" subnet mask than you should would place an undue load on your gateway router and unnecessary traffic on your network.

Web2PDF
converted by Web2PDFConvert.com

Theory Into Practice

How does knowing all this theory benefit you? Believe it or not, knowing the OSI layers is very helpful in troubleshooting network and computer (both PCs and servers) problems. Someone comes to you and says "The Web server is down." What do you do? Well, given that almost all networks use TCP/IP these days, and given that IP is layer 3 and TCP is layer 4, those are the two layers you want to look at initially.

At layer 3, can you ping the Web server? If you can't ping it either a connectivity link between you and the server is down *or* the server's OS is not functioning properly because its IP stack is not loaded. If you can ping it you know that you have layer 3 connectivity between you and the Web server *and* that the server's operating system is operating because the IP stack is loaded and functioning so it's on to Layer 4.

Layer 4 is TCP ports. At this layer the Web server *application* itself (Apache or IIS) opens port 80. Is port 80 open? Because you have layer 3 connectivity, run a test to see if you can connect to port 80 using telnet. Assuming the IP address of your Web server is 192.168.1.10, at a shell prompt (or on a Windows machine open a DOS window a.k.a. command prompt) and type in:

```
telnet 192.168.1.10 80
```

If your screen (or DOS window) goes blank when you press Enter that's a good thing. It means you connected to the port and the Web server application is waiting for you to enter an HTTP protocol command. If port 80 isn't open you'll get some sort of "connection failed" message. Since the server OS is OK because you could ping it, a failed connection indicates there's something wrong with the Web server application (Apache or IIS) because it doesn't have the port open.

If you can both ping the server and telnet to port 80 the Web server is up and operational so you'll need to talk to that user to find out what they meant when they said the Web server was "down". It could be that the pages aren't displaying properly or they could be trying to access the Web server by name and they're having DNS issues on their workstation.

You can test the layer 4 functionality of other server applications too. For an e-mail server try to telnet to port 25 (for SMTP) and to port 110 (for POP3). For an FTP server try to telnet to port 21. All e-mail server applications open port 25 because that's the standard for SMTP. All FTP server applications open port 21 because that's the standard. However, not all server applications have standard port numbers, such as database applications. The Microsoft SQL Server database application opens port 1433. The MySQL database application opens port 3306. Don't try to telnet to port 53 to check a DNS server though. DNS servers have **UDP** port 53 open to respond to name resolution queries and you can't telnet to a UDP port. If you can telnet to port 53 on a DNS server that means it has TCP port 53 open for zone transfers (which better be secured).

Note that most server applications can be reconfigured to open a port other than the standard one. For example, some Web servers are configured to use port 8080 instead of 80. Because Web browsers are coded to send requests to port 80, if you want to access a Web server that's using a non-standard port you have to specify the port in the URL like so:

http://192.168.1.10:8080

Using a non-standard port on a Web server is a way to "hide" a Web server because, as we've just shown, you have to know which port number to use in order to access it. So how do you find what ports have been opened by server applications?

One alternative to trying to telnet to standard ports to see if they're open, you can get on the server (Linux or Windows) itself and type in:

```
netstat -an
```

and you'll see what ports are open on the server (i.e. ports in the LISTEN state). Note that this command will list both open TCP and UDP ports so make sure you look at the TCP ports.

Another alternative would be to use a port scanner like **nmap** on a workstation and point it at a server. To scan for open ports on the Web server mentioned above, the nmap command would be:

```
nmap -v -sS -p1-9000 192.168.1.10
```

This will perform a SYN scan on ports 1 through 9000. **nmap** is a free command line penetration testing tool thats available in both Linux and Windows version. It has a ton of scanning options listed in the documentation.

The point of this section is that all this theory isn't just hot air. It's how this stuff works. The better you understand how something works that better you will be able to troubleshoot it.

At The Office

The wide range of addresses available with the 172.x.x.x space allows you to simplify things in larger networks. You pick **one and only one** value for the first 'x' (the second octet) from the available range of 16 through 31, but you can use different values for the

Web2PDF
converted by Web2PDFConvert.com

third octet to try and keep track of things.

For example, lets say you decide to use 172.18.x.x for your network. You could manually assign all of your servers addresses in the 172.18.1.x range. All of your printers could be manually assigned addresses in the 172.18.2.x range, and you could set up a DHCP server on your internal network to automatically assign addresses in the 172.18.3.x and 172.18.4.x ranges to your workstation computers. (In DHCP lingo, the address spaces you set up for the DHCP server to hand out individual addresses from are called "scopes" and the addresses are "leased" to workstations.)

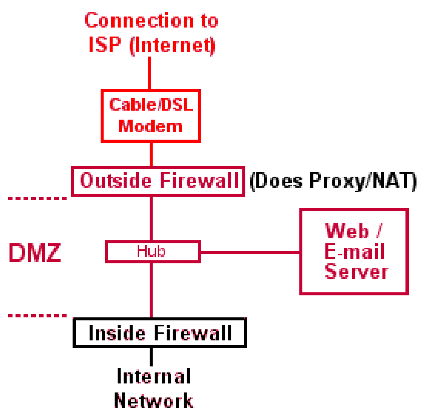
Even though the systems in your network would have a value for the third octet ranging from 1 to 4 in this example, because you'd only be using a Class B subnet mask (255.255.0.0), they would all still be on the same network. You're still using two octets to identify individual computers instead of just one octet as with a Class C address. It's just that you're using the third octet as a means of logically grouping or identifying devices.

> **Note:** Using different numbers in the "computer" portion of address as in the previous example is not the same as the "subnetting" we referred to when discussing the 10.x.x.x network address space earlier. While a subnetted Class B network would likely have different values for the third octet on each subnet, simply assigning certain a certain group of computers the same value for the third octet in a Class B address range does not create a subnet. Subnetting is done with routers to reduce the size of broadcast domains which has the effect of increasing available LAN bandwidth. (Subnetting is covered in more detail in the **Subnetting** section on this page.)

If you want to allow your internal private network to access the Internet you can set up a broadband connection to the Internet. This isn't much different than getting cable or DSL access at home. You would then set up a proxy/NAT server to allow everyone on the internal network (with their non-routable IP addresses) to share this broadband Internet connection. Most ISPs require businesses to get a "business account". However, most business accounts have special features that make them preferable.

For one, check to see if the business-account broadband connection is symmetrical (i.e. the speed of the connection is the same in both directions). Home accounts have an asymmetrical connection where the download speed is much faster than the upload speed. The other important feature is that most business accounts include several static public IP addresses that you are assigned. These two features are necessary if you want to set up your own Web and/or e-mail servers (as you will see on the Internet Servers page). The cable or DSL interfaces for most home accounts are dynamically assigned temporary public IP address. (A comparison of DSL and cable services is given later on this page.)

Keep in mind though, that security becomes a major issue when you start connecting business networks to the Internet. Ideally you'd want to put any Web and/or e-mail servers in a "DMZ" with one firewall between them and the Internet. You'd then put a second firewall between the DMZ and your internal network to further restrict traffic. A DMZ is also referred to as "screened subnet". It's also a good idea to put an IDS (Intrusion Detection System) in your DMZ.



We get more into how to set up a DMZ using Linux systems for the outside and inside firewalls on the Firewall page. We also show you how to set up and test a Snort IDS system on the Security page.

Check out our Network Monitoring and Intrusion Detection pages !

At Home

Setting up a network at home is easy. Hardware-wise you just need to install some 100 mega-bit PCI NICs (Network Interface Cards) in the PCs, buy an 8-port 100 mega-bit switch, and the RJ-45 Cat 5 UTP cables to connect the PC NICs to the switch. (Get an 8-port instead of a 4-port switch. They don't cost that much more and it's always better to have too many network connections than not enough.) For home networks a hub would work just as well as a switch, but not too many companies make hubs anymore because switches have dropped in price so much. Some computers come with NICs integrated into the motherboard. Check the back of your computer for an RJ-45 jack. It looks like a telephone jack but has 8 contacts instead of the 4 contacts with telephone jacks.

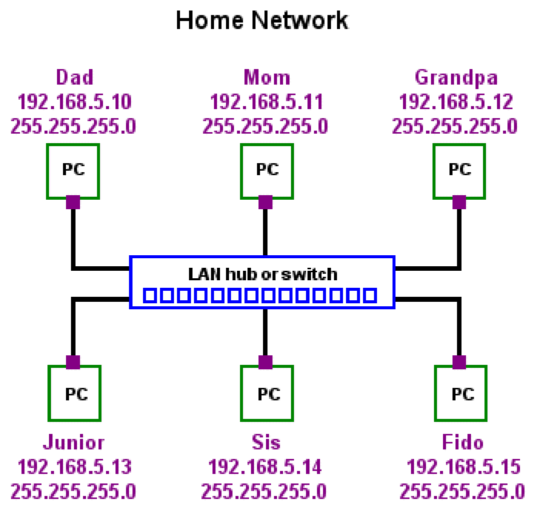
Web2PDF
converted by Web2PDFConvert.com

If some of your systems are scattered or you don't want to run cables to them you could always add a wireless NAP (Network Access Point - which is like a wireless hub that would plug into a port on your switch) to your network but this would also require getting wireless NICs for those PCs. They also make PCMCIA versions of NICs so you can connect your notebook into your home network. Once you've got the hardware in place, you just configure the network settings in the PC software. We'll cover the Debian network configuration below. On Windows PCs, go to

Start/Settings/Control Panels/Networking

and get into the TCP/IP properties for the NIC where you can enter an IP address for the PC, a subnet mask, and possibly a default gateway (if you have a broadband Internet connection). When you enter an IP address into the TCP/IP properties like this it's called a "static" IP address because it won't change unless you change it.

The family in the diagram below chose a Class C private address range of 192.168.5.x



When you set the TCP/IP properties you are assigning the private addresses to the NICs inside the computers. You can have a NIC with a private address assigned to it on your home network and still use a dial-up modem connected to the same system to get on the Internet. Your modem gets a different (public routable) IP address from your ISP every time you dial in and retains it as long as you're connected.

The names above are not just identifying family members. They are the names given to the systems (hostnames). Since a home network is too small for a DNS server, you can set up name resolution using a simple **`hosts`** file. A **`hosts`** file is just a simple text file you can edit using any text editor (like NotePad on Windows systems). Note that this file does not have an extension. On most Linux and UNIX systems the path to open the file is:

`/etc/hosts`

On Windows systems the paths are typically:

```
     95 / 98:   C:\Windows\hosts
   NT / 2000:   C:\WINNT\system32\drivers\etc\hosts
```

Once you open the file in a text editor, add the address/hostname pairs for the systems on your network. For the above home network the **`hosts`** file would look like this:

```
127.0.0.1           localhost
192.168.5.10        dad
192.168.5.11        mom
192.168.5.12        grandpa
192.168.5.13        junior
192.168.5.14        sis
192.168.5.15        fido
```

(That's a Tab character between the address and hostname.) Each system that would want to do name resolution would need to have this **`hosts`** file but once you create one you can just copy it to other systems. Then instead of entering the command:

```
ping 192.168.5.12
```

you could just enter:

```
ping grandpa
```

If Grandpa had a Debian system running the Apache Web server software, Junior could open up a Web browser on his system and

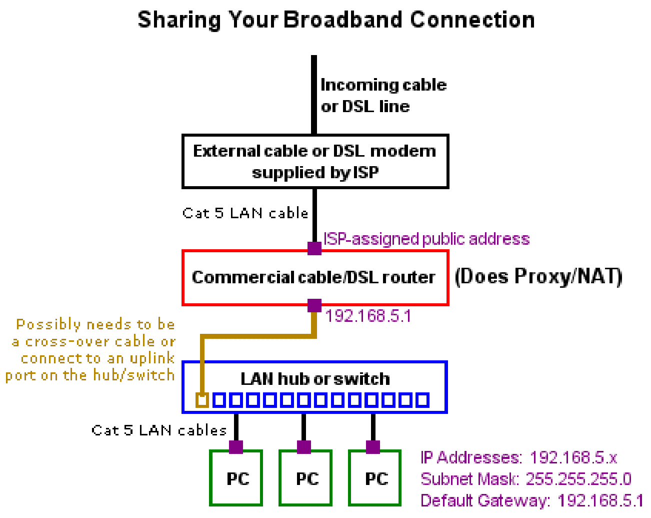
Web2PDF
converted by Web2PDFConvert.com

enter the following URL to see Grandpa's Web pages:

`http://grandpa/`

The downside of `hosts` files is that you have to have one on every system on your network and if a change needs to be made it needs to be made to all of the files manually by editing them with a text editor. As an alternative to setting up a `hosts` file on all the computers you could set up a DNS server for your LAN. This DNS server would also have the ability to resolve Internet host/domain names if you have a broadband connection to the Internet. Actually, you don't need a separate server. You can just run the DNS service on your existing Debian system. We show you how it all works, and how easy it is to set up, on the DNS page.

If you have cable or DSL service at home, you can share this broadband connection with all of the computers on your home network. Cable or DSL providers typically give you the option of having an external cable or DSL modem, which then connects to your computer's NIC via a Cat 5 LAN type cable or connects to your computer's USB port, or an internal model that gets installed in one of your computer's slots. Stay away from anything USB and **get the external model that plugs into a NIC.** That way, instead of plugging the cable into your computer's NIC you can plug it into a device known as a **cable/DSL router**. This cable/DSL router would then plug into the hub or switch that you use to connect all of your computers together. (Some cable/DSL routers have a built in 4-port hub.) A cable/DSL router is basically just a "proxy-server-in-a-box".


Sharing Your Broadband Connection
Incoming cable or DSL line
External cable or DSL modem supplied by ISP
Cat 5 LAN cable
ISP-assigned public address
Commercial cable/DSL router
(Does Proxy/NAT)
192.168.5.1
Possibly needs to be a cross-over cable or connect to an uplink port on the hub/switch
LAN hub or switch
Cat 5 LAN cables
PC
PC
PC
IP Addresses: 192.168.5.x
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.5.1

The "modem" you get from your cable or DSL provider is actually just a device called a "bridge". It bridges traffic from one type of network (cable or telephone) to another (ethernet-based LAN) without doing any packet inspection, filtering, or manipulation. It is essentially a media converter.

> Note that a cable/DSL router is **not** the same as the cable or DSL "modem" or "interface" that you get from your cable company or ISP. The router allows you to take the incoming Internet connection, which normally goes to a single computer, and plug it into a hub or switch so that the Internet connection can be shared by your entire internal network. Some can even be set up to act as DHCP servers to hand out private IP addresses to your home PCs each time you turn them on. If you don't want to assign static addresses to the PCs on your home network you just set them up to use a DHCP server and then enable the DHCP function on the cable/DSL router.

Your cable or DSL service provider assigns you a single public (routable) IP address. If you don't have a cable/DSL router, and because of the bridging action of the **modem** they give you, it is your **PC** which receives this public, routable IP address. As a result, only one PC can access the Internet. A cable/DSL **router** performs the NAT function explained earlier which allows this single public address (which is assigned to the router rather than a single PC) to be shared by multiple systems on your LAN.

Note in the above diagram that if you use a commercial cable/DSL router you *may* need one of two things (it depends on the model you buy). The cable that connects the router to your hub or switch may have to be a Cat 5 "crossover" cable, or you may have to use a regular Cat 5 LAN cable and plug it into the "uplink" port found on most hubs/switches. If you try using a regular cable but the "link" indicator on your hub/switch don't light up for that connection, you'll likely have to do either of the above. (If your hub or switch doesn't have an uplink port, you'll have no choice but to get an uplink cable.) None of this applies if you are using a Linux system as a proxy server.

IMPORTANT: Because Windows enables file and print sharing by default, having a publically routable IP address assigned to your Windows PC is a security risk. The file and print sharing functions open ports on your system and the public address makes your system accessible from anywhere on the Internet. So even if you don't need to share your broadband Internet connection with other PCs (i.e. the cable or DSL modem now plugs directly into your PC), you should either disable file and print sharing or have some kind of NAT or

Web2PDF
converted by Web2PDFConvert.com

firewall system (cable/DSL router or Linux system) between your PC and the modem to protect your PC. (You could disable these sharing functions but this would negate the point of having a home network in the first place - to share files, printers, etc.)

In the above diagram, the public IP address dynamically assigned by your ISP is on the router's "external interface" (connection), and you assign a static private IP address to the "internal interface". Note that the "default gateway" setting on all of your home network PCs is this same static private address you assigned to the internal interface of the router.

The default gateway setting on a PC is kind of a "last resort" setting. (Cisco routers actually refer to the default gateway setting as the "gateway of last resort".) It basically tells the computer that if it wants to contact another computer, and it can't find that other computer on the local network (which Internet servers won't be), it should send the traffic for this other computer to the address of the default gateway. The default gateway device is typcially a router, so it will know what to do with any traffic that's destined for a distant network (which is why they call it a "gateway"). So in addition to doing address translation, a commerical "proxy-server-in-a-box" or Linux proxy server is also a router which acts as a gateway.

Cable/DSL routers are available from manufacturers such as Linksys and DLink for under $100. Here's an example list using mostly Linksys equipment of what you'd need to set up a home network where you can share a broadband Internet connection. (You may want to spend a little more and get 3Com 3C905 NICs because they have wide driver support.):

- An 8-port 10/100-mb Switch * - approx $40
- A Cable/DSL Router for Internet access sharing * - approx $50
- One 100-mb NIC for each computer - (get used 3Com 3C905C NICs on eBay)
- One PCMCIA NIC for each notebook - approx $40 ea
- One Cat 5 LAN cable for each computer/notebook (variable lengths) - approx $15 ea

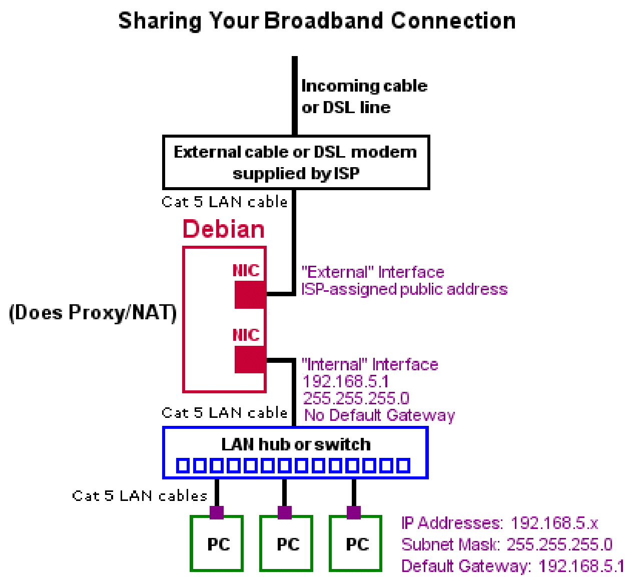
* You could replace the switch and router with an integrated model but that limits your options for cable runs, experimenting, and troubleshooting.

For adding wireless nodes:

- A Wireless NAP/Bridge - approx $50
- One Wireless NIC for each computer - approx $50 ea
- One Wireless PCMCIA NIC for each notebook - approx $50 ea

So to network (wired) four desktop PCs at home without a broadband Internet connection you're looking at approximately $175 (switch, NICs, and cables). Around $250 (additional cable and router) if you do have a broadband connection to share. This can always be added on as a separate piece later if you don't have a broadband connection now.

Alternative: Instead of buying a commerical cable/DSL router product you could just use a Linux box to act like one. Since a cable/DSL router is nothing but a proxy-server-in-a-box, you could set up a Linux system to be a proxy server and it's easy to do. (You'll see how easy on the Proxy/NAT page.) The advantage of using a Linux system as your cable/DSL router is that you can customize the firewalling capabilities of the router using something called IPTABLES. This not only includes customizing what traffic you allow into your network, but also allows you to restrict which systems on your network can access certain services or sites on the Internet. (You'll see how to do this on the Firewall page.)



Sharing Your Broadband Connection
Incoming cable or DSL line
External cable or DSL modem supplied by ISP
Cat 5 LAN cable
Debian
NIC
"External" Interface
ISP-assigned public address
(Does Proxy/NAT)
NIC
"Internal" Interface
192.168.5.1
255.255.255.0
No Default Gateway
Cat 5 LAN cable
LAN hub or switch
Cat 5 LAN cables
PC
PC
PC
IP Addresses: 192.168.5.x
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.5.1

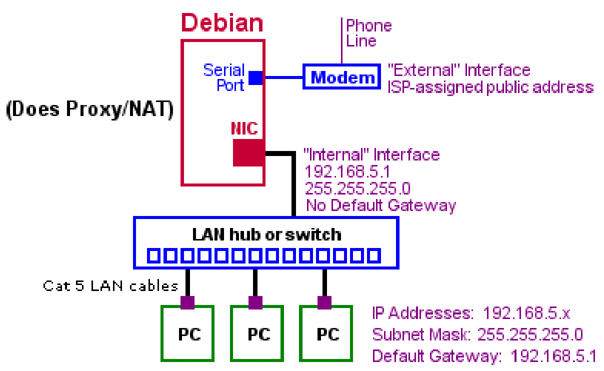
Another advantage of using a Linux server over a commercial cable/DSL router is that you could also use it as **your own home**

Web2PDF
converted by Web2PDFConvert.com

Web and e-mail server using the dyndns.org dynamic DNS service. We cover this on the DNS Services page. And since you've got a Web server running, you can easily **add a Web cam** to it so you can keep an eye on your home from anywhere that has a Web browser. We show you how to add that on the Web Cam Server page.

The concept of sharing an Internet connection is the same no matter what type of Internet connection you have. With a Linux system acting as a proxy server or firewall, you can just as easily share a modem connection.

Sharing a Modem Connection



This is essentially the same type of configuration you have if you use the "Internet Connection Sharing" option found on Windows systems. See the Modems page for information on getting a modem to work on your Debian system. Then see the Proxy/NAT page for a script that you can use to get your Debian system to act as a proxy server.

One important point to keep in mind when setting up a gateway type of system (which includes proxy servers and firewall systems) is that the **"internal" interface should NOT have a Default Gateway entry.** Leave it blank. The Default Gateway entry for the "external" interface will either be automatically assigned when you connect to your ISP or should otherwise be available from them.

So the proxy server that sits between your private network and the Interent actually has (or should have) three functions. A **gateway** (router), an **address translator**, and a **firewall**. The firewalling capabilities of the commercial cable/DSL routers are limited at best and typically not upgradable if a new sort of attack is developed. We'll show you how to set your Debian system up to protect your network on the Firewall page.

> **Note:** When we use IP addresses in our examples on these guide pages, they will be in the private IP address ranges even if it's an example where a public address would normally be needed (such as with an Internet-connected interface). We use addresses like this so we're not using IP addresses that have been legally licensed to companies or ISPs.

If you also want your Debian system to act as a DHCP server handing out addresses to the PCs on your home network when they boot up, you can easily accomplish this. However, we won't cover it here. Based on the information we've given so far on searching for and installing packages, this would be a good one for you to try on your own. If you do set it up, be sure to set the PCs on your network to use a DHCP server. In Windows, this is modifying the TCP/IP properties for the LAN interface to "Obtain an IP address automatically".

If you are going to connect to your ISP's cable or DSL modem to the "external" interface (NIC) on your Debian system, you'll likely need to run some sort of DHCP *client* software on the system so that it can pull the ISP-assigned address and apply it to the external NIC. This is because it's not the cable or DSL modem that gets assigned an IP address. A cable or DSL modem merely acts as a bridge. Whatever device is connected to the customer side of the modem (PC, router, etc.) is what actually gets assigned an IP address. We likewise won't cover installing a bootp or DHCP client here. See the Web HOWTO documents for cable or DSL modems. (The reason that no DHCP client software is necessary if you're using a dial-up modem connection is because the dynamic addressing is handled by the PPP protocol.) See the Web HOWTO documents for cable or DSL modems.

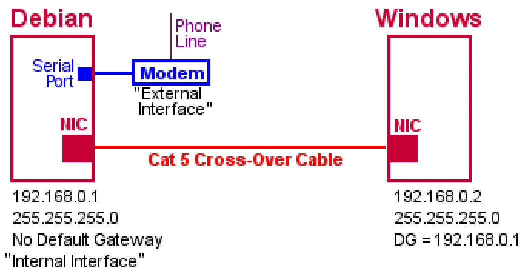
A No-Network Network

On the Proxy/NAT and Firewall pages we'll show you how to configure your Debian system so you can share your broadband connection with all of the computers on a network, including a home network. However, you don't need a formal network or even a broadband connection if you want to try setting up the proxy and firewall functions demonstrated on these pages.

For those that don't have a home network or access to a business network you can still try the examples we present on these pages by setting up a two-system network (your Debian system and a Windows PC for example). All it requires you to do is to have a NIC installed and configured in each computer, and to purchase a special cable called a "Cat 5 cross-over" cable. (As mentioned above, your computers may have network cards inegrated into the motherboard already. Check the back of the systems for an RJ-45 jack.) If one of the computers you use is a notebook, and it doesn't have an integrated NIC, you'll need to get a 100 mb PCMCIA card. You can get those used for about $20 on eBay or similar sites.

converted by Web2PDFConvert.com

A cross-over cable flips the transmit and receive pairs of the cable so they are on different pins on the connectors on each end. This allows you to connect the two systems in a back-to-back fashion so that you don't need an ethernet hub or switch. Note that you would have to have had the NIC installed in the system and installed the appropriate NIC driver module during the Debian installation to be able to use the NIC.

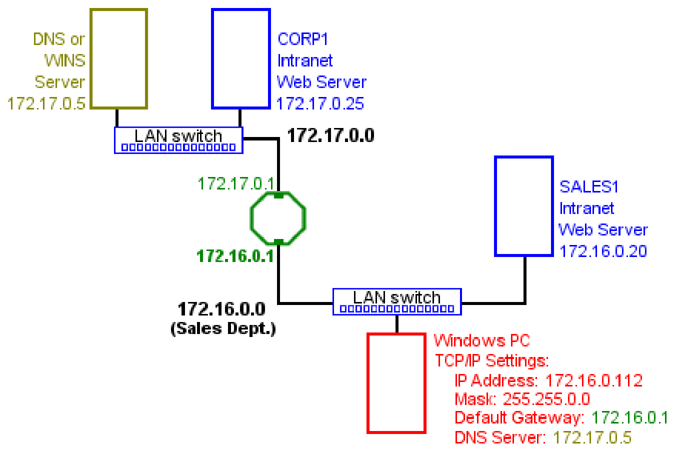


The back-to-back connection between the two systems represents the internal LAN where each system would need a unique IP address. You would then hook your modem up to the Debian system to set up the Internet connection (as outlined on the Modems page). And just as shown in the above diagram, the IP address of the NIC on the Debian system would be the default gateway entry on the Windows system. This is because the Debian system will be acting as the gateway for this two-system network.

This configuration will also work if you want to play around with the servers we set up on the Internet Servers page.

How It All Works

So you've got IP addresses and MAC addresses and subnet masks and default gateways and DNS and ARP and hardware and software. How do all these pieces fit together? Consider the network below.



The user of the Windows PC in the Sales department fires up their Web browser so that they can access the company's internal intranet Web servers. There's a corporate intranet Web server with company-wide information and one right in the user's department on the same network segment with information for Sales staff.

Remember that each system has **two** addresses (logical and physical) and in order for one system to communicate with another, it needs to include **both** of these addresses when it puts a packet together to send to the destination system.

Scenario 1 - Accessing SALES1 - SAME Network

The Windows PC user wants to look at sales figures for last month on the intranet Web server in their department. In the browser URL line they enter:

http://sales1.widgets.com

or in the case of a Windows-based network that is only using WINS for name resolution:

http://sales1

Here's what happens:

1. The Windows PC sends a DNS or WINS query to the DNS or WINS server to get the IP address of the intranet Web server named in the URL (in this case SALES1)
2. The DNS or WINS server sends a response back with the IP address of 172.16.0.20

Web2PDF
converted by Web2PDFConvert.com

3. The Windows PC uses its subnet mask to see if the received IP address is on the same logical network as it is (i.e. is the network portion of the IP addresses of both SALES1 and the Windows PC the same)
4. Because they are on the same logical network, the Windows PC sends out an ARP request using the SALES1 IP address
5. SALES1 responds to the Windows PC supplying the Windows PC with its (SALES1) MAC address so the Windows PC now has both of the destination addresses it needs to send the HTTP packets to SALES1
6. The Windows PC sends out the HTTP packets to SALES1. The switch, knowing the MAC addresses of both systems, sets up a virtual direct connection between the two and the HTTP traffic is sent

Scenario 2 - Accessing CORP1 - DIFFERENT Networks

The Windows PC user in the Sales department wants to look at the company's telephone directory on the main corporate intranet Web server. In the browser URL line they enter:

http://corp1.widgets.com

or in the case of a Windows-based network that is only using WINS for name resolution:

http://corp1

Here's what happens:

1. The Windows PC sends a DNS or WINS query to the DNS or WINS server to get the IP address of the intranet Web server named in the URL (in this case CORP1)
2. The DNS or WINS server sends a response back with the IP address of 172.17.0.25
3. The Windows PC uses its subnet mask to see if the received IP address is on the same logical network as it is (i.e. is the network portion of the IP addresses of both CORP1 and the Windows PC the same)
4. Because they are **not** on the same logical network, the Windows PC sends out an ARP request using the **default gateway's IP address**
5. The default gateway (router) responds to the Windows PC supplying the Windows PC with its (router interface) MAC address so the Windows PC now has both of the destination addresses it needs to send the HTTP packets to CORP1
6. The Windows PC sends out the HTTP packets to the default gateway and the switch, knowing the MAC addresses of the Windows PC and the default gateway router interface, sets up a virtual direct connection between the two and the HTTP traffic is sent
7. The default gateway router removes the Ethernet frame (de-encapsulation) and looks at the network portion of the IP address in the incoming packets to determine what network the destination computer is on
8. The default gateway router checks its routing table and finds that the destination network is connected to one of its interfaces (the 172.17.0.1 interface)
9. The default gateway router sends an ARP request out of this interface with the IP address of the destination computer (CORP1)
10. CORP1 responds to the router with supplying it (router) with its (CORP1) MAC address so the router now has both of the destination addresses it needs to forward the packets onto CORP1
11. The default gateway router re-encapsulates the packet with a new Ethernet frame using CORP1's destination MAC address and the source MAC address of its 172.17.0.1 interface and forwards the packets to CORP1. The switch, knowing the MAC addresses of the router interface and CORP1, sets up a virtual direct connection between the two and the HTTP traffic is forwarded

In both of the above scenarios, the intranet Web servers will be the ones doing all the DNSing and ARPing when they wish to respond to the received HTTP request. (Recall that on small networks having a `hosts` file on each system could be used instead of a DNS or WINS server in which case the DNS or WINS queries wouldn't be necessary.) Also, in actuality a PC or router will check it's local ARP table first to see if it already has an entry for the MAC address of the destination system before sending out an ARP request. It it does need to issue an ARP request it uses the received MAC address to update its table.

Note one important point. As packets travel over router hops, the MAC addresses change (as the routers re-encapsulate the packets with new frames). However, the source and destination IP addresses stay the same. MAC addresses (and ARP) are used **within** networks (or sub-networks as in the case above). IP addresses are used both within and across sub-networks and networks.

In order to fully appreciate the above material you have to be familier with the OSI layer model and what takes place at each layer. However, if you understand the above material you've got a pretty good handle on how networks work.

Network Configuration Files

If you set up networking during the Debian OS installation, these files will be pretty much set up already. If you wanted to change anything related to the network settings on your system, you simply make the relevant changes in the appropriate files (just make sure you make all of the necessary changes in all of the files so the networking configuration is consistent). As with most Linux/UNIX configuration files, these are all text files that can be changed using a simple text editor.

Where necessary, changes to these files are detailed on the Internet Servers page in the configuration of various server services.

converted by Web2PDFConvert.com

The primary network configuration files on a Debian system are:

- `/etc/hosts`
  This file is used for name resolution of machines on the **local** network. Just as DNS is used to resolve Internet domain names (i.e. translate the domain name into an IP address that your computer can use to communicate with a distant computer), the **hosts** file is used to resolve the names of computers on your local network. In addition to having a line for each of the other computers on your internal network, you have to have one for the computer that the hosts file is on (the "local" computer). For example, if you accepted the default host name of "debian" during the installation, the hosts file entry for the local system would be:

  ```
          192.168.10.10     debian debian.mylastname.net     localhost mailhost
  ```

  Note that the host name and the FQDN (Fully Qualified Domain Name) are both listed. The extra space between the IP address and host name, and between the FQDN and the word "localhost" indicate Tab characters. You can also have entries for your Windows systems (provided they're running TCP/IP and not just NetBEUI). You just use the system's "Computer Name" so an example of an entry for a Windows system would be:

  ```
                   192.168.10.20     win95box win95box.mylastname.net
  ```

  You will also see a "localhost" entry with an IP address of 127.0.0.1 which is called the "loopback" address. It has an interface designation of **lo** when you use the **ifconfig** command. The loopback address is a TCP/IP standard found on all systems no matter their OS. It is used for testing. If you can successfully ping the loopback address it indicates that your TCP/IP stack is properly configured. If you can ping the IP address of the NIC that's in the computer you're on (the local system) it indicates that the NIC and TCP/IP stack are both OK.

  Note that with hard-core networking types like Linux/UNIX and Cisco, a "host" is any end-node like a server, a workstation computer, or even a networked printer. So what we refer to as the "computer" part of the IP address on this page is often referred to as the "host portion" of the IP address in networking literature.

- `/etc/host.conf`
  This file is what you use to tell the system what to use to resolve host (computer) and domain names. The most common possible selections are:

  - hosts - look for host names and addresses in the **/etc/hosts** file
  - bind - use the DNS server(s) specified in **/etc/resolv.conf**
  - nis - Network Information System - kind of an internal network version of DNS used on large Linux/UNIX networks

  The order in which the above possible selections are listed is the order in which they are used. For example, you should always list "hosts" first so your system doesn't go out to the Internet trying to find a system that's on your local network. If using the first method doesn't resolve the name, the next entry is tried. Running NIS on your network is no small feat so if you don't know if you should use NIS or not you're probably not running it so you don't need the "nis" entry. As a result, the important line in this file is:

  ```
                          order hosts,bind
  ```

- `/etc/resolv.conf`
  As noted above, this is the file you use to enter your DNS server information. Unless you have your own Internet DNS server, this file will contain information about your ISP.

  ```
  search yourisp.com
  nameserver 172.25.188.66
  nameserver 172.25.188.77
  ```

  Your ISP's domain and DNS server IP addresses would be entered in place of the blue entries above.

  What about having your own Internet DNS server? Bad idea. First of all, when you register a domain name with someone like Network Solutions you're required to enter the IP addresses for two DNS servers. This is because DNS is critical. If it fails no one will be able to "find" (get the IP address of) your Web, e-mail, or other Internet servers. Thats why a proper DNS server setup includes two servers with two different addresses (for redundancy). And since an ISP will typically only allocate a few static IP addresses to you, using two of them just for DNS isn't very efficient. If you need DNS records for an Internet Web or e-mail server, check with your ISP. They will usually host your DNS records on their DNS servers for a small one-time setup fee ($5 in the case of our ISP).

- `/etc/network/interfaces`
  This file contains the IP information that your system uses to work with the NIC(s). There is a parent entry for each NIC, and the information for the NIC is listed underneath it like so:

  ```
  auto eth0
  iface eth0 inet static
  ```

Web2PDF
converted by Web2PDFConvert.com

```
        address 192.168.10.10
        netmask 255.255.255.0
        network 192.168.10.0
        broadcast 192.168.10.255
 gateway 192.168.10.1
```

Note that the above would be appropriate for an internal LAN interface. Also note that the "network" (aka "wire") and broadcast addresses for the network the system is on are also listed.

Be careful about using the `gateway` setting. This should only be used if you truly do have a gateway router that leads off your network, most likely to the Internet. (A proxy server or firewall is one such type of router.) Having a default gateway address in the NIC configuration when you don't have a default gateway router will cause problems if your try and connect to the Internet using a modem. (See the Modems page for more information on this.)

As you will see on the Internet Servers page, the Apache Web server software allows you to host any number of Web sites on a single server by setting up a "virtual host" for each domain (Web site). For example, if you wanted to use your system to host the Web sites `www.shoes4men.com` and `www.shoes4ladies.com` you would set up a virtual host for each in the Apache configuration file. However, in order to use an SSL certificate to conduct secure financial transactions you would need a unique IP address for each of these domains, but at the same time you only have one NIC connected to your broadband connection. No problem. You just create a "virtual interface" for each domain.

You create virtual interfaces by creating an additional parent entry in the `/etc/network/interfaces` file for each virtual interface. Where the above only had one parent entry for the NIC, we create multiple parent entries for each virtual interface append the actual interface designation with a colon (`:`) and the number 1 or higher. For example:

```
auto eth0

iface eth0:1 inet static
        address 172.30.156.115
        netmask 255.255.255.240
        network 172.30.156.112
        broadcast 172.30.156.127
 gateway 172.30.156.1

iface eth0:2 inet static
        address 172.30.156.116
        netmask 255.255.255.240
        network 172.30.156.112
        broadcast 172.30.156.127
 gateway 172.30.156.1
```

Notice the `netmask` entry. This indicates that most of the address is the network address. Only the last 4 bits of the last octet identify the computer. That only allows for 16 computer addresses (actually only 14 are usable due to the wire and broadcast addresses).

```
            172.30.156.116
               N.N.S.S/C
            255.255.255.240
1111 1111.1111 1111.1111 1111.1111 0000
nnnn nnnn.nnnn nnnn.ssss ssss.ssss cccc
```

More precisely, because the IP addresses are in the Class B address range, the first two octets are the **network** portion of the address and the third octet and the first half (4 bits) of the fourth octet are the **subnet** portion of the address. Subnets are created by "borrowing" some of the computer bits in an address and using them to identify sub-networks. Naturally, this leaves you with fewer computer bits so you must have fewer computers on a subnet.

This is something that ISPs routinely do. They will take their public address space and subnet it into a lot of small public subnets. These small public subnets are then assigned to business customers looking for static IP addresses.

The point? Looking at a subnet mask will give you some idea of a networks size. The higher the numbers in the subnet mask:

- the more bits that are used to identify the network/subnet portion of an IP address
- the more networks (subnets) there are in the address space
- the smaller these individual networks will be (fewer number of computers per network)

It's kind of like cutting up a pie. You can get more slices (networks) if you make the individual slices smaller. More details on subnetting are given in the **Subnetting** section below.

- `/etc/network/`**`options`**
  There's only one line of interest to us in this file:

converted by Web2PDFConvert.com

`ip_foward=no`

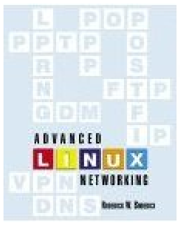
This is the setting that allows (or in this case doesn't allow) your system to act as a gateway when acting as a proxy server or firewall. I point this out because you will see in the shell scripts on the Proxy/NAT and Firewall pages that you can use a command to dynamically change this setting. The actual file isn't edited at all, but there is a way to change this setting in memory (which is lost at the next reboot).

- `/etc/inetd.conf`
  This file allows you to control which services are made available by the server. As a matter of fact, commenting out the lines for some of the services listed in this file is one of the ways you can help to secure an Internet server. When you comment out a line its corresponding service does not start and its associated port is never opened. More details about this file are given on the Internet Servers page.

On Windows systems, when you want to change the network settings you go into the Network control panel and select the TCP/IP Properties. You enter your information into a GUI window and it is then written to the files that make up the registry. With Linux you basically eliminate the GUI middle-man and edit the files yourself.

When it comes to networking, all operating systems do pretty much the same thing. They just put a different face on it.

Where to learn more - *The best of our bookshelves:*


More info...

What the "Debian GNU/Linux Bible" is to the Linux OS, **Advanced Linux Networking** is to Linux servers. It's 26 chapters covers a *wide* variety of servers without getting heavy into the technical details of each specific one. In addition to covering the usual variety of LAN and Internet servers (Web, e-mail, DNS, NFS, print, Samba, etc.) it also covers the less common types of servers including news (NNTP), time, kerberos authentication, and X servers. Setting up a gateway-to-gateway VPN and a VPN access server running PPTP to support Windows clients are also covered. As is typical with type of book, you won't become an expert in any single server application, but you will be able to get a lot of different types of servers up and running quickly so that you can play around with them while pursuing more in-depth technical material.

Replacing A Network Card

NICs fail or you may want to upgrade to a faster NIC. If you're replacing a failed NIC with one of the same type, you shouldn't need to do anything to any of the configuration files. Simply swap the card out and boot the system.

If you're replacing a NIC with a different model, check which driver the new NIC uses. If it's the same driver, again you shouldn't need to do anything except swap the NIC and boot the system. Here are some common NICs and the drivers they use:

NIC	Driver
3C509-B (ISA)	`3c509`
3C905 (PCI)	`3c59x`
SMC 1211 SiS 900 Allied Telesyn AT2550	`rtl8139`
SMC 8432BT SMC EtherPower 10/100 Netgear FX31 Linksys EtherPCI Kingston KNT40T Kingston KNE100TX D-Link DFE500TX D-Link DFE340TX D-Link DE330CT	`tulip`

Many other cards use the `pcnet32` or `lance` drivers. If your NIC is not one of the ones listed above you may find it, and its corresponding driver name, in the Ethernet HOWTO list.

If the NIC you're installing uses a different driver, you only need to manually edit one file. The `/etc/modules` file lists the kernel modules that should be automatically loaded when you boot the system. (NIC drivers are kernel modules.) All you need to do is edit this file using the nano text editor with the command:

Web2PDF
converted by Web2PDFConvert.com

```
nano /etc/modules
```

Simply backspace out the name of the driver for the NIC being removed and type in the name of the driver module for the new NIC. Then just shut down the system, swap the NICs, and turn on your system. The settings in your current `/etc/network/interfaces` file will be applied to the new NIC. This because of the:

```
auto eth0
```

line in this file. While NIC driver modules can be loaded with optional parameters, it's best to not use any parameters and let the NIC auto-negotiate the speed and duplex of the connection with the switch it is connected to.

Installing A Network Card

If you have an existing Debian system without a NIC and you'd like to add one to put your system on a network, you'll have to add the NIC's driver module to the system configuration and then use the nano text editor to take care of the necessary network files.

The first thing to do (after installing the NIC) is run the `modconf` command at a shell prompt. Highlight the **Net** selection in the modconf menu and then highlight your NIC driver and press Enter to install it, leaving the optional parameters field blank. Then exit out of modconf.

Use nano to edit the `/etc/network/interfaces` file and add an `eth0` section like that shown earlier. You'll also want to edit the `/etc/hosts` file to add a line for you system. If your system's name is 'debian', it's IP address is 192.168.10.50, and your domain name is 'smith.net', you'd want to add the following line to your hosts file:

```
192.168.10.50     debian.smith.net     debian
```

Those are tab characters separating the above entries. You'll also have to use the nano editor create a `/etc/resolve.conf` file and add the entries like those shown earlier. You'll also want to verify the contents of the `/etc/hosts.conf` file as discussed earlier.

Add the NIC driver module, editing two files, and creating a third file is all you should need to do to get your new NIC working. Reboot your system and verify that it all works by trying to ping another workstation on the local network.

Adding A Second Network Card

Setting up a proxy or firewall system requires that the system have two NICs so you would have to add a second network card to your existing networked Debian system. If you want to use a Linux system as a router you would have a system with multiple NICs, one for each of the subnets you wish to interconnect.

Installing a second NIC (or more) in a system that already has one is easy to do. You just have to make sure that the second NIC is configured for a different network (or subnet) than that of the first.

Scenario 1 - Two NICs - Same Driver

If the second NIC you're adding uses the same driver as the one already installed, all you have to do is add the information for the `eth1` (second) NIC to the `/etc/network/interfaces` file. Given the file we had earlier, adding this information would result in a file like this:

```
auto eth0
iface eth0 inet static
        address 192.168.10.10
        netmask 255.255.255.0
        network 192.168.10.0
        broadcast 192.168.10.255
 gateway 192.168.10.1

auto eth1
iface eth1 inet static
        address 172.16.1.10
        netmask 255.255.0.0
        network 172.16.0.0
        broadcast 172.16.255.255
 gateway 172.16.0.1
```

Since the same driver module will be used for both NICs, this is all that is necessary to get both NICs to work (after saving the file and rebooting the system).

The important thing to remember about this scenario is that when the same driver is used for multiple NICs, Linux will designate the

converted by Web2PDFConvert.com

NIC in the lower-numbered PCI slot as the `eth0` interface and the NIC in the higher-numbered PCI slot (closer to the end of the motherboard) as the `eth1` interface.

Scenario 2 - Two NICs - Two Drivers

If you have two different make/model NICs that require two different drivers you'll want to add the `eth1` information for the new NIC to the `/etc/network/interfaces` file as above. You'll also want to run the `modconf` utility as described earlier to pick the driver for it.

When the drivers load at system boot they will initialize the proper NIC. The key word is "when". The NIC designated as `eth0` will be the NIC who's driver module is loaded first. Because modconf will add the selected driver module to the bottom of the `/etc/modules` file, the newly-added NIC will be designated as the `eth1` interface. If you wish to flip this, simply edit the file to place the driver for the new NIC above that of the original NIC.

Using Static Routes

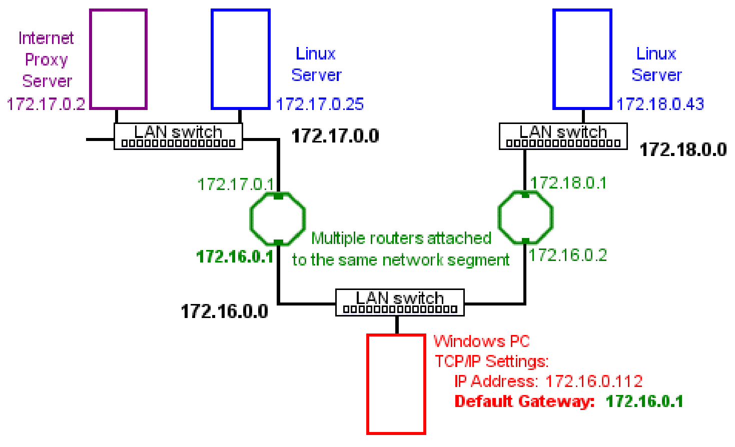
Every computer system (servers and workstations) maintains a "routing table" in memory. This table contains entries that specify which gateway to use to get to other networks or subnets. If a network or subnet only has one gateway (which most do), the routing table isn't very complex. The routing table will typically consist of information taken from the "default gateway" setting in the system's TCP/IP configuration and a local loopback address (127.0.0.1). The local loopback information is only used for testing a TCP/IP configuration.

On Linux systems you can look at the routing table simply by entering `route` by itself at a shell prompt. On Windows systems you have to open up a DOS window and enter the `route print` command. If you have a "default gateway" entered in the system's TCP/IP settings, you should see one or more entries in the routing table with this address. If you look at the routing table, and then dial up your ISP, you'll see that the table was updated to include entries for your ISP's network. In this case, the modem is acting as a gateway off of your local network (or stand-alone system).

Static routes are entries that you manually enter into a system's routing table (they can also be entered on routers). They **are useful when a network segment has multiple "gateways"**, or paths off the local segment. In the diagram below the multiple gateways are represented by two routers. However a multi-homed system like proxy or VPN server could be substituted for one or both of the routers.

If a computer system wants to send data to another computer that's not on the "local" segment, it will send that data out through whatever **default gateway** is specified in its TCP/IP configuration. But what if the destination computer isn't accessible by the default gateway? That's where static routes come in.

Lets consider a scenario using the diagram below. A Windows PC on the 172.16.0.0 network segment runs a special application that accesses data on the Linux server with the address 172.18.0.43. Without a static route, the Windows PC will go through the 172.16.0.1 router trying to find this server because thats what its default gateway is set to. We need to tell the Windows PC to use a different (172.16.0.2) gateway when it wants to communicate with the 172.18.0.43 Linux server.



Windows Static Route commands:

```
route add 172.18.0.43 mask 255.255.255.255 172.16.0.2
                     or
route add 172.18.0.0 mask 255.255.0.0 172.16.0.2
```

On the Windows PC we would open up a DOS window and enter one of the static route commands shown in the diagram above. The first command will set up a static route that applies only to that one specific Linux server (172.18.0.43). Traffic destined for all other systems on that same network segment (172.18.0.22 for example) would still be erroneously sent out the Windows PC's default gateway. The second command would create a static route that would correctly route traffic to any system on the 172.18.0.0 network segment. In other words, the first command sets up a static route to a specific system, while the second command sets up a static route to the entire 172.18.0.0 network. Setting up static routes to specific systems rather than entire networks (or subnets) is

converted by Web2PDFConvert.com

used as an access control (security) measure.

The syntax for the Linux `route` command is similar to the Windows version. The Linux equivalents of the two Windows commands given in the diagram would be:

```
route add -host 172.18.0.43 gw 172.16.0.2
route add -net 172.18.0.0 netmask 255.255.0.0 gw 172.16.0.2
```

Note that you don't need to enter a mask when using the `-host` switch as a mask of 255.255.255.255 is assumed.

> Note that I said when the `-host` switch is used to specify a single system the subnet mask of 255.255.255.255 is assumed. This may not seem logical since this subnet mask means "all network bits and no system bits". Actually, what you are specifying with a 255.255.255.255 mask is a single-address network (the address of the specified system).

In the diagram above, the Windows PC uses the 172.16.0.1 for a default gateway because this is the route to take to get onto the Internet. The default gateway router (the one with the 172.16.0.1 interface) would itself likely have a "gateway of last resort" (default gateway) entry of 172.17.0.2 which would cause it to send all non-local traffic out through the proxy server.

In the above diagram the static routes are being entered on a Windows workstation PC but they are also useful on servers. For example, we have a mail server on our 172.17.x.x LAN segment that needs to pull Internet e-mail from one of our ISP's servers. To get to the ISP server the mail server uses it's default gateway (which is set to our proxy server). However, it also has to pull messages from an internal enterprise Lotus Notes/Domino or Microsoft Exchange server on a different internal network segment (172.30.x.x). Entering a static route on the mail server takes care of the problem:

```
route add -net 172.30.0.0 netmask 255.255.0.0 gw 172.17.0.10
```

The `172.17.0.10` is the address of the router interface (not shown in the above diagram) which connects our `172.17.0.0` LAN segment to the rest of the enterprise network where the internal (Domino or Exchange) mail servers are located. Enabling server-to-server communications is probably the most common use of static routes.

Static routes can be thought of as "explicit" routes. You use them to explicitly tell IP to use a certain gateway for a given destination network or subnet. The default gateway setting could be thought of as an "implicit" static route. You are telling IP that for any destination network or subnet that doesn't have an explicit route defined, send the traffic to the default gateway router.

Subnetting

The information in this section isn't necessary for setting up a Linux server. It is presented here for those that may want to know more about the subnetting process than was presented above.

As mentioned earlier, the decimal numbers that make up an IP address or subnet mask are referred to as "octets" because their values are derived from 8 binary digits. (These 8 binary digits are often written in two groups of 4 to make for easier reading.) And, like all other number systems, the further you go to the left, the more value (weight) the digit has. For example, with a base-10 number system like we use for money, with "$111" the first '1' represents $100, the second '1' represents $10, and the third '1' only represents $1 (a factor of 10 for each position to the left because it's a base-10 number system) all added together. With $101 there wouldn't be any 10-value to add in so it's just $100 plus $1.

With the binary number system it's pretty much the same thing. It's just that because the binary number system is base-2, the value (weight) of a digit **doubles** (a factor of 2 rather than a factor of 10) for each position to the left.

```
128 64 32 16    8  4  2  1
 0   0  0  0    0  0  0  0


128    32                     =  160
 1   0  1  0    0  0  0  0
```

In a binary number, any bit that's a '0' has no value. Any bit that's a '1' has the value of its position. Add up the values of the positions that have a '1' bit and you have the decimal equivanlet. If all the bits were a '1' you would add up the values for all of the positions it would total '255' (as in a subnet mask). See how easy binary-to-decimal conversion is!

When they created the IP addressing system they decided to create the classes based on how many '1' bits started the first octet like so:

Web2PDF
converted by Web2PDFConvert.com

Address Class	First Octet	Possible Values	Network Portion
A	0000 0000 0111 1111	0 through 127 '0' network not used 127 reserved (loopback)	First octet only
B	1000 0000 1011 1111	128 through 191	First two octets
C	1100 0000 1101 1111	192 through 223	First three octets
D	1110 0000 1110 1111	224 through 239 Reserved for multicast addresses	N/A

That's how the numeric ranges for IP addresses in the the various address classes are derived.

Now ask yourself this question; If you can tell the class of an address just by looking at the first few bits, and the class determines how much of the address is the network portion, why do you need a subnet mask?

Technically, you don't. At least not with a "classful" addressing scheme. In classful addressing schemes, routers don't even forward a subnet mask in their routing updates. The class (and subsequently how much of an IP address constitutes the "network" portion of the address) is determined simply by looking at the first few bits in the first octet. The very common RIPv1 routing protocol is strictly classful.

You need a subnet mask because these days most IP-based software, like the networking software that's part of your operating system (and the RIPv2 and OSPF routing protocols), operates in a "classless" manner. It doesn't assume anything so you need a subnet mask to tell it how much of the IP address identifies the network. There are several benefits to using a classless addressing scheme that we won't get into here because they mainly deal with routers. However, play it safe and always assume you're dealing with classless IP addressing and enter an appropriate subnet mask when configuring any IP-based software.

Earlier we compared a computer network with the telephone system, comparing an area code to a network number and a person's telephone number to an individual computer's address. But notice that in a person's telephone number not all of it is unique. Only the last four digits are unique to that person. Many people share the same first three digits. These first three digits are called the "exchange".

If we think of each area code being a network, we can think of an exchange as sort of a "sub network" within that area code network. Actually, area code networks have quite a few exchange sub-networks in them. The telephone network is divided up into this hierarchical structure of a network (area code) being divided into sub-networks (exchanges) consisting of individual nodes (last four digits of a telephone number) because it makes it easier to design systems that route calls. It makes things more "modular".

Breaking up a data network into sub-nets is done for pretty much the same reason. It makes it easier to route and manage network traffic. However, another important reason is that it saves on IP addresses. If an ISP were to assign an entire Class C address range to a business customer with only 50 computers, 201 addresses would be wasted. Subnetting allows an ISP to parcel out their available public addresses in much smaller ranges in order to reduce this waste.

We can take one of the standard Class B private network ranges, say 172.16.0.0, and subnet it by taking a piece of the computer part of an IP address (analogous to a phone number) and using it to sub-divide the network range (as with an exchange).

213-555-1212

```
1010 1100.0001 0000.1010 0000.0000 1100
            172.16.160.12
               N.N.S.C
            255.255.224.0
1111 1111.1111 1111.1110 0000.0000 0000
```

Here we "borrowed" three bits from the computer part of the IP address (by using the subnet mask) to use them to identify several sub-networks. (Remember that the '1's in a subnet mask have to be contiguous so we always borrow from the left end of the computer part of the mask.) Given that we borrowed three bits, and 2 raised to the power of 3 is 8, we can identify up to 8 subnets. Note that the **S**ubnet portion of mask does not have to coincide with an octet boundary. You can borrow as many bits as you want as long as it's at least two. (If you only borrowed one bit you'd only end up with two addresses - because 2 raised to the power of 1 is 2 - and those would be the wire and broadcast addresses with nothing left for computers.) Now you see why they call it a "subnet" mask.

That leaves us with 13 bits to use for computer identification on *each* of those 8 subnets. 2 raised to the power of 13 is 8,192. But like each network, each subnet has a wire and broadcast address so we have to subtract off the two which gives us 8,190 IP

Web2PDF
converted by Web2PDFConvert.com

addresses for computers on each subnet.

Instead of using the private Class B address space in the diagram above as an example, lets say a regional ISP had a public Class B address space assigned to it by ARPA and they had operations in six small cities. They could use a subnetting scheme similar to that above to subnet their space so that each city had its own subnet. The operation in each city would have nearly 8.200 addresses for customers. They could then further subnet their space (by using even more '1's in the subnet mask) to have smaller ranges to give to business customers wishing to have a small range of static public addresses for their Internet servers. Also in the above diagram we show the binary bit patterns and their corresponding octet values.

Subnetting is the practice of taking of given address space and dividing it up into numerous, smaller sub-networks. We mentioned that ISPs will subnet public address space to help conserve IP addresses. On privately-addressed Ethernet LANs subnetting cuts down on the size of things called "broadcast domains".

As local networks are expanded, more and more systems share the same bandwidth. This also means more and more systems are sending broadcasts to locate servers or resolve addresses. A broadcast domain is just a logical area of a network where all systems can see each others broadcasts. When broadcast domains get too big, bandwidth is gobbled up with broadcasts and the network users only get what's left.

Switches are useful in that they cut down on collisions (common in Ethernet networks) by breaking up the **physical** (OSI layer 1) segments (shared cable), but they don't break up the **logical** (OSI layer 3) segment that represents a network (i.e. a collection of machines that all have the same network part of their IP addresses). As a result, switches pass broadcasts. The only way to stop broadcasts is to subnet a network into different logical segments (sub-networks) using routers which operate at OSI layer 3.

Because of the ability to borrow some of the bits from the computer part of the address and use them to identify the subnet part of the address, you can end up with subnet masks like:

255.128.0.0
255.255.248.0
255.255.255.192

These are only a few of a myriad of different numbers you could encounter in subnet masks. To make things easier to document, subnet masks are often indicated using "bit mask" notation. In bit mask notation, the subnet mask is indicated with a slash (/) followed by the number of 1s in the subnet mask. Note that when this notation is used, it is specifying **the length of the prefix**. Here are some examples of subnet masks and their equivalent bit mask notations:

255.0.0.0	/8
255.255.0.0	/16
255.255.128.0	/17
255.255.255.0	/24
255.255.255.128	/25
255.255.255.192	/26
255.255.255.224	/27
255.255.255.240	/28
255.255.255.248	/29
255.255.255.252	/30

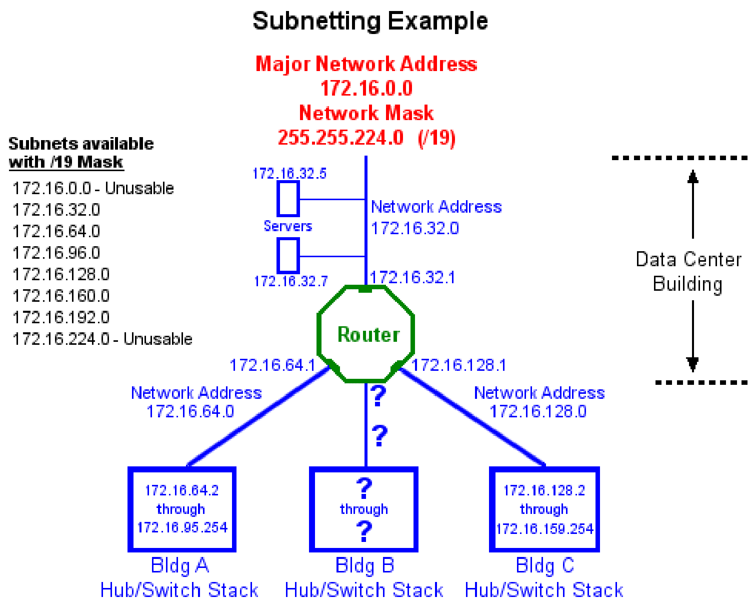
So instead of writing out an IP address and a subnet mask, it is becoming more and more common in networking literature and specifications to see an IP address with a bit mask after it to indicate the length of the subnet mask (how many 1s there are):

192.168.1.5/24

A /30 bit mask (255.255.255.252) yields only two usable computer addresses and is typically only found when connecting two routers together via a point-to-point link (i.e. each router's interface gets one of the addresses).

Lets take a look at a typical subnetting situation. An insurance company has multiple buildings in close proximity. Note that this does NOT constitute a WAN (Wide Area Network). It is commnly known as a "Campus LAN". The buildings are close enough that LAN technologies (Gigabit or Fast Ethernet over fiber for example) are used to interconnect the buidlings, not WAN technologies like frame relay or leased lines. Fiber (100-Base-FX) allows you to extend the length of Ethernet segments so that buidlings too far apart for copper (100-Base-T) can be interconnected.

Web2PDF
converted by Web2PDFConvert.com



In this example the Class B private address network of 172.16.0.0 is subnetted into six usable subnets even though only four are being used. If this network was not subnetted and the router was replaced with a switch, you can imagine what would happen to the bandwidth of this network if every computer in every building could see broadcasts sent from every other computer in all of the buildings. For example, a PC in Building C would send an ARP broadcast intended to resolve a MAC address for another PC in building C but all of the systems in Buildings A and B and the data center would also see it. Multiply this by every PC in the enterprise and you'd have a lot of broadcast activity over the entire network.

Like full Class A, B, or C networks, every subnet has both a wire and a broadcast address. (The wire address is also referred to as the "network address".) No computers can use either of these addresses.

The first subnet can't be used because its address range includes the network (wire) address for the entire 172.16.0.0 network. Likewise, the last subnet can't be used because its address range includes the broadcast address for the entire 172.16.0.0 network (172.16.255.255). In addition, addresses ending in .0 and .255 **on each subnet** can't be used because these are the network and broadcast addresses for the individual subnets respectively. This is one downside of subnetting, available addresses are "lost" (can't be used by systems). The more subnets created, the more addresses that are lost. This is important to consider when you are trying to decide on which private address class to use for your enterprise. If you choose a Class C private address range and you're close to using all available addresses without subnetting, you may want to consider using the Class B space so you have some room to subnet should it become necessary in the future.

QUIZ! Looking at the above diagram, determine the following for Building B:

- Network (wire) address
- Router interface address
- Starting and ending addresses in the range available for systems in the building

When you think you've got the right answers, drag your mouse over the above bullet points to see what they are.

Another QUIZ! This network uses a bitmask of /19 so 19 bits of a 32-bit IP address represent the network/subnet portion of the address (the prefix). How many **usable** addresses are available for systems on each subnet?

When you think you have the answer, drag your mouse over the blank area above. When trying to determine how to best subnet a network, it is often best to determine the absolute maximum possible number of systems (computers, printers, etc) you would have in the largest subnet. For example, if Building A was the largest of all buildings, and it currently has 1,800 systems and would never get over 2,500 systems because there simply isn't room for more, you would figure out how many host bits you would need to support 2,500 systems.

2^{10} = 1024 so 10 computer bits isn't enough
2^{11} = 2048 so 11 computer bits still isn't enough
2^{12} = 4096 so 12 computer bits would be enough

Web2PDF
converted by Web2PDFConvert.com

So if we need 12 bits for systems, 32-12=20. We should have used a /20 bitmask (255.255.240.0) in our example network in the diagram above. What would be the benefit of using /20 instead of /19? Note on the above diagram that the /19 bitmask only gave us six usable subnets with four already being used. What if the insurance company wanted to expand and add remote offices in three other cities? It couldn't be supported with a /19 mask. Even though these offices in remote cities would be connected via WAN technologies like frame relay or leased lines, their subnet addressing schemes would still have to fall within the addressing scheme set up at headquarters.

So here's a few basic principles of network design. We'll use the above example again here:

1. Determine the largest possible number of systems on any given subnet

   2,500 systems

2. Calculate how many computer bits you would need to support this number (remember to subtract two):

   11 bits aren't enough (2,048-2) so 12 bits support 4,096-2 or 4,094 systems

3. Calculate how many bits this leaves you with for subnets (for example, a Class B network has 16 computer bits to start with so after subtracting the required number of computer bits how many computer bits are left for subnetting)

   16 computer bits - 12 bits needed for systems = 4 bits for subnetting

   Note that these 4 bits plus the Class B 16 network bits gives a bitmask of /20.

4. Determine how many subnets the remaining computer bits will allow you to have (remember to subtract two)

   2^4 = 16 - 2 = 14 subnets available

5. If this isn't enough subnets, go to the next higher address class (ex: Class B to Class A)

An important rule to remember with this type of subnetting is that **all subnets must be the same size** (i.e. all have the same number of available system addresses). With this type of subnetting the real waste of IP addresses is not due to the necessity that each subnet have it's own network and broadcast addresses. It's that even point-to-point links, which only have two connections (one at each router interface), have to be set up as their own subnet. So in the above example, even though serial links would only use two addresses, they would have to be given their own subnet consisting of 4,094 addresses. 4,092 of those addresses would never be used.

In the case of the insurance company above, if they opened three remote offices, not only would three subnets be required for systems in each of those offices to use, but three **additional** subnets would be required for each of the three serial (leased line) links to those offices. That's six subnets for three offices! And on each of those three serial link subnets, only two of the available 4,094 addresses would be used. That's (3 x 4092) or 12,276 wasted addresses. This doesn't even take into account all of the unused addresses on each of the building subnets.

What to do about all this address wasting? That's where VLSM (Variable Length Subnet Masking) and CIDR (Classless Interdomain Routing) come in. VLSM allows you to set up subnets of different sizes. However, this requires a that your routers use a more complex "classless" routing protocol where the bitmask value is included with every IP address. (Simpler "classful" routing protocols like RIP v1 assume the subnet mask that corresponds to the class of any given address.)

Address wasting isn't really a problem when you use private IP address ranges except that having all the same-size subnets can cause excessive routing traffic on routed networks. We won't get into the details of VLSM and CIDR here. Cisco has a book called Top-Down Network Design that covers the material nicely. But we wanted to point out the above so you know why the Internet is in danger of running out of addresses. Many companies use standard subnetting like that presented above but do it using the public address space assigned to their company resulting in many wasted routable IP addresses.

DSL vs Cable

As previously stated, if you want to set up one or more serious Internet servers, you're going to need three things:

1. A high-speed symmetrical connection to an ISP
2. One or more static IP addresses
3. An ISP that will host your DNS records

These things are typically only available with "business" accounts offered by cable and DSL providers. Even if the servers will be located in your home, you're going to need a business account to get these features.

Business accounts cost considerably more than you average asymmetrical residential account, but if you're currently paying a Web hosting service to host several Web sites on their servers, you may be able to save some money by hosting your sites yourself.

Web2PDF
converted by Web2PDFConvert.com

There are advantages and disadvantages to this.

Advantages:

- By hosting your own Web sites you're not restricted by artificial lmiits on disk storage, e-mail accounts, and monthly bandwidth allocations.
- You can configure your Web server with all of the bells and whistles that hosting companies charge extra for.
- The money you're paying your Web hosting service and the money you're paying your ISP for monthly dial-up access could be put toward your own broadband connection.
- You can host as many sites as you want on one server (the way the hosting companies do).
- Because your server will host your sites, and your sites only, your sites won't be affected by all of the other Web sites you share a server with when you use a hosting service.
- The same server could also be set up to have Sendmail handle your Internet e-mail needs rather than using an ISP's mail server.
- You could charge others a monthly fee for hosting their Web sites on your server also (but you'd better check with your ISP on this as it may violate their service contract).

Disadvantages:

- You would be responsible for the monitoring and support of your Web server. There is cheap software available to routinely check the availability of your Web sites (www.ipsentry.com), but you would have to be reachable via pager or cell phone 24/7 should the software discover a failure.
- You'd have to set up a means of routinely backing up your server so you can recover from server hard-drive failures quickly.
- You'd want to provide some means of extended power backup should the power go out at your home or office.
- You'd have to become **very** knowledgable in the areas of system security (prevention, monitoring, detection, and recovery).

This last point is especially important if you want to conduct secure (SSL) transactions on your Web sites. Frankly, I wouldn't recommend "hosting your own" if you do need to do this. Hosting companies have a lot of experience with setting up and securing SSL-enabled Web servers. This is one area that may be best left to the professionals.

Also, while the "baby bells" do offer static IP addresses with some of their business DSL packages, they usually only offer asymmetrical service. I suspect this is because they don't want symmetrical DSL cutting into their high-priced T1 and fraction T1 product lines (which are also symmetrical).

Below are some points you should consider when comparing symmetrical DSL and cable broadband services.

DSL	**Cable-modems**
• If you get the more expensive business-class DSL service you receive a DSL router (instead of a DSL adapter for a single computer) which you can plug into an Ethernet hub. This means that all the computers on your network can share the Internet access.	• Most cable-modem providers charge extra for each additional PC you want to have use the Internet access. However, Linksys (www.linksys.com) and D-Link (www.dlink.com) both sell a cable/DSL router for under $100 that would allow you to get around this. But hook it up **after** the cable guy leaves because the cable companies do not support the use of them. (They will also not help you if you have problems setting one up.) Like the straight DSL router, you just plug the cable into the cable/DSL router and then plug that into your hub to share the access.
• The speed of your connection is dependent upon your physical distance from a telephone company facility. You may be limited to a 144 kbps service if you are beyond the distance limitations of the higher-speed service.	• Cable-modem users share their pipe with other cable users. The more business and private users between you and the cable company's office the slower your speed will be.
• Your "up-time" is dependent on the reliability of your local telephone service which is typically less susceptible to weather-related problems and does not involve neighborhood line power provided by your local electric utility.	• If your cable-tv service often goes out due to storms or other weather-related events, your Internet access will also go out. The electronic components used in the cable distribution network (mounted on utility poles, etc. around town) use line power in those neighborhoods. This means that you are also affected by power outages that occur for any reason. If the power goes out anywhere between you and your cable service provider, your service will go down.

converted by Web2PDFConvert.com

- Service considerations include:
  - You will be dealing with your local baby bell for the line (installation and maintenance), the DSL provider for the DSL service over that line, and an ISP.
  - Cases of installations taking over a month to complete are not uncommon.
  - A lot of DSL providers have gone out of business.

- Service considerations include:
  - The cable company will likely be slower to respond to data line problems. As part of their contract with municipalities, most cable companies have to respond to "no picture" service calls quickly when it comes to video service. However, they are under no such obligation for data service. A friend of mine was told it would take three weeks to get a technician to his house when his cable-modem died.
  - Because most municipalities only have one cable service provider, the lack of competition means your cost for the service could rise substantially over the next few years (look at how much video service has gone up in the last few years).
  - Most cable companies block SNMP (network device status monitoring) traffic.
  - Not all cable companies host customer DNS records.
  - Not all cable companies allow you to "dial in" when you're away from home.
  - It's the cable company. **:^(**

If you've got more time than money, hosting your own may be the way to go. However, maintaining the 24/7 operation of Internet servers is a big responsibility.

Did you find this page helpful ?
If so, please help keep this site operating
by using our DVD or book pages.

Site, content, documents, original images   Copyright © 2003-2013   Keith Parkansky   All rights reserved
Duplication of any portion of this site or the material contained herein without
the express written consent of Keith Parkansky, USA is strictly prohibited.

This site is in no way affiliated with the Debian Project, the debian.org Web site, or
Software In The Public Interest, Inc. No endorsement of this site by the Debian Project
or Software In the Public Interest is expressed or implied. Debian and the Debian logo
are registered trademarks of Software In The Public Interest, Inc. Linux is a registered
trademark of Linus Torvalds. The Tux penguin graphic is the creation of Larry Ewing.

LIABILITY

IN NO EVENT WILL KEITH PARKANSKY OR BLUEHOST INCORPORATED OR ANY OF ITS' SUBSIDIARIES BE LIABLE TO ANY PARTY (i) FOR ANY DIRECT, INDIRECT, SPECIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF PROGRAMS OR INFORMATION, AND THE LIKE), OR ANY OTHER DAMAGES ARISING IN ANY WAY OUT OF THE AVAILABILITY, USE, RELIANCE ON, OR INABILITY TO USE THE INFORMATION, METHODS, HTML OR COMPUTER CODE, OR "KNOWLEDGE" PROVIDED ON OR THROUGH THIS WEBSITE, COMMONLY REFERRED TO AS THE "ABOUT DEBIAN" WEBSITE, OR ANY OF ITS' ASSOCIATED DOCUMENTS, DIAGRAMS, IMAGES, REPRODUCTIONS, COMPUTER EXECUTED CODE, OR ELECTRONICALLY STORED OR TRANSMITTED FILES OR GENERATED COMMUNICATIONS OR DATA EVEN IF KEITH PARKANSKY OR BLUEHOST INCORPORATED OR ANY OF ITS' SUBSIDIARIES SHALL HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND REGARDLESS OF THE FORM OF ACTION, WHETHER IN CONTRACT, TORT, OR OTHERWISE; OR (ii) FOR ANY CLAIM ATTRIBUTABLE TO ERRORS, OMISSIONS, OR OTHER INACCURACIES IN, OR DESTRUCTIVE PROPERTIES OF ANY INFORMATION, METHODS, HTML OR COMPUTER CODE, OR "KNOWLEDGE" PROVIDED ON OR THROUGH THIS WEBSITE, COMMONLY REFERRED TO AS THE "ABOUT DEBIAN" WEBSITE, OR ANY OF ITS' ASSOCIATED DOCUMENTS, DIAGRAMS, IMAGES, REPRODUCTIONS, COMPUTER EXECUTED CODE, OR ELECTRONICALLY STORED, TRANSMITTED, OR GENERATED FILES, COMMUNICATIONS, OR DATA. ALL INFORMATION, METHODS, HTML OR COMPUTER CODE IS PROVIDED STRICTLY "AS IS" WITH NO GUARANTY OF ACCURACY AND/OR COMPLETENESS. USE OF THIS SITE CONSTITUTES ACCEPTANCE OF ALL STATED TERMS AND CONDITIONS.

Web2PDF
converted by Web2PDFConvert.com