Daryl's TCP/IP Primer

Addressing and Subnetting on the Near Side of the 'Net

Contents

If you find this site helpful,
please make a donation to the Red Cross:
www.redcross.org/donate/donate.html

1. Disclaimer

The Boring Disclaimer. The rest of Daryl's TCP/IP Primer is a better read.

---

2. Overview and Scope

This document is designed to give the reader a reasonable working knowledge of TCP/IP subnetting, addressing, and routing. It is not intended to be complete, or to cover all issues. This is targeted toward LAN administrators just moving to TCP/IP, however it should help anyone who wants to know a little (more) about how TCP/IP works. This document does not, generally, apply to dial-up SLIP/PPP connections.

The difference between this (a primer) and an FAQ, is that most FAQ's, in practice, tend to be question-and-answer oriented, and generally seem to try to cover ALL issues, not just the ones frequently asked about. This primer is intended as a starting point for someone who has an interest in the subject, but doesn't know where to start or what questions to ask. This should also help to broaden the understanding of people who have worked with TCP/IP for a while, but either haven't had the time to study all the less-than-useful theory behind the subject, or have been somewhat overwhelmed by the many theoretical details and have missed the big picture.

This is maintained in HTML. I have made it available as one large page for the benefit of those who prefer to print off a copy and read it that way. Also useful for sharing via hard copy. If you choose to print this out and distribute this, I ask that you distribute it in its entirety, and that you don't charge for it.

Feedback, of course, is always greatly appreciated, and will help determine the direction and growth of this living document. In fact, just a quick email to say "thanks" (if it helped) will help motivate me to keep this current and expanding :-)

---

3. Intro to Ethernet

Developed in the early 1970's, Ethernet has proven to be one of the most simple, reliable, and long-lived networking protocols ever designed. The high speed and simplicity of the protocol has resulted in its widespread use.

Although Ethernet works across a variety of layer one media, the three most popular forms are 10BaseT, 10Base2, and 10BaseF, which use unshielded twisted pair (UTP), coaxial, and fiber optic cables respectively. UTP is used in a "star" configuration, in which all nodes connect to a central hub. 10Base2 uses a single coaxial cable to connect all workstations together in a "bus" configuration, and does not require a hub. 10BaseF uses fiber optics, which, though expensive, can travel long distances (2km) and through electrically noisy areas.

An interesting difference between coaxial Ethernet and other types is that coax Ethernet is truly a one-to-many (or, 'point-to-multipoint') connection; fiber and UTP connections are, from a layer one perspective, one-to-one (or, 'point-to-point') connections, and require an additional networking device (typically, a repeater, or Ethernet hub) to connect to multiple other workstations. This is why coax Ethernet does not require a hub, and Ethernet over other media typically does.

Ethernet Topologies			
	Pro	**Con**	**Typical Use**
10BaseT	*Very reliable- one fault usually doesn't affect entire network.	*Relatively short distance from hub to workstation (100m). *Requires a lot of wiring (a separate link for each workstation.)	*Offices and home networks.
10Base2	*Cheap- no hub required, no	*Any break in connectivity	*Small or home

	wiring except from station to station. *Well shielded against electrical interference. *Can transmit longer distances (200m).	disrupts entire network segment. *Problems can be very difficult to troubleshoot.	networks, hub to hub links.
10BaseF	*Long distance networking (2000m). *Immune to electrical interference.	*Very expensive to install.	*Long distance hub-to-hub or switch-to-hub links.

Ethernet is like a bunch of loud people in an unmoderated meeting room. Only one person can talk at a time, because communication consists of standing up and yelling at the top of your lungs. People are allowed to start communicating whenever there is silence in the room. If two people stand up and start yelling at the same time, they wind up garbling each others' attempt at communication, an event known as a "collision." In the event of a collision, the two offending parties sit back down for a semi-random period of time, then one of them stands up and starts yelling again. Because it's unmoderated, the likelihood of collisions occurring increases geometrically as the number of talkers and the amount of stuff they talk about increases. In fact, networks with many workstations are generally considered to be overloaded if the segment utilization exceeds 30-40%. If the collision light on your hubs is lit more often than not, you probably need to segment your network. Consider the purchase of a switch, described below.

Ethernet hubs are used in 10BaseT networks. A standard hub is just a dumb repeater-- anything it hears on one port, it repeats to all of its other ports. Although 10BaseT is usually wired with eight wire jacks (known as RJ45 connectors), only four wires are used-- one pair to transmit data, and another pair to receive data. While transmitting, an Ethernet card will listen to its receive pair to see if it hears anyone else talking at the same time. These two behaviors (listen for silence before talking, and detect other people talking at the same time) are described by the acronym people as CSMA/CD, or "Carrier Sense Multiple Access, Collision Detection."

One hundred megabit Ethernet (100BaseTX) works just like ten megabit Ethernet, only ten times faster. On high-quality copper (known as Category 5, or CAT 5 UTP), 100BaseTX uses the same two pair of copper to communicate. If you have standard network-quality copper, an alternative is to use 100BaseT4, which uses all four pairs, but can communicate at 100Mbps on CAT 3 UTP.

Gigabit Ethernet works just like hundred megabit Ethernet, only ten times faster (1000Mbps, or 1Gbps.) There are some Gigabit Ethernet devices floating around out there, but it's unlikely that you'll find such devices on the small LANs that you'd find on the "Near Side of the 'Net."

If your conference room gets too busy, you may consider splitting them into two groups by putting a partition wall with a door between the halves, and putting a person in the doorway. This person would listen to the conversations in both rooms, memorize the names (Ethernet card addresses) of everyone in each room, and forward messages from room to room when necessary. A device to do this is called a "transparent bridge." It's called "transparent" because it's smart enough to learn the Ethernet addresses on its own without the workstations suspecting anything is going on. ["Source-route bridges" are uncommonly used so I'm not going to discuss them.]

Ethernet switches are little more than high-speed, multi-port bridges. They learn the Ethernet addresses of everyone attached to each port, and make intelligent forwarding decisions based on Ethernet card address (aka MAC address.) Because communication between 100Mbps and 10Mbps networks requires buffering, Ethernet switches are often used for this purpose. Many inexpensive switches have many 10Mbps ports and one or two 100Mbps ports. Typically, you would connect your server(s) to the 100Mbps port(s), and workstations or entire hubs to the 10Mbps ports. The

buffering and intelligent forwarding allows another interesting feature to exist-- "full-duplex" Ethernet. "Half-duplex" means you can either talk or listen, but not both, at a given time, such as when using a radio. "Full-duplex" communication means you can talk and listen at the same time, such as when on the phone. Since 10BaseT uses separate pairs of copper for sending and receiving, it's physically possible to do both if there are no other workstations on your network segment-- which is the case if you are directly attached to a switch. Note that both the switch port and your network card must be configured for full duplex operation for this to work, but the result is worth it: a full 20Mbps for "regular" Ethernet and a whopping 200Mbps of bandwidth available for full-duplex fast Ethernet. Since collisions are eliminated, the 30% rule does not apply. When considering the purchase of a switch, there are a few important considerations, not all of which may apply to your requirements:

- Does the switch support 100Mbps on any ports? How many, and will it autodetect 10/100BaseT?
- Does the switch support full duplex? Even on the 100Mbps ports?
- How many MAC (Ethernet card) addresses does it store? 500? 5000? "Unlimited" is not a rational answer.
- Some "workgroup" switches only allow one MAC address per port, so these would not be suitable if you plan to connect hubs to switch ports.
- Some hubs are advertised as "switching hubs" but actually require the purchase of a separate "switch module" to function as such. I won't name names, but I'll never buy a hub, nor anything else, from that manufacturer, if given a choice. I consider it to be an outright lie.
- You tend to get what you pay for. If a switch seems unreasonably inexpensive compared to other switches that appear to have similar specs, look closer, or check the detailed specs on the manufacturer's web site. Often, you'll find that a cheap switch either isn't a switch at all (see last item) or only allows one workstation per port (see item above last item.)

<SOAPBOX>
I have increasingly seen people install 100Mbps networks without paying any attention to whether or not there is a need to do so. Most smallish networks do not need 100Mbps switched Ethernet; in many cases, excellent results can be obtained by purchasing a 10/100 Ethernet switch. Connect the 100Mbps port to the server, and the 10Mbps ports to the workstations or hubs. You greatly increase the amount of bandwidth available, without pulling new cable and installing new cards in the workstations. Even switched, full-duplex 10Mbps Ethernet increases the available bandwidth by almost 600%. People talk about the panacea of "reduced latency" that switched networks provide, but most modern protocol implementations are designed to be almost completely unaffected by a few milliseconds of latency. Most computing environments are disk I/O bound, not network or CPU bound; yet people will recable their networks and install new network cards, or buy servers with faster and faster (read: more and more idle) processors, but the most performance benefit typically lies in installation of more memory in the server, or addition of a caching RAID controller and a RAID-based disk subsystem. Before blindly "upgrading stuff" to improve the speed of your network, try to find out where the true bottleneck lies.
</SOAPBOX>

---

4. The Bottom of the OSI Model

The OSI Networking Model is used as a reference point to describe how the various "layers" of networking interoperate. For this discussion, I will describe the bottom three layers:

Layer	Name	Protocols / Terms	Devices that operate in this layer	Addresses are called...

3	Network	IP, IPX, AppleTalk	Routers	Network Addresses
2	Datalink	Ethernet, Token Ring, PPP, SLIP, HDLC	Bridges, Switches, Repeaters, Hubs	Datalink, or MAC* addresses
1	Physical	Unshielded Twisted Pair, Shielded Twisted Pair, Coax, Twinax, Serial cable	Modems, CSU/DSUs	N/A (cables don't have addresses)

*MAC, in this case, stands for Media Access Control, not to be confused with an address for a Macintosh...

Combinations that include a term from each layer describe fully how a packet is getting from a given point "A" to a directly connected point "B". For example, A may be talking to B using IP over Ethernet over Unshielded Twisted Pair; or, "my computer talks to my ISP using IP over PPP over a serial cable" (a modem is simply a serial cable extender in this sense.) From the physical layer standpoint, devices have no addresses. On the datalink layer, all Ethernet and Token Ring cards all have 6-byte addresses manufactured into them, called MAC addresses (nothing to do with Macintoshes.) Point-to-point links such as serial lines do not have MAC addresses, which creates special cases from a data transmission standpoint, that are outside the scope of this document.

The Physical layer defines the electrical media and signaling used to transmit information on a wire (or wires.) The datalink layer defines the format of the data as it is transmitted (e.g., an Ethernet frame.) Network layer information is encapsulated inside datalink layer frames. If you look at an IP packet on an Ethernet wire it would look something like this:

Ethernet Header (with dest and src MAC addr)	IP Header (with dest and src IP addr, and checksum)	Actual Data

Note that this indicates that, in order for two Ethernet-attached stations to communicate with each other via IP, they must know the MAC address of each other. If station "A" knows the IP address of station "B", and knows station "B" is on the same subnet, station "A" will issue an Address Resolution Protocol (ARP) broadcast. An ARP broadcast is a message that says, "Who out there is 192.168.1.1?" The TCP/IP software running on the workstation or router at 192.168.1.1 is responsible for sending back an ARP response that says, "I am 192.168.1.1, and my MAC address is 08:00:09:AF:24:33." All stations keep an ARP cache with the MAC and IP addresses of all the stations it recently communicated with directly. Try the command "arp -a" sometime on a UNIX or Windows workstation; on a Cisco router, the command is "show arp".

Note that layer 1 devices are "invisible" to layer 2; and layer 2 devices are "invisible" to layer 3. In other words, TCP/IP doesn't care if you're running over Ethernet or Token Ring, as long as it's connected properly. In fact, you can put bridging and/or switching devices on your network without disturbing any of your IP subnetting. Similarly, you can convert between different types of media (e.g., coax to twisted pair) without any layer 2 devices being aware of the change. To change layer 1 media, you typically need a layer 2 device (e.g., "I have a Ethernet Coax to Ethernet Twisted-Pair repeater".) To change the layer 2 protocol (e.g., Ethernet to Token Ring) you typically need a layer 3 device (a router.) All this is good, since it allows some measure of media independence within the network; you can run IP over just about anything better than two cans and a string, and even that, if you can find transceivers to handle it ;-)

---

5. Why is IP so much more difficult than IPX?

I have gotten some interesting feedback on the title of this section. From a LAN administrator's standpoint, IPX is almost completely auto-configuring. Since TCP/IP requires substantially more

administrator understanding and time to properly implement, then IP, from a LAN administrator's standpoint (this document's target audience), is substantially more difficult to work with than IPX. You don't find 40+ page documents on the Internet about "the fundamentals of IPX", do you?

The four items you need to use IP effectively on the Internet (that you don't need to set up an IPX workstation) are the IP Address, the IP Subnet Mask, the IP Address of the Default Router, and the IP Address(es) of your Domain Name Servers (DNS Servers, often shortened to "Name Servers.")

IP Addresses: IP uses 4-byte addresses, like 192.168.1.1. IPX uses 10-byte addresses, like 10000001:0000C04C1141. Those happen to be the IP and IPX addresses of the workstation I'm using now. "But wait," you ask, "I've used IPX before and all it uses are four byte addresses." Well, that's not entirely correct. The 4-byte "IPX Address" configured into IPX-based servers is only the *network* portion of the address. All addresses used by routable protocols have a "network" portion, which gets your packet to your nearest router, and a "host" portion, which indicates which host station you are on that routed segment. The 4-byte "IPX Address" you define is actually a 4-byte "IPX *Network* Address." The other 6 bytes is the hardware address of your NIC. Since IP addresses don't use the unique hardware address of your NIC, you must define them manually (or semi-manually by configuring a BOOTP or DHCP server, a task which is currently outside the scope of this document.)

IP Subnet Masks: Subnet masks (described in more detail in the next section) are used in IP to determine which part of the four-byte IP address describes the network you're on, and which part describes which host you are on that network segment. In IPX, the first four bytes always indicate the network you're on, and your six byte MAC layer address indicates which host you are on the network segment. In IP, the portions used to describe which network you're on can range from the first 8 bits of the address, to including all except the last two bits of the whole address. More in the next section.

Default Router: In IPX, routers are identified by sending out a broadcast that says, in essence, "Hey? Who out here is a router?" In IP, there has historically NOT been any automatic method for router discovery. There is now a protocol for IP router discovery, but it is not widely implemented. Therefore, you must tell the workstation what the address of the local router is. Note that with end-station PPP (like Win95 Dial-Up Networking), the default route is automatically set to, "out the serial cable." You do not need to set more than one default route. If the default router feels the packet would reach a destination better through a different router, the default router will tell your IP stack to use the other router (this is an ICMP Redirect.) If you specify no default route, no packets from that workstation can make it off the local wire; therefore, it is better to set a *wrong* default route than *no* default route. If in doubt, set the default route to the address of any known router on the local subnet.

DNS: In IPX, designed by Novell, the names (and corresponding addresses) of ALL services available on the network are stored in ALL Netware servers as a SAP table (SAP stands for Service Advertising Protocol.) Netware servers will share SAP information with each other automatically. Unfortunately, since ALL servers must know about ALL services, SAP tables can get very unwieldy on large networks, and without the benefit of advanced routing/advertising algorithms (NLSP), can flood networks with SAP broadcasts. The way IP handles name-to-address translation is called DNS. When you query your DNS server for a given name's address (such as www.novell.com), the DNS server will query one of the "root" servers for .COM. The root server tells the DNS server the address of the "authoritative" DNS server for novell.com. Your DNS server then asks the DNS server of novell.com what the address of www.novell.com is; when novell.com's DNS ponies up the address of www.novell.com, your local DNS "remembers" where www.novell.com was, so it doesn't have to look again the next time someone asks for that name's address. Note that DNS uses special records for mail routing, called MX records, that usually differ from the host addresses. Therefore, an ftp (or www, or gopher,...) connection to microsoft.com probably reaches a different address than mail sent to somebody@microsoft.com. Of course, the giveaway that you're talking mail ("MX"

record) addresses, rather than host ("A" record) addresses, is the "@" in the address. Host names never have @ symbols, which is why you connect to www.microsoft.com, never www@microsoft.com.

BOOTP and DHCP: BOOTP was designed to ease the configuration of desktop IP stacks. In a nutshell, a BOOTP-enabled workstation sends out a broadcast BOOTP request, which is answered by a BOOTP server. The answer includes workstation address, subnet mask, default route, and DNS location(s). DHCP is generally accepted as the "next generation" of BOOTP. Whereas BOOTP statically assigns IP addresses by MAC address, DHCP supports address "leasing" where an address is granted to a specific MAC address for a finite amount of time, and can be reused after a specified amount of time. DHCP also supports fields beyond BOOTP, most notably returning information about the location of WINS server to Windows NT clients, and the location of DSS servers to Netware/IP clients. (A DHCP service is included with NT, and is available for free download as part of the Netware/IP upgrade for Netware 4.10 servers, see http://support.novell.com.)

---

6. IP Addresses, Subnet Masks, and Subnetting

There are two sets of rules for subnetting TCP/IP networks. The original set of rules can be found in RFC 950, and the new set of rules can be found in RFC 1812.

<RANT>
Although RFC 1812 came out in June of 1995(!), most certification tests still test you on the RFC 950 rules, for (in my opinion) one of the following reasons:

- Their software still follows RFC 950 rules (this is rare.)
- Since RFC 1812 simplifies things significantly, there's not enough material to test on. Test items from RFC 950 are added as "filler".
- They are ignorant of the fact that the material on their tests has been out of date for more than five years.
- They are mean-spirited, perniciously forcing you to learn material that will never be relevant to your job.

Please keep the fact that the following information in Part A is no longer relevant to the real world; however, it may be necessary to understand it if:

- You are planning on taking one of the aforementioned tests; or
- You need to communicate with someone who holds certain certifications, and believes everything they were tested on still has some relationship to the way the 'Net works.

I still get many questions to the effect of, "I don't understand. This source says I can break this Class C into six subnets, but this other source says you can break the same network into eight subnets. What gives?" The short answer is, it depends on which RFC is valid in your environment. If you are still running an unpatched Netware 3.11 server, you will find yourself constrained by RFC 950 rules. However, a patch has been available for that 1991 platform since sometime around 1995; if you are still running version 1.00 of the Netware TCPIP.NLM, then IP Routing issues are the least of your concerns. :-)
</RANT>

Part A: The World According to RFC 950 (the old way of doing things)

An IP Address is broken up into three parts: the network portion, the subnet portion (optional), and the host portion. The size of the network portion is determined by the first byte of the address:

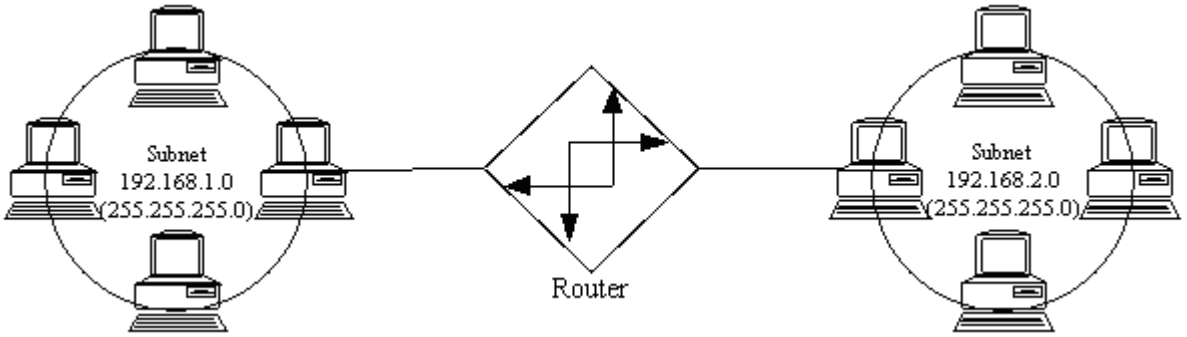
First Byte	Class	Network Mask (explained later)
1-126	"A"	255.0.0.0
128-191	"B"	255.255.0.0
192-223	"C"	255.255.255.0

Note: people often refer to any subnet with a mask of 255.255.255.0 as being a class "C" network; however, the only "true" class "C" networks have a first byte in the range of 192-223. This becomes important when you start subnetting.

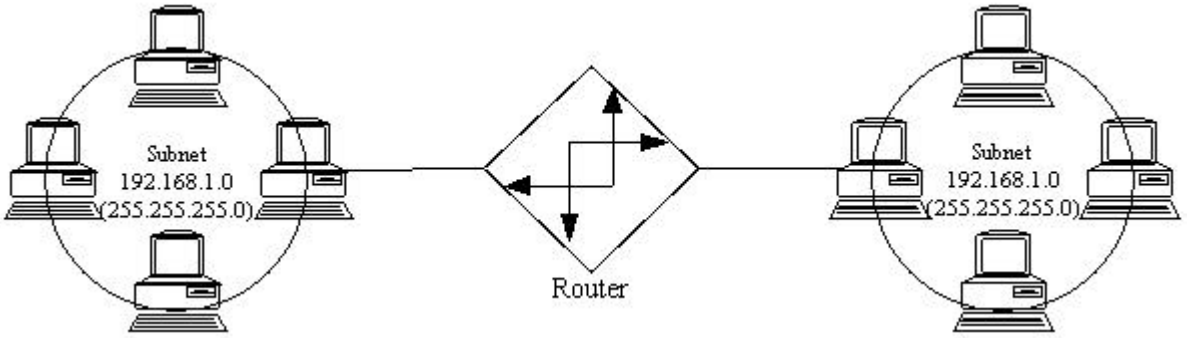
The Subnet portion of an IP address is actually optional, and, in fact, is rarely used on class "C" networks. Generally, you can subnet any network you have control over, in any valid way you want. The tricky part is understanding what is valid.
Lets start with some ground rules:

- All hosts on the same subnet must agree on the subnet mask, particularly the routers. Otherwise, packets actually intended for another subnet may never leave the existing subnet: a host won't give to the router a packet it thinks is destined for the local segment. This behavior is important to understand: the router doesn't automatically forward packets, the hosts have to actually *give* the packets *to* the router.
- No two different subnets can include the same host address. This can get tricky when subnetting in an unusual manner.
- The top and bottom host numbers are reserved; the bottom one (usually ?.?.?.0) is shorthand for the whole subnet, and the top one (usually ?.?.?.255) is the broadcast address. Some implementations also use .0 as a broadcast address, so it is never safe to use for a host.
- The bits in the subnet portion cannot be all ones. This requires a bit of binary arithmetic to determine which subnets would be invalid

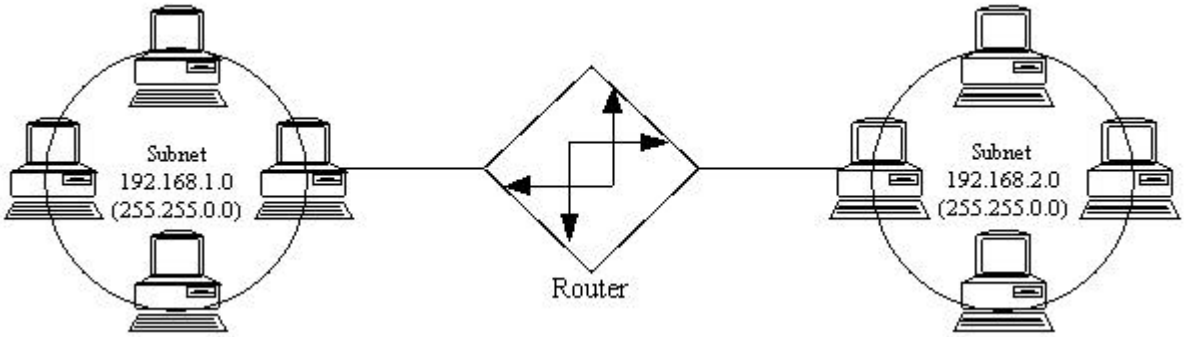
Valid Configuration:



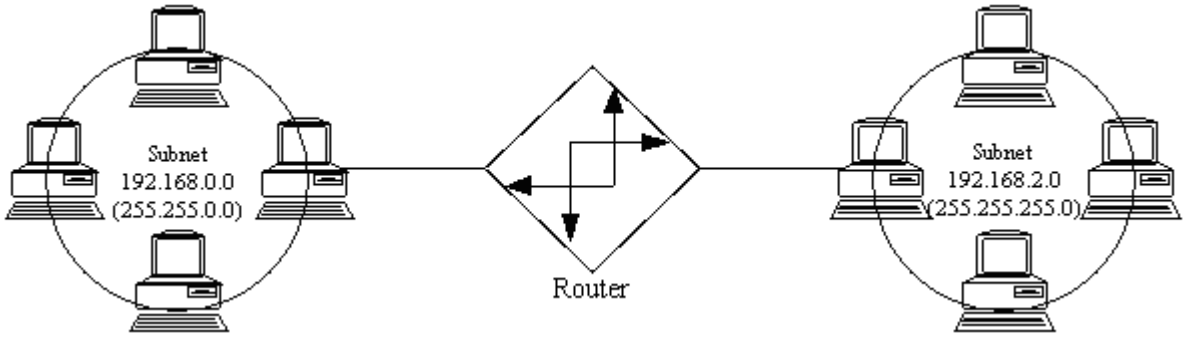
Invalid Configurations:



...This is invalid since the [exact] same subnet exists on both sides of the router.



...This is invalid since the same subnet exists on both sides of the router. Watch that subnet mask! (See below.)



These images created using SmartDraw. Click Here for a free trial copy.

...This is invalid because a the same host address could be "valid" on either subnet, e.g. 192.168.2.100. Even though the right side subnet is valid by itself, it is actually a small piece of the left side network.

Exception!	Address overlap of this sort is *usually* not allowed between two physical subnets: unless the router was specifically configured to "pretend" it was every address on 192.168.2.0 for its left-side interface in the diagram, it would be impossible for hosts on one side of the router to communicate with hosts on the other side. In this diagram, the 192.168.2.0 subnet is known as a "stub subnet"; the process of pretending you are hosts you're not, in order to facilitate routing packets to a stub subnet, is known as "proxy arp." No two hosts on the Internet can have the same IP address. If you create a stub subnet, no host on the "main" side can have an address that might be valid on the "stub" side. [Please also note that the diagram in question is talking about two physical subnets attached to one router, not routing tables on upstream routers, which would aggregate both networks into one route of 192.168.0.0/16.]

The Glossy Explanation

When using a subnet mask of 255.255.0.0, the first two bytes indicate the network you're on, and the last *two* bytes indicate the host you are on that network. Very rarely will you find a network segment with 65,534 hosts on it, though. You'll only find network masking like that used closer to the Internet backbone, in the context of, "All them hosts [and subnets thereof] are thataway." Now, that brings up one of the nice features of subnet masking: you can lump a bunch of networks together by using unusual subnet masking; however, that sort of activity generally doesn't happen on the near side of the 'net.

When using a subnet mask of 255.255.255.0, the first three bytes indicate the network you're on, and the last byte is the host you are on that network. Hosts .1 through .254 are available.

By using a subnet mask of 255.255.255.128, you can split that network into two halves, the first half containing the host addresses .1 through .126, the second half containing the host addresses .129 through .254. Note that on a true class "C" network, you can't use the top subnet, since the bit in the subnet portion (one bit on a class "C") would be one (refer to ground rule "D".)

By using a subnet mask of 255.255.255.192, you can split the network into four portions, each with 64 hosts (62 usable.) Subnetwork one includes the addresses .1 through .62, subnetwork two includes the addresses .65 through .126, subnetwork three includes .129 through .190, and subnetwork four includes the hosts .193 through .254. On a true class "C" network, subnetwork four is not valid.

You can not arbitrarily cut a piece out of one network and place it on another segment; the best you can do with a given subnet (or network) is chop it in halves, or quarters, or eighths, or sixteenths... (note the "powers of two" progression; this is an effect of stealing bit positions from the host address section, and giving those bits positions to the subnet portions. It gets complicated...)

Part B: The World According to RFC 1812 (the "new" way of doing things)

or, By The Way - Forget Everything You Just Learned, It Became Obsolete in 1995

Under RFC 1812, things have changed..!

Perhaps the most significant change on the near side of the 'net under RFC 1812 is Classless Inter-Domain Routing (CIDR, pronounced "Cider"). Under CIDR, the concept of separate "network" and "subnet" portions is now considered outdated, and is being replaced by a "classless" addressing scheme where addresses can be "subnetted" more freely, without consideration of the "class" of address. With the removal of the subnet portion, and the liberalization of (what is now called) the network prefix, there is no longer a consideration of whether or not the bits within the subnet portion are all ones; in other words, you no longer lose a subnet when you break up what used to be known as a class "C" network. You can also aggregate formerly class "C" networks together using network prefixes fewer than 24 bits long. For example, you could combine the formerly class "C" networks 192.168.2.0 and 192.168.3.0 into a single subnet with 510 usable addresses, by using a network mask of 255.255.254.0. What you're really saying here is that the last bit of the third byte now belongs to the "host number" portion of the address, and the "network prefix" is 23 bits (two bytes and seven bits) long. Therefore, the two networks being combined must be contiguous, and the third byte must be even on the lower numbered network. You could *not* combine, for example, 192.168.2.0 and 192.168.5.0; not could you combine 192.168.11.0 and 192.168.12.0. You could follow similar rules to combine four contiguous class "C" style networks, but the third byte of the lowest numbered network would have to be a multiple of four. This sort of thing is routinely done (on an increasingly larger scale) as you get closer to the Internet backbones.

Most of the other effects of RFC 1812 and CIDR routing affect areas of the 'net closer to the backbone, and mostly work to reduce the size (or at least the rate of growth) of routing tables in backbone routers.

Part C: Huh? (or, Perhaps you could apply an analogy to all this?)

A good analogy for IP addressing and packet forwarding (routing) is the snail mail analogy. Consider an IP packet to be an envelope containing data, and having an address on the front. Every TCP/IP-enabled network interface can be compared to a mailbox. Every mailbox (interface) has an IP address. The four bytes of an IP address can be compared to the state, city, street, and house number fields on the front of a snail mail envelope. A router in this analogy is a post office, that sorts and

forwards mail based on the address on the envelope (packet header.) If the address is on the same street (based on the subnet mask,) the envelope (packet) is sent directly to the destination mailbox (interface) via local courier (Ethernet?). If the address is determined to be on another street, or in another city or state, the envelope (packet) is delivered via local courier (Ethernet?) to the street's post office (router), where the postal workers (routing software) sort and forward mail based on established post office sorting procedures (routing tables.) The breakdown in this analogy, of course, is that no routing software has ever been known to shoot people. (Just Kidding :-)

---

7. Subnetting, Bit by Bit

A. Binary arithmetic

You may have heard that computers represent all numbers as "bits", or "zeros and ones." It would be more fair to say that computers work primarily with groups of eight 0's or 1's, called bytes. In practice, most desktop PC's work with clumps of four bytes at a time, or 32 bits. That's why 80386 through Pentium IV processors are called 32-bit processors. [Athough Pentium class processors have some 64-bit attributes such as a 64-bit external memory bus,they still do most operations as 32-bit operations.]

Now, think back to first grade math, when the teacher was describing the decimal numbering system. As it happens, it's called "decimal" (the root of the word is from Latin *decima*, a tenth part or tithe) because it's a numbering system that uses ten numbers: the numbers zero through nine. If you need to represent a number larger than nine, you have to start adding additional digits; then the teacher described the ones place, the tens place, the hundreds place, etc. For example, the number 45678 has a four in the "ten thousands" place, a five in the "thousands" place, a six in the "hundreds" place, a seven in the "tens" place, and a 8 in the "ones" place:

Ten Thousands	Thousands	Hundreds	Tens	Ones
4	5	6	7	8

Since computers work in binary, and only have "0" and "1" to work with, they have to start new digits ("binary places", not "decimal places") as soon as they get past the number one! In decimal, the "decimal places" were all powers of ten:
10^0=1,
10^1=10,
10^2=100,
10^3=1000, etc.
In binary, the "binary places" follow powers of two:
2^0=1 (1 binary),
2^1=2 (10 binary),
2^2=4 (100 binary),
2^3=8 (1000 binary),
2^4=16 (10000 binary),
2^5=32 (100000 binary),
2^6=64 (1000000 binary),
2^7=128 (10000000 binary),
2^8=256 (100000000 binary), etc.

The number 45678 is represented in binary as follows:

(Binary Places, expressed as Decimal:)	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

	1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	0

(Add up the columns where you find ones: 32768 plus 8192 plus 4096 plus 512 plus 64 plus 32 plus 8 plus 4 plus 2 equals 45678!)
Counting to Forty:

Decimal	Binary		Decimal	Binary		Decimal	Binary		Decimal	Binary
1	1		11	1011		21	10101		31	11111
2	10		12	1100		22	10110		32	100000
3	11		13	1101		23	10111		33	100001
4	100		14	1110		24	11000		34	100010
5	101		15	1111		25	11001		35	100011
6	110		16	10000		26	11010		36	100100
7	111		17	10001		27	11011		37	100101
8	1000		18	10010		28	11100		38	100110
9	1001		19	10011		29	11101		39	100111
10	1010		20	10100		30	11110		40	101000

Now, an IP Address is four bytes, eight bits each, represented as decimal numbers with periods in between; for example, 10.5.72.230. This number can be represented in binary (remember when I said that IP Addresses are best expresses as 32-bit binary numbers? I *did* mention that, didn't I?) as b00001010.00000101.01001000.11100110. (The "b" means "binary"; that and the periods are added for your convenience.) Now, 2^{32} (two to the thirty-second power) is 4294967296, or just over four billion. So, theoretically, there are over four billion IP addresses available to the world; so why is there a shortage? (Oh yeah, have you heard? There's a shortage. Last I checked, they're projecting to run out of IP addresses around the year 2025.) Well, as it turns out, trying to keep track of where four billion individual hosts are would be pretty much impossible for equipment today, and certainly impossible for equipment many years ago when TCP/IP routing was being developed. So, routing was (over)simplified by splitting the IP address space into "classes"; those IP addresses whose first byte was in the range 1-126 would belong to networks of 16,777,214 (2^{24}-2) hosts; these were called "Class A" networks, and there are 127 of them. In Class A networks, the first eight bits are the "network portion", and the last 24 bits are the "host portion." Those IP addresses whose first byte was in the range 128-191 were called "Class B" networks of 65,534 (2^{16}-2) hosts, and there were 16,384 (that's (192-128)*256) of them. That's 16 bits for the network portion, and 16 bits for the host portion. "Class C" networks, where the first byte is in the range 192-223, have a 24 bit network portion, and an 8 bit host portion. Note how neatly everything lines up on byte boundaries:

Class	Network bits	Network Mask	Network Mask (binary)
A	8	255.0.0.0	b11111111.00000000.00000000.00000000
B	16	255.255.0.0	b11111111.11111111.00000000.00000000
C	24	255.255.255.0	b11111111.11111111.11111111.00000000

Now, since it's unlikely that a network administratior is going to want to have some 16,777,214 (nearly seventeen million) hosts on the same network segment(!), network administators were allowed to administratively split up their networks by subnetting them. Routing on the Internet backbones was fairly simple... until they started to hit the Class C networks hard. If your company needed 1,000 IP addresses, you'd probably get four Class C networks to accomodate them... but that would add four individual routes propagated to every "backbone" router on the Internet! Hence the need to split up networks on other than just byte boundaries.

This is where everything got hard.

It turns out that you can combine four "Class C" networks together into one routing table entry by

using a subnet mask (aka Network Prefix) of 255.255.252.0. But not just any four; as it happens, they must be contiguous, and the third byte of the first network must be a multiple of four (like the number 204 is.) If you want to join eight of them together, the first network must be a multiple of eight (which the number 204 is not.) If you want to join ten networks together... well, you can't. Ten is not a power of two. Funny how everything follows powers of two...

B. Boolean Logic and The Binary "AND"

Named after the nineteenth-century mathematician George Boole, Boolean logic is a form of algebra in which all values are reduced to either TRUE (1) or FALSE (0). All math performed by modern computers is done using Boolean algebra. A few basic operations:

Operation	Result	Examples
AND	true if A *AND* B are true	1 AND 1 = 1 1 AND 0 = 0 0 AND 1 = 0 0 AND 0 = 0
OR	true if A *OR* B are true	1 OR 1 = 1 1 OR 0 = 1 0 OR 1 = 1 0 OR 0 = 0
XOR (eXclusive Or)	true if *either* A *or* B are true	1 XOR 1 = 0 1 XOR 0 = 1 0 XOR 1 = 1 0 XOR 0 = 0
NOT	opposite of A	NOT 1 = 0 NOT 0 = 1

The binary "and" operation is often used when you want to see only certain bits of a given byte-- a procedure called "masking." Some of you may have seen a similar thing in school; some of my teachers used to conduct multiple-choice tests where you would fill in a circle cooresponding to the answer you thought was correct. The teacher would then take an overlay, or mask, and place it over the answer sheet. This overlay had holes only where the marking spots for the correct answers were, and the teacher would mark any answers where he/she didn't see a mark, as incorrect. The subnet mask is used in this fashion by the computer to determine which address bits are in the network portion of an IP address, and which bits are used for the host, or workstation, portion.

C. The Subnet "Mask"

The subnet mask is used to figure out what network you're on. The reason it's called a "mask" is the same reason the tape you use to cover trim when painting is called "masking tape"; you use it to cover up the parts you don't want to deal with right now. Did you notice how, in a binary AND, any time B is zero, the result is zero? And any time B is one, the result is whatever A is? Hmmm.....

The primary use of the subnet mask (from our perspective at the Near Side of the 'Net) is for workstations to determine whether or not the server or workstation they're trying to talk to (the "destination IP address") is on the same subnet as itself; if the destination IP address is on your subnet, you'll send the IP packet directly to the other computer via the Ethernet or Token Ring (or whatever) network you're on, without bothering the router... at all! **The first routing decision made on an IP packet is made by the workstation sending it; it decides whether or not to send the packet to a router.** Doing this is a four step process:

1. **Step 1:** Convert the IP Addresses to Binary.

If necessary, the IP address is converted from the familiar dotted-decimal into a 32-bit binary value. It sucks as much for the computer to do it as it does for humans to do it, but computers generally complain less, and they're good at math :-)

2. **Step 2:** Apply Source subnet mask to Source addresses:
   The **network portion** of the **workstation's IP address** is determined by performing a binary AND operation on the workstation's IP address and its subnet mask. This operation "masks off" all of the bits of the "host portion" of the IP address, and leaves the "network portion" behind for comparison with the destination's network portion. Hey, wait a minute? How do we know what the subnet mask of the destination is?
3. **Step 3:** Apply **Source** subnet mask to **Destination** addresses:
   As it happens, we don't care what the subnet mask of the destination is. We only care if the destination is on our same network segment! **Since every workstation on our network segment shares the same subnet mask, we can apply our subnet mask to the destination to determine if its network portion matches ours.** So, **the network portion of the destination workstation's IP address that we can use to see if it matches ours** is determined by performing a binary AND operation on the destination IP address and **our subnet mask**.
4. **Step 4:** Compare the derived **network** portions for equality:
   At this point, we can compare the network portions we have masked from the source and destination IP addresses to see if they're the same. If they are, then we **must be on the same subnet** so we send the packet directly; if they are different, even by only one bit, the destination is on another network segment...somewhere. We don't know where. Maybe the router does...

OK, so let's try this a few times ourselves; get a few IP addresses and subnet masks together and plug 'em into Daryl's Subnet Calculator! (The next section of the Primer.) Requires JavaScript to be enabled on your browser. If you're reading a hard copy of this, the full URL is http://ipprimer.windsorcs.com/subnet.cfm .

Remember the part about combining four "Class C" networks together? Watch your binary arithmetic:
(network prefix bits shown in green)

Networks	Networks, in Binary
192.168.8.0	b11000000.10101000.00001000.00000000
192.168.9.0	b11000000.10101000.00001001.00000000
192.168.10.0	b11000000.10101000.00001010.00000000
192.168.11.0	b11000000.10101000.00001011.00000000
Mask, 255.255.252.0	b11111111.11111111.11111100.00000000

Notice how all of the bits above the ones in the subnet mask stay the same; following the rules above, all hosts on these networks, if you apply the mask, are on the same network. This was called "supernetting", but now is called "CIDR Routing", pronounced "Cider Routing".

Doing it wrong:
*(carefully watch the network-portion bit in **red**)*

Networks	Networks, in Binary
192.168.10.0	b11000000.10101000.00001**0**10.00000000
192.168.11.0	b11000000.10101000.00001**0**11.00000000
192.168.12.0	b11000000.10101000.00001**1**00.00000000
192.168.13.0	b11000000.10101000.00001**1**01.00000000
Mask, 255.255.252.0	b11111111.11111111.111111**1**00.00000000

Oops-- seems the sixth bit of the third byte changed within the network prefix portion (the part above the 1's in the subnet mask), so with the given subnet mask (22 bits, or 255.255.252.0), 10.0 and 11.0 would ALWAYS be on a different network aggregation than networks 12.0 and 13.0. Confused? Play with it in the Subnet Calculator, and compare the network portions.

D. "Slash" Notation

Subnet masks are often abbreviated using a forward slash "/" and the number of "one" bits in the mask. For example, a network 192.168.1.0 with a subnet mask of 255.255.255.0 can be expressed as 192.168.1.0/24 (since 255.255.255.0 is 24 binary ones followed by eight binary zeros.) Therefore, a /25 subnet is a subnet with a mask of 255.255.255.128, and a /26 subnet has a mask of 255.255.255.192, etc.

E. A Neat Trick

Now that you actually understand the binary arithmetic behind subnet masking (well, I hope you do, anyway) we can cover some of the neat tricks for computing subnet masks. To determine the number of hosts on a given subnet (assuming the subnet is smaller than class "C",) simply subtract the last number of the subnet mask from 256. For example, a subnet mask of 255.255.255.224 has 32 hosts (256-224=32.) Then you can just divide the result into 256 to determine the number of subnets (256/32=8.) So, using a subnet mask of 255.255.255.224 gives you 8 subnets of 32 hosts each. Of course, this only works when you are subtracting a number that is a power of two (1, 2, 4, 8, 16, 32, 64, or 128.) When the network prefix is larger than class "C", you can determine how many class "C" networks are aggregated by subtracting the third byte from 256-- so a network prefix of 255.255.240.0 is an aggregation of (256-240) 16 class "C" networks.
Thanks to Gael M. for this tip.

F. In closing...

Why all this crap about binary arithmetic? Do I *have* to know this stuff? I'm afraid so; subnet masks are created and used on a bit-by-bit basis; in order to effectively use subnet masks that *don't* fall on byte boundaries (like 255.255.255.0 does), you have to determine what hosts are on each subnet by using binary arithmetic. It sucks, it's hard, it's confusing (espically since IP addresses and masks are expressed in decimal instead of hexadecimal notation) but you must use and understand IP addresses and subnet masks as binary.

---

8. Daryl's Subnet Calculator

[Old, non-DHTML version]

Enter IP Address: Enter Subnet Mask: < > Enter Destination Address: Calc	**Step 1:** Convert all values to **Binary:** a. IP Address: *Enter a valid IP address* b. Subnet Mask: *Enter a valid network prefix* c. Destination Address: *Enter a valid IP address*
Step 2: Apply Source subnet mask to **Source** address: a. Address (from 1a): *Enter a valid source IP address* Mask (from 1b): *Enter a valid Network Prefix*	**Step 3:** Apply **Source** subnet mask to **Destination** address: Address (from 1c): *Enter a valid destination IP address* Mask (from

= Src Network: *Enter a valid source IP address and mask* = Src Network:	1b): *Enter a valid Network Prefix* = Dst Network (?): *Enter a valid source mask and dest IP address* This *might* be the destination network.
Step 4: Compare the **network portions** for equality: Source Net: *Enter a valid source IP address and mask* Destination Net (?): *Enter a valid source mask and dest IP address* ***Are they the same?***	**Result:** The result will display here if you have JavaScript enabled.

Ever get one of those spam e-mails where the IP address is an integer? Type it in here and it will be decoded:

Translate

Excercises:

- Enter your IP address and your default router's IP address. Are you on the same subnetwork?
- Compare your IP address and the IP address of your DNS server(s). Are you on the same network as your DNS server(s)?
- Remember, this principle is used for routing, too. Watch what happens when you increment and decrement the number of bits in the subnet mask (use the arrow buttons).
- If your ISP has assigned your organization multiple contiguous subnets, see if you can determine the network prefix they use for routing to you by entering the top and bottom IP addresses you own, then widening the network mask until they are on the same subnet!

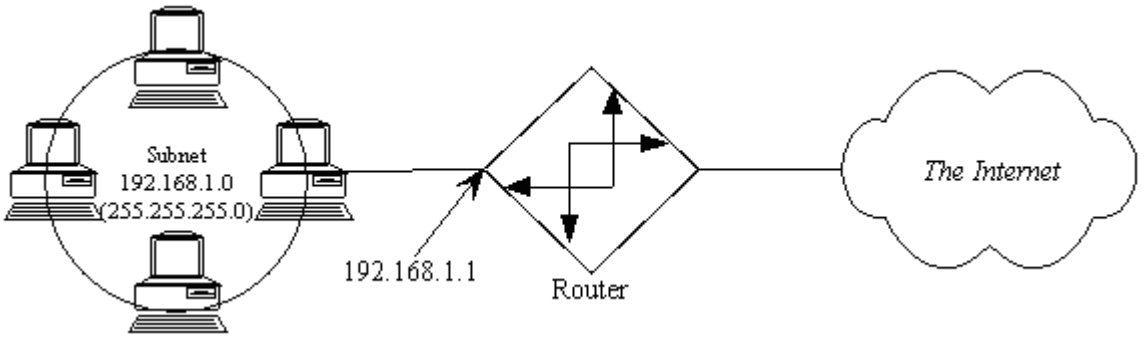
I've also noticed a freeware Win32 app that does similar things at http://www.net3group.com/ipcalc.html-ssi.

---

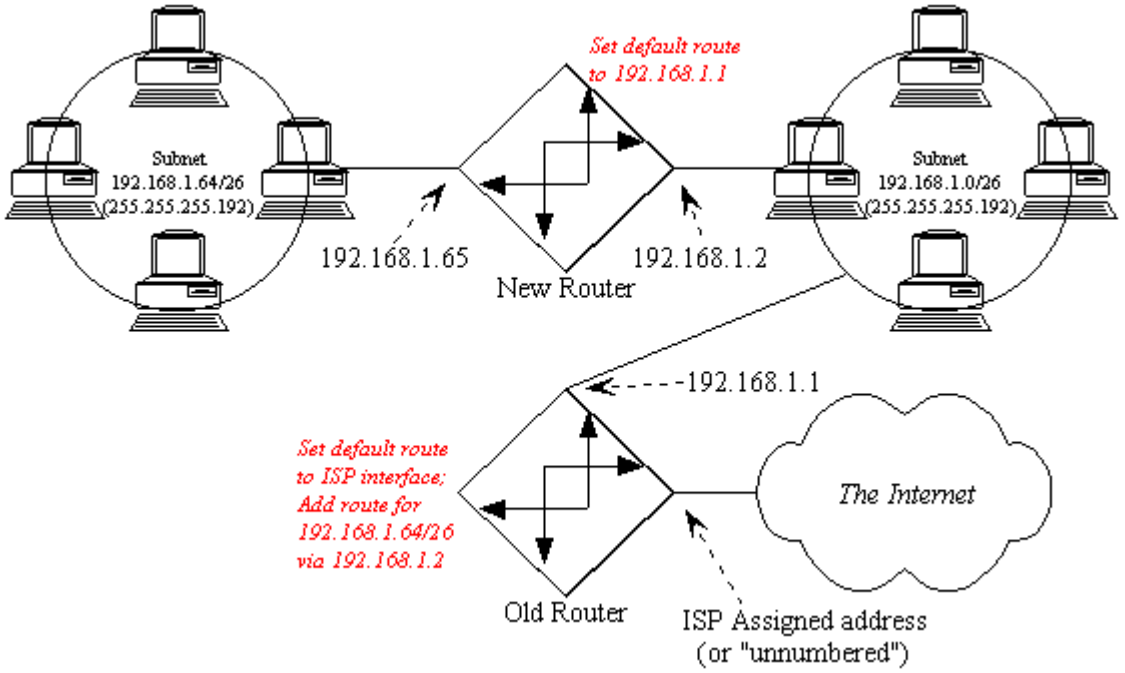
9. Routing and Static Routes

I'm not going to go into a ton of detail here. Instead, I'm going to offer a single example of a network split into two halves.

Before: Network 192.168.1.0:



After: Split into three parts using a subnet mask of 255.255.255.192



These images created using SmartDraw. Click Here for a free trial copy.

What we need to do now is tell the router what happened...

First, you have to tell the old router that the network attached to its Ethernet interface has changed (specifically, the network mask has changed, and often, the address of the Ethernet interface has changed.) If you were adding a new subnet, rather than splitting an existing one, then you could probably skip this step.

Second, you have to tell the old router where to find the new network (what the next hop is.) A typical command would look something like this:

ROUTE 192.168.1.64/255.255.255.192 192.168.1.2

What you're telling the old router with that statement is, "if you need to route packets to the subnetwork that starts at 192.168.1.64 and has a subnet mask of 255.255.255.192, you should forward all packets intended for that network to the router at 192.168.1.2."

Third, be sure the default route for the new router is set to 192.168.1.1.

Note that the automatic routing protocol (IP) RIP does not understand subnet masking. If you are using protocols that do, such as OSPF or EIGRP, then you probably aren't reading this document.

Actually using routing protocols tends to be irrelevant on the "near side" of the net, since there is generally only one path to the Internet from any given workstation on a LAN. Multiple routes tend to be a problem only closer to the backbone, and that's your ISP's problem.

---

10. Troubleshooting

The second most useful tool in troubleshooting client IP issues is PING. Ping is a low-level method of determining is a specific host is alive.

Step #1: Determine if the IP stack is alive. There is a reserved address 127.0.0.1 called "localhost". A successful ping to 127.0.0.1 means your IP stack is working properly. A ping to localhost doesn't even make it on the wire.

Step #2: Determine if you can talk onto the wire. Ping yourself. If your address is 192.168.1.1, then ping 192.168.1.1. Actually, the packet may or may not actually make it on the wire, depending on your implementation. But it doesn't hurt.

Step #3: See if you can ping anyone else. Ping your default router. Make sure your default router is on your same subnet! The easy way to do this is to refer to the "glossy explanation" of subnetting in Section 4, and to make sure both addresses *can* exist in the same subnet. If you can't ping your default router, either the router is down (easily checked from another workstation) or there's something wrong at your workstation. Make sure your workstation has the subnet mask set correctly, and that you and the router are using the same frame type. The default frame type for TCP/IP is Ethernet_II on Ethernet LANs, and TOKEN-RING_SNAP on Token-Ring LANs. Cisco routers refer to Ethernet_II as encapsulation type ARPA.

Step #4: See if you ping the far interface of the default router. All routers have more than one interface (or they wouldn't be routers, right?) If you know the interface of the far side of the router, ping that. That verifies that your default route is set properly. If you don't know the address of another router interface, skip to step 5.

Step #5: Ping the address of your name server. Your name server address is given to you by your ISP. If you cannot ping your name server, try to trace your route to it. The UNIX version of the command is "traceroute". The Win95/WinNT version is called "tracert". An example:

```
D:\WINDOWS>tracert ns.orbis.net

Tracing route to ns.orbis.net [205.164.72.2]
over a maximum of 30 hops:

1 1 ms 1 ms 1 ms 192.168.1.254
2 60 ms 61 ms 64 ms 205.164.75.1
3 64 ms 62 ms 65 ms tamino.summit-ops.orbis.net [205.164.72.129]
4 78 ms 77 ms 78 ms ns.orbis.net [205.164.72.2]

Trace complete.

D:\WINDOWS>
```

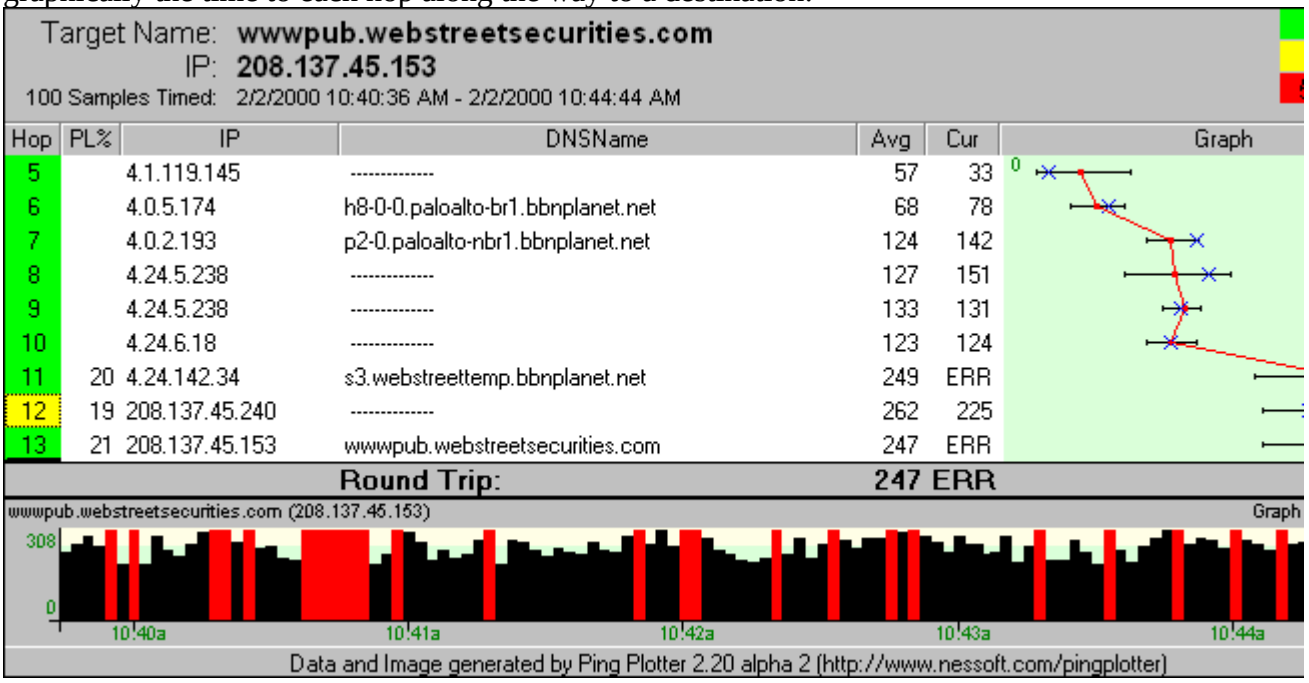
Note: if you actually get names, you not only have verified Internet connectivity, but you also know your DNS is properly set up. Congratulations! You are on the Internet. If you have problems at this point, it's time to call your ISP.

Step #6: If you didn't get any names in your route trace, don't panic: Try to ping www.novell.com or

www.microsoft.com. If you can ping, by name, either of those addresses, you are set up for Internet access. If you get a message like, "Unable to resolve novell.com" then you need to make sure your DNS is set up properly. If you get a "host unreachable" then you probably are set up OK but the 'net is just a bit congested. (Or you haven't set your workstation's default route properly.)

Typically, I start with step #6, and if that fails, go to step #1.

*Second most useful? Probably the most useful tool for diagnosing connection problems across the Internet is traceroute (or tracert for Windows users.) My absolute favorite utility, and the first program I run when I'm having a problem, is Ping Plotter, which is a GUI traceroute tool that shows graphically the time to each hop along the way to a destination:



Ping Plotter is available at http://www.pingplotter.com/ Both "freeware" and registered ($15) versions are available. And if you end up using it more than once or twice a week, do the Right Thing and register it. It's a more than reasonable price, and maybe then Pete will release enhanced versions on a more regular basis :-)

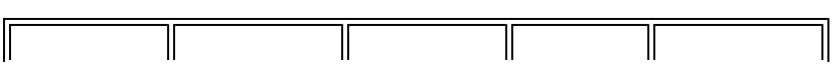
---

11. TCP and UDP Communication

TCP and UDP are layer 4 protocols that help organize process-to-process communication. When a Web browser establishes a connection to download an HTML document from www.mydomain.com, the browser

1. Resolves the IP address for www.mydomain.com
2. Opens a TCP connection to port 80 on the web server www.mydomain.com
3. Transfers the data over the TCP connection
4. Closes the TCP connection

Every TCP (or UDP) communication has a source port and destination port number in the TCP (or UDP) header. **Every TCP/IP communication can be uniquely identified as [Source IP]:[Source Port] <---> [Dest. IP]:[Dest Port].** This is how a Web browser can load several images at once and keep track of which packet is for which image. The source port is different for each TCP image-download connection, though the destination port is 80 in each case. For example:



Source IP	Source Port	Dest IP	Dest Port	Notes
192.168.1.1	1025	10.101.10.1	80	index.html
192.168.1.1	1026	10.101.10.1	80	logo.gif
192.168.1.1	1027	10.101.10.1	80	backgrnd.gif

Note that each file getting downloaded has a different source port number; this is how the communications are differentiated (this packet is part of logo.gif, this packet is part of index.html, etc). Now, let's assume that index.html is finished, but the graphics are loading slowly. While the user is waiting, he/she decides to open a telnet session to rs.internic.net. The table of open sessions would look like this:

Source IP	Source Port	Dest IP	Dest Port	Notes
192.168.1.1	1026	10.101.10.1	80	logo.gif
192.168.1.1	1027	10.101.10.1	80	backgrnd.gif
192.168.1.1	1028	198.41.0.9	23	telnet rs.internic.net

Now, I could go into exhaustive detail on how a TCP connection is set up and torn down, flow is controlled, and dropped packets are resent. Instead, I'll just say that TCP connections are set up and torn down, and there is flow control and automatic dropped packet redelivery. TCP is like certified mail; if no return receipt is gotten, the packet is resent (I'm oversimplifying but it's close enough.) TCP is used for "reliable" communications, where all data must get through, and must get there in the correct order.

A UDP packet, on the other hand, is more like junk mail. No effort is expended to make sure it arrives at the destination, or that all packets arrived that were sent. UDP is generally used for real-time applications like Internet radio and online gaming, where dropped packets need not be resent, and would probably be old if they were. UDP is also used when upper-layer protocols do their own flow control and data stream checking and correcting, as is the case in NCP/IP (Netware/IP) and SMB/IP (Microsoft Networking).

Web, Telnet, Mail, and other servers "listen" for new communications at "well-known" TCP port numbers. A short list:

Service	"Well-Known" Port Number
FTP	21&20 (don't ask)
Telnet	23
SMTP Mail	25
HTTP (Web)	80
POP3 Mail	110
News	119
IRC	6667
QuakeWorld :-)	27500

Publicly available services are generally *always* reached by connecting to their well-known port numbers.

A more complete list of assigned Well-Known Ports can be found at http://www.isi.edu/in-notes/iana/assignments/port-numbers

A list of common vulnerabilities by port can be found at http://www.networkice.com/advice/Exploits/Ports/. Incidentally, the maintainers of this list produce a "personal firewall" product for Windows systems called BlackICE Defender http://www.networkice.com/html/blackice_defender.html that I use and recommend highly for servers. For personal use, look at ZoneAlarm (http://www.zonealarm.com/), which is effective bidirectionally, but requires too much user intervention to be used for servers in a lights-out environment. And you can't beat the price. :-)
(Thanks to B.B. for noticing the broken links and suggesting these new ones..!)

---

12. Network Address Translation (NAT)

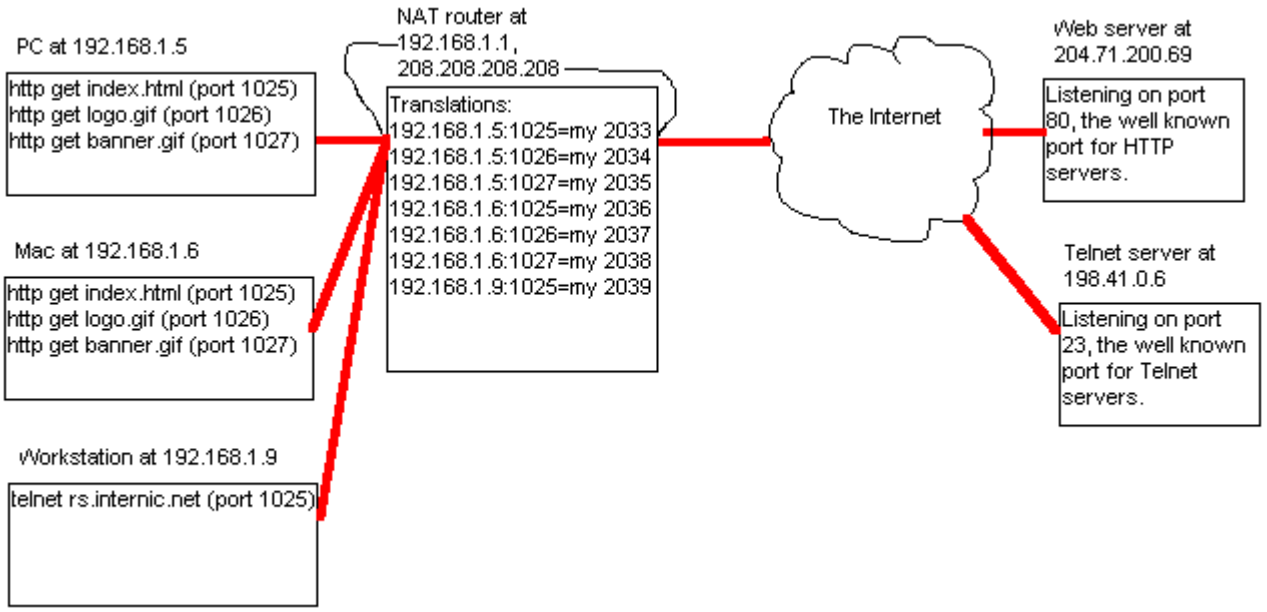
Network Address Translation, or NAT, is accomplished using software that can hide one or more subnets behind a single IP address. NAT software is typically found in newer Internet routers and almost always used in firewalls and proxy servers. NAT is not the same as an HTTP Proxy server. HTTP Proxy servers must be configured on the client side. Once configured, your Web browser asks the HTTP proxy to make connections to the Internet on your behalf; as far as the Web site you're connecting to knows, it's the proxy server that's reading the Web page, not your browser. NAT *is* an effect of the HTTP proxy in this case; the requests from all of the browsers using the HTTP proxy appear to be coming from the proxy server, not from the workstation. The workstation does not need to be using IP addresses that are routable to the Internet; in fact, it is normal to use addresses that are reserved for this purpose, such as 10.x.x.x (*see* Tips and Tricks, later in this document.)
"Transparent" NAT is easier to implement (since nothing needs to be changed at the workstations). However, "configured" NAT (e.g., HTTP proxy servers) often add additional features, such as Web page caching.

NAT software accomplishes three basic things:

- NAT can allow you to connect many more machines to the Internet than you have IP addresses for. I first used NAT to connect my home LAN to a dial-up ISP via a single-IP-address PPP connection. (I used the reserved address block 192.168.1.0/24 on my LAN).
- NAT is a good security measure when you use reserved addresses behind the NAT router, since the addresses are not globally routable. It is harder to attack hosts when you can't reach them directly.
- NAT is a good security measure because *no inbound connections are allowed* through the NAT translator unless it is specifically configured to allow them; as we will see, this is a side effect of using NAT software.

I like to refer to NAT routers as "transparent TCP proxy routers." Transparent, because unlike HTTP proxies, NAT routers do not need any configuration nor application software support to work with most TCP-based protocols. NAT routers will proxy outbound connections "automagically."

For every outbound TCP connection, the NAT router intercepts and creates its own TCP connection to the destination host. The NAT router builds a growing list of port translations. Consider two computers that open three TCP connections each to a web server to download the same Web page. At the same time, a Linux workstation opens a Telnet session to rs.internic.net:

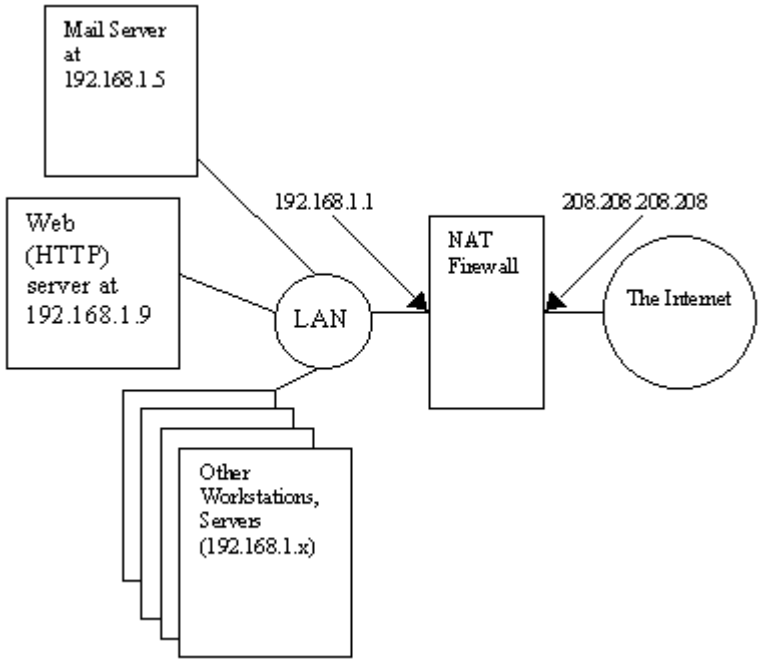


The web server thinks the NAT router at 208.208.208.208 has two browsers running that both just opened the same document and images; the Telnet server thinks that the same computer at 208.208.208.208 opened a Telnet session to it; only the NAT software knows that three computers have seven connections open from behind it.

Transparent NAT works well for TCP connections, but due to the connectionless nature of UDP, NAT works less well for unusual UDP connections (sorry, Quake fans..!)

Since NAT routers *are* hiding many machines behind a single IP address, putting server(s) behind a NAT router becomes a problem, since the NAT software has no way of determining for itself what IP address to forward the inbound connection requests to. This dropping of inbound connections, while allowing outbound connections, makes NAT routers into cost-effective low-end firewalls. Though NAT routers do nothing to prevent users from downloading viruses or trojan horse programs (like the well-publicized trojan horse Back Orifice), but does go a long way toward blocking attempts to connect inbound to the running trojan horse, if accidentally or maliciously installed.

If your NAT router only supports one "real" IP address, you can only have one service on your network listening on the "well known port" for that service; you could have two Web servers listening on different ports, but not two web servers both listening on (e.g.) 208.208.208.208:80. For example, you have a LAN configured as follows:



This image created using SmartDraw. Click Here for a free trial copy.

You would configure the NAT software to listen to ports 25 and 80 on 208.208.208.208, and forward connections as follows:

"Listening" Port	"Internal" Address
208.208.208.208:25	192.168.1.5:25
208.208.208.208:80	192.168.1.9:80

If you want to play with NAT software, and you have an old PC-compatible machine lying around (NAT is easy for routers to do and does not require much in the way of hardware), Look at two Linux-based standalone router efforts: FreeSCO at http://www.freesco.org/, or the Linux Router Project (http://linuxrouter.org/ Please, read the manual and experiment a bit with the software before sending me questions specific to FreeSCO or LRP. You'll learn more that way, and I didn't help write the program, so I probably shouldn't be the person you talk to for tech support about it, anyway. :-)

Platform Specific Infomation: Note that TCP/IP proxies are not platform-specific. In other words, it works fine to place a MS-DOS based proxy server (such as IPRoute) on a Mac network, or a Linux proxy on a Novell-based IP network. But if you only want to add software, not hardware, to your network, then here are some options I've found. (Note: I do not explicitly endorse the use of any of these products, they're merely listed here for your convenience.)

- **Netware networks:** I understand Border Manager does address translation. Novell's IPX-to-IP gateway (a different product) works for IPX-only networks by tunneling IP sockets through the IPX network to the Netware server, which makes the "real" TCP/IP connection to the destination server. Workstations are protected by virtue of the fact that they're not actually running TCP/IP locally, and don't have IP addresses of their own (they all use the server's IP address).
- **NT networks:** Microsoft Proxy Server does HTTP proxying, and I'm told newer versions do full NAT, but other solutions to consider are WinRoute (http://www.winroute.com) and Sygate (http://www.sygate.com) for NAT and HTTP proxy services. I've used both. Winroute has more features, but Sygate is a bit less expensive. I'm currently using Winroute, since it's quite a bit easier to manage the packet filtering and address translation. And they've improved their Quake support substantially :-)
- **Macintosh networks:** I rarely work with Mac networks. A Mac-based NAT router is available

from VicomSoft (http://www.vicomsoft.com/softrouter/sfr.main.html), but I have no experience with it.

- **Linux:** Vaguely recent (v1.3.x or greater) Linux kernels include support for "IP Masquerading," which is its name for network address translation. There is a newer kernel option called "transparent proxy" which is not NAT, but rather *forces* all outbound connections to use a proxy server, without the user's knowledge or explicit configation. Linux goes a step further with Masquerade Loadable Modules (the link, http://ipmasq.home.ml.org, is down at the time of this writing; it will probably reappear at a new address), which can explicitly support "wierd" connections such as the seperate UDP connections Quake servers use. See any of the many Linux IP Masquerade sites, such as http://www.indyramp.com/masq/
  New: Check out the Linux Router Project at http://linuxrouter.org/. A full-featured Open Source Linux-based router product that can boot off a floppy disk.

---

13. The Domain Name System (DNS)

The Domain Name System, or DNS, is a service that translates computer names into IP addresses. A name-to-address system is necessary because we humans do not easily remember numbers like, "207.68.156.61", but we can easily remember names like, "www.microsoft.com". The DNS is a hierarchical system, with the top of the system called the "root", and represented by a single period ".". There are twelve (very, very busy) "root" servers on the Internet at the time of this writing. Root servers know where the servers are for the "top-level domains" like .com, .net, .edu, .org, .uk, .de, .nz, .us, and so on.

Let's start with an example: If you ask your local name server for the address of "www.north-america.example.com" the name server will do the following:

1. Check to see if it already knows the address of "www.north-america.example.com" (let's assume it doesn't. The example is more interesting that way.)
2. The DNS server queries a "root" server for the address of "www.north-america.example.com". All fully-functional DNS servers are configured with a static list of root servers, available at ftp://ftp.rs.internic.net/domain/named.root.
3. The root server will refer your DNS to a list of ".com" servers.
4. Your DNS will query one of the ".com" servers for the address of "www.north-america.example.com"
5. The ".com" name server queried refers your name server to a list of name servers for "example.com".
6. Your DNS server then asks one of the "example.com" name servers for the address of "www.north-america.example.com".
7. One of two things can happen here. If the "example.com" name server queried knows the address of "www.north-america.example.com" then it returns that address to your DNS server. If the "north-america" subdomain has been delegated to some other name server(s), then that name server list (of name servers that service the zone, "north-america.example.com") will be returned to your DNS, and your DNS will query one of those servers for the address of "www.north-america.example.com".

Note that your DNS remembers, or caches, all the information it retrieves this way. Therefore, if you asked your local DNS for the address of "ftp.north-america.example.com", then it would directly ask the name server finally referenced in step 7 (above) for the address of "ftp.north-america.example.com". This prevents the top-level and root servers from being more heavily loaded than they already are. (It's also interesting to note that the root servers are also the top-level domain servers for the US domains.) It is possible to set up a caching-only DNS server that processes and caches requests, but isn't directly knowledgeable ("authoritative") about any domains itself.

Domains, Zones, and Authority

There are several different types of name servers. There is one Primary name server for each domain or delegated subdomain ("zone"). A "zone" refers to the domain and subdomain(s) (if any) a server is authoritative for. In many cases "zone" and "domain" mean the same thing, but when you start delegating authority for subdomains, they get their own *zone* to administer, although it's part of your *domain*. For example, the root servers are authoritative for the ".com" *zone* but they aren't authoritative for the entire ".com" *domain*. "example.com" is, in fact, a *subdomain* of the ".com" *domain*, but is a different DNS *zone*. Zone boundaries typically follow administrative control boundaries: since the people managing the ".com" domain are not the same as the people managing the "example.com" domain, a new zone is created and authority for the zone is delegated to that zone's name servers.

Every Primary name server should have at least one Secondary name server. A Secondary name server simply copies the zone information from the zone's Primary server. Secondary name servers also answer DNS requests authoritatively. It is *strongly* suggested that at least one secondary name server be on another physical network. If someone wants to send you mail, and your mail server is unreachable, the mail is queued and retried, but eventually delivered. If the sending mail server is told there is no mail server or host information about your network (which is what happens if all authoritative DNSes are unreachable) then the mail bounces.

If you set up a Primary name server, it is necessary to have the parent domain delegate authority for your zone to you. For example, if you wanted to be the authoritative name server for the domain "reallyslow.net", you would have to ask the administrators for ".net" (InterNIC, in this case) to delegate the zone authority for "reallyslow.net" to you. Similarly, if the engineering department wanted to run there own name server for "engineering.reallyslow.net", then they would have to ask you to delegate the zone "engineering.reallyslow.net" to their name server(s).

It is usually possible to look up an address and come up with a machine name. This is called a "reverse lookup," because instead of getting an address from a name, you are getting a name from an address. The reverse lookup system behaves very similarly to "normal" DNS; in fact, you could almost consider it to be a parallel DNS system. Lookup is done in reverse order by octet with the domain "in-addr.arpa" appended. Let's say you "own" a network 192.168.45.0 with a subnet mask of 255.255.255.0. You would contact the administrator for "168.192.in-addr.arpa" and ask him/her to delegate the authority for the zone "45.168.192.in-addr.arpa" to your name server. On your name server you would create a zone file for reverse lookups that would be authoritative for that zone.

Types of DNS Records

SOA: A Start of Authority record is used at the top of every zone file to indicate the zone that the file is authoritative for. The SOA record also contains administrative contact information, the serial number for the file (which must be incremented whenever the file is updated), and various default timeout and retry values for the domain.

```
reallyslow.net.         IN SOA  turtle.reallyslow.net root.reallyslow.net ([vario
```

A: Address records actually provide name-to-address mapping:

```
turtle.reallyslow.net.       IN A       192.168.45.10
caterpillar.reallyslow.net.  IN A       192.168.45.12
```

CNAME: Canonical name records are "alias" records that are often used to map conventional names like "www.reallyslow.net" to the actual name ("A" record) of the computer providing World Wide Web services for the domain. Other names use by convention include "ftp." for ftp services, "mail."

for e-mail servers, and "ns" for name servers.

```
www.reallyslow.net.          IN CNAME   turtle.reallyslow.net.
snail.reallyslow.net.        IN CNAME   caterpillar.reallyslow.net.
```

NS: Name Server records indicate which machines are used as name servers. NS records sometimes point to host names ("A" records), sometimes point to aliases ("CNAME" records), and sometimes just list an IP address.

```
reallyslow.net.               IN NS      turtle.reallyslow.net.
reallyslow.net.               IN NS      snail.reallyslow.net.
```

MX: Mail eXchanger records indicate which machines are mail servers for a domain and what their preference is. The lower the number, the higher the preference (hey, I didn't invent it.) Other mail servers will try to send mail to the highest preference mail server first. We want email for anyone@reallyslow.net to be delivered to the machine mail.reallyslow.net:

```
reallyslow.net               IN MX 10   mail.reallyslow.net.
```

or, if you used another company to handle your email services...

```
reallyslow.net               IN MX 10   mail.notquitesoslow.net.
```

MX records should not point to CNAME records.

PTR: Reverse lookup pointers are used by the reverse lookup system to map addresses to names (notice the reversed order of the octets:)

```
10.45.168.192.in-addr.arpa.  IN PTR     turtle.reallyslow.net.
12.45.168.192.in-addr.arpa.  IN PTR     caterpillar.reallyslow.net.
```

Note that host names never include "@" symbols. An "@" symbol *almost* always indicates an email address [one notable exception being a message identifier for a Usenet newsgroup posting. Tks again B.B.] The name to the right of the "@" sign is queried for an MX record and mail is delivered to the machine indicated by the MX record(s). In a DNS file, the "@" symbol is a placeholder used to represent "the current domain" as it was named in named.boot. named.boot is the standard file name used by DNS ("named", pronounced "name dee") servers. A basic named.boot looks like this:

```
primary reallyslow.net db.reallyslow.net
primary 0.0.127.IN-ADDR.ARPA db.127.0.0
primary 45.168.192.in-addr.arpa  db.inaddr
```

We're telling BIND that it is authoritative for the "standard" zone "reallyslow.net", and also primary for the reverse lookup zones for the subnets 192.168.45.x and 127.0.0.x. (The only entry for 127.0.0.x is 127.0.0.1, which maps to LOCALHOST, which is a reserved address and name for "this machine". In other words, you will always have a VERY fast ping to localhost :-) The zone file for 45.168.192.in-addr.arpa contains standard PTR records after the SOA record. Note that it's really easy to forget to update named.boot if you add a new domain to your name server (hint, hint.)

If you are going to set up your own name server, I *highly* recommend the book DNS and BIND by Paul Albitz and Cricket Liu (O'Reilly & Associates, ISBN 1-56592-010-4). On the 'net, check out the "BIND Operations Guide" in Windows Write format at ftp://ftp.software.com/BIND-NT/BOG.wri.

---

14. Tips for Building an IP LAN

The part you were waiting for, right?

- First, if you skipped ahead to this section, go back and read the previous sections. A collection of tips will not replace knowing what the heck you're doing.
- If you're not connected to the Internet, and don't already have one or more IP networks assigned to you, use the addresses reserved for this purpose. They are, 10.x.x.x, 172.16.x.x-172.31.x.x, and 192.168.x.x (*see* RFC 1597)
- Create a subnet address policy (e.g., .1-.5 reserved for routers, .1 always the default route, .6-.30 reserved for static IP's such as servers, .50-.254 dynamic through bootp/dhcp.)
- Use DHCP or BOOTP to assign workstation addresses. When it comes time to change (after your network has grown a bit), you'll thank me.
- Meticulously track static IP assignments. Create a central database or document listing all static IP's and their associated devices.
- Label router interfaces with their addresses.
- Keep a **current** diagram of your subnets and router connections (include detail on router interface addresses.) If you get into trouble, it'll save you two hours of onsite time if you have to call someone in to help. Personally, I use SmartDraw for this purpose (http://www.smartdraw.com/.) Although Visio also works well, SmartDraw (IMHO) is easier to use and the price is right.
- If you have IP-enabled servers, use a firewall. If you are using Windows-based file sharing and have no firewall, use a non-IP protocol to do it (IPX or NetBEUI). You will then need to set either IPX or NetBEUI as your default protocol. Or, get a firewall. IPRoute from Dave Mischler (http://www.mischler.com, $50US) can be used as an effective, low cost firewall, and it'll run on any '386 or better PC with two NICs; or, if you want to experiment with putting your LAN on the Internet, a low cost and very secure way to do this is with IPRoute and an old '386 PC with a good serial port chip and modem.
- [Windows Only] If a firewall is not an option, look at BlackICE Defender http://www.networkice.com/html/blackice_defender.html, which I use and recommend highly for servers. For personal use, look at ZoneAlarm (http://www.zonealarm.com/), which is effective bidirectionally, but requires too much user intervention to be used for servers in a lights-out environment. And you can't beat the price. :-)
- Check out my drawing of an example LAN here.
- If you want to see your network working, packet by packet, check out Ethereal, an Open Source packet sniffer: http://www.ethereal.com/.
- More to come / Your Tip Here.

---

15. Packet Analysis

While talking about the theory of TCP communications is helpful, nothing beats a wire's eye view of actual IP communication.

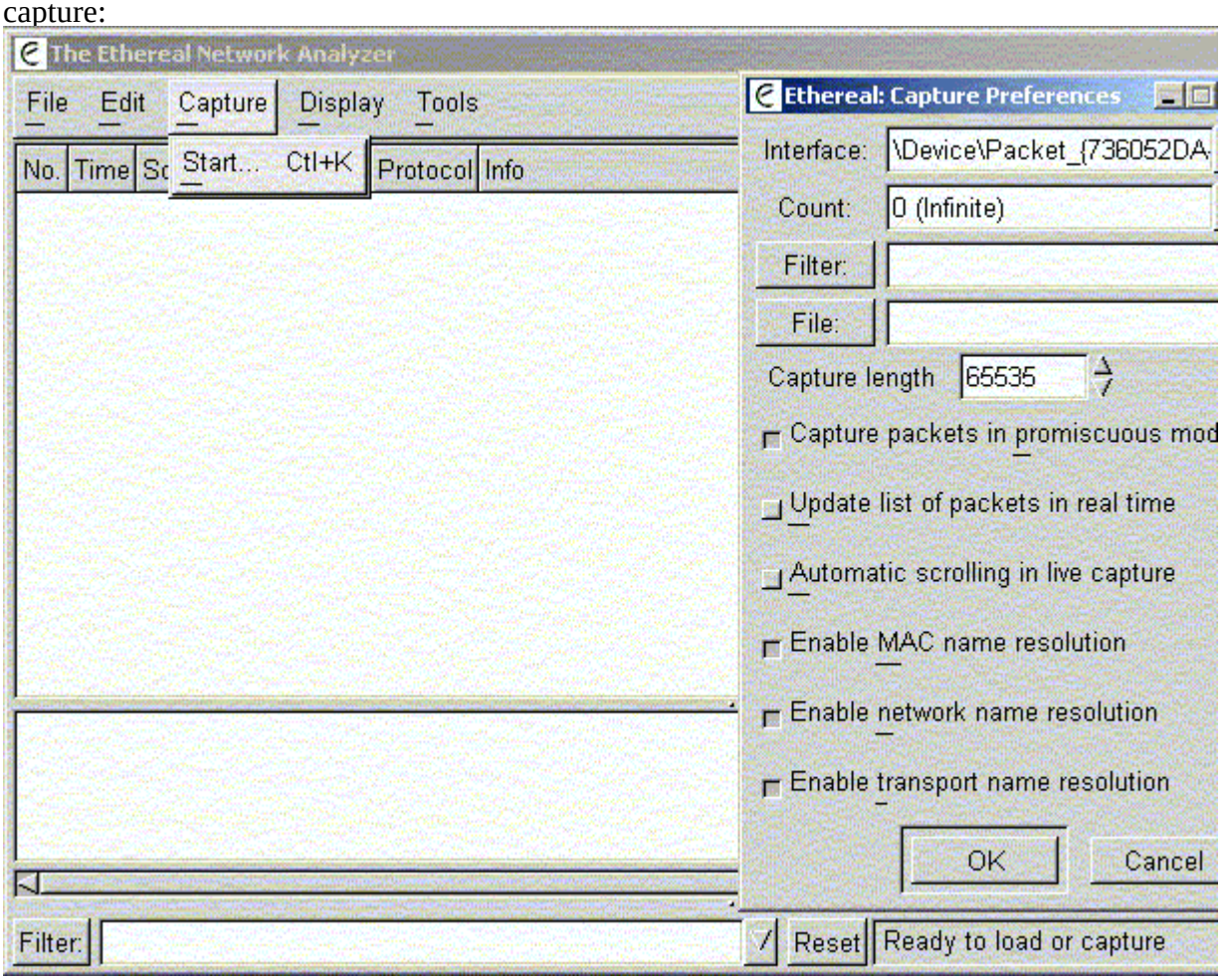
If you plan to have a career in this field, then get used to packet sniffing, and learn to "read" packets and TCP communications. Also, learn how to read RFCs, and practice correlating communications captured from the wire against the protocols described in the RFCs. This way, you'll be able to debug system problems that hide from everyone else involved. What better way to achieve job security, than to make yourself indispensable to the organization?

The tool I'll use for this demonstration is Ethereal, an open-source packet analyzer. (I'd call it a Packet Sniffer, but Network Associates never tires of pointing out that they have a trademark for the word "Sniffer", when used in that context.) **I encourage you to download and install Ethereal on your computer, so you can follow along.** I'll be using the Windows version, since most of the people that access Daryl's TCP/IP Primer do so from Windows-based machines.

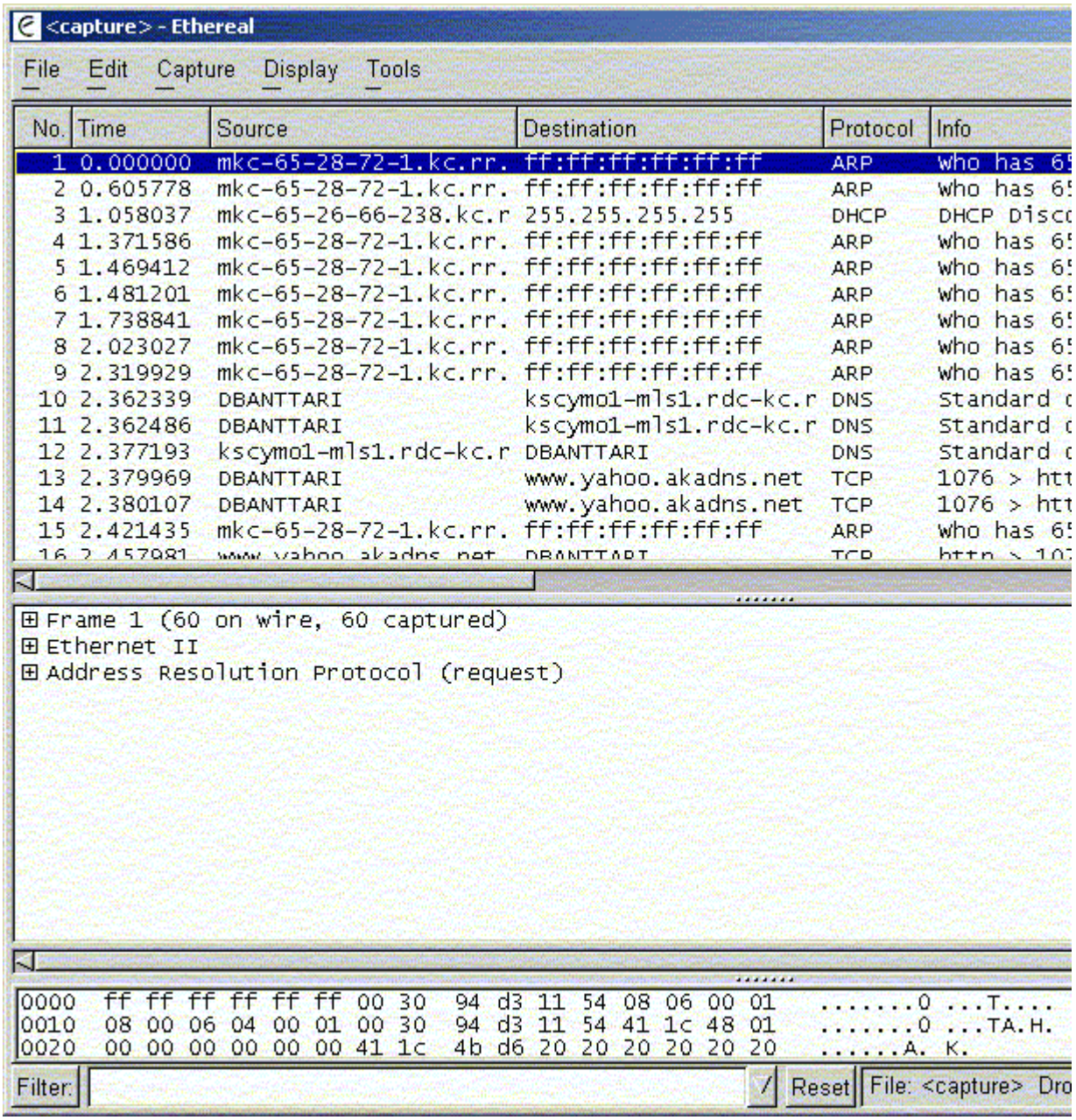
First, download and install Ethereal. the last time I checked, the two parts needed are a low-level packet driver, available at http://netgroup-serv.polito.it/winpcap/install/default.htm, and the Ethereal package itself, available at http://www.ethereal.com/download.html#binaries. Install both, in order. (The main page for Ethereal, by the way, is http://www.ethereal.com/)

We are going to capture the packets sent and received as we load the front page of Yahoo.com, then decode and view the communication.

1. Under the Capture menu, choose Start (or, just press Ctrl-K). Then click OK to begin the capture:



2. Click on this link, which will open a new window: http://www.yahoo.com/ (target=_blank)
3. Now stop the packet capture by clicking on the Stop button. For some reason, on my machine, the software takes an inappropriately long time to load the packet display, which I'm sure will be corrected in a version after 0.8.19...
4. Now review the packets you captured. If you're on a shared-media network (such as Ethernet or a cable modem) you may see many packets unrelated to your current communication:



What you can see in my capture is the cable network's router, at 65.28.72.1, asking for the MAC address of several other machines on the network. It would seem that the cable network simulates an Ethernet network, since the ARP packets have Ethernet II headers (as seen in the middle window.)

If we skip past the ARP and DHCP broadcast noise, we see the first packet related to our communication, a DNS lookup, needed to map "www.yahoo.com" to an IP address (partially expanded):

```
Frame 10 (62 on wire, 62 captured)
Ethernet II
    Destination: 00:30:94:d3:11:54 (mkc-65-26-66-1.kc.rr.com)
    Source: 00:04:75:19:dd:be (00:04:75:19:dd:be)
    Type: IP (0x0800)
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: kscymo1-mls1.rdc
User Datagram Protocol, Src Port: 1075 (1075), Dst Port: domain (53)
Domain Name System (query)
    Transaction ID: 0x000d
    Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
```

```
    Authority RRs: 0
    Additional RRs: 0
    Queries
        www.yahoo.com: type A, class inet
```

Notice that the destination Ethernet address is that of the router, while the destination IP is beyond the router. Each hop is really only concerned with finding the correct device to handle the next hop, then passing the packet to that device by whatever layer 2 protocol is available. So, my machine is simply passing the packet to the router via Ethernet. I assume the router will then pass it to the next router that is close to the DNS server, and so forth until the DNS server is reached.

The response packet lists all A records associated:

```
Frame 13 (62 on wire, 62 captured)
Ethernet II
    Destination: 00:04:75:19:dd:be (00:04:75:19:dd:be)
    Source: 00:30:94:d3:11:54 (mkc-65-26-66-1.kc.rr.com)
    Type: IP (0x0800)
Internet Protocol, Src Addr: kscymo1-mls1.rdc-kc.rr.com (24.94.163.165), Dst Addr
User Datagram Protocol, Src Port: domain (53), Dst Port: 1077 (1077)
Domain Name System (response)
    Transaction ID: 0x0009
    Flags: 0x8180 (Standard query response, No error)
    Questions: 1
    Answer RRs: 13
    Authority RRs: 8
    Additional RRs: 7
    Queries
        www.yahoo.com: type A, class inet
            Name: www.yahoo.com
            Type: Host address
            Class: inet
    Answers
        www.yahoo.com: type CNAME, class inet, cname www.yahoo.akadns.net
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.223
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.228
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.224
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.177
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.178
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.222
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.225
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.229
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.179
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.176
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.226
        www.yahoo.akadns.net: type A, class inet, addr 64.58.76.227
    Authoritative nameservers
        akadns.net: type NS, class inet, ns ZA.akadns.net
        akadns.net: type NS, class inet, ns ZB.akadns.net
        akadns.net: type NS, class inet, ns ZC.akadns.net
        akadns.net: type NS, class inet, ns ZD.akadns.net
        akadns.net: type NS, class inet, ns ZE.akadns.net
        akadns.net: type NS, class inet, ns ZF.akadns.net
        akadns.net: type NS, class inet, ns ZG.akadns.net
        akadns.net: type NS, class inet, ns ZH.akadns.net
    Additional records
        ZA.akadns.net: type A, class inet, addr 216.32.65.105
        ZB.akadns.net: type A, class inet, addr 216.200.14.118
        ZC.akadns.net: type A, class inet, addr 204.178.107.227
        ZD.akadns.net: type A, class inet, addr 206.132.160.36
        ZE.akadns.net: type A, class inet, addr 12.47.217.11
        ZF.akadns.net: type A, class inet, addr 63.215.198.79
        ZG.akadns.net: type A, class inet, addr 204.248.36.131
```

Looks like Yahoo has no shortage of name servers or IP addresses.

Now that we have an IP address (or 12), we can choose one of them and try to connect, by sending a SYN ("Synchronize") packet to port 80:

```
Frame 14 (62 on wire, 62 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525356
    Header length: 28 bytes
    Flags: 0x0002 (SYN)
    Window size: 16384
    Checksum: 0x1f12 (correct)
    Options: (8 bytes)
        Maximum segment size: 1360 bytes
        NOP
        NOP
        SACK permitted
```

And the response:

```
Frame 16 (60 on wire, 60 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906338462
    Acknowledgement number: 1166525357
    Header length: 24 bytes
    Flags: 0x0012 (SYN, ACK)
    Window size: 17680
    Checksum: 0x57f0 (correct)
    Options: (4 bytes)
        Maximum segment size: 1460 bytes
```

Note that the Synchronize and Acknowledge flags are set on the response.

We can now acknowledge the TCP connection open packet, and send our actual message:

```
Frame 17 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525357
    Acknowledgement number: 906338463
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x6fad)

Frame 18 (329 on wire, 329 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525357
    Next sequence number: 1166525632
    Acknowledgement number: 906338463
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 17680
```

```
    Checksum: 0x17e5 (incorrect, should be 0x4547)
Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Accept: */*\r\n
    Referer: http://www.ipprimer.com/packets.cfm\r\n
    Accept-Language: en-us\r\n
    Accept-Encoding: gzip, deflate\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)\r\n
    Host: www.yahoo.com\r\n
    Connection: Keep-Alive\r\n
    Cookie: B=shyg4f0xg4tmp&b=2&f=v\r\n
    \r\n
```

The Push flag indicates that we're done transmitting; the destination IP stack should send the data to the receiving application without attempting to buffer any more of the communication. Note that UDP does not have PSH nor ACK flags; UDP is not buffered at the transport layer. TCP required 4 packets to get any data to the server; DNS has very short communications, so UDP is used because it has very low overhead. HTTP uses TCP, since the downloads can be very long, and TCP has built-in flow-control, resending of dropped packets, and resequencing of misordered packets.

We will now start receiving response packets:

```
Frame 20 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906338463
    Next sequence number: 906339823
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x4df8 (correct)
Hypertext Transfer Protocol
    HTTP/1.0 200 OK\r\n
    Date: Sun, 14 Oct 2001 03:53:44 GMT\r\n
    Vary: User-Agent\r\n
    Connection: close\r\n
    Content-Type: text/html\r\n
    \r\n
    Data (1242 bytes)

0000  3c 68 74 6d 6c 3e 3c 68 65 61 64 3e 3c 74 69 74   <html><head><tit
0010  6c 65 3e 59 61 68 6f 6f 21 3c 2f 74 69 74 6c 65   le>Yahoo!</title
(blah, blah, blah...)
04c0  3d 79 61 68 6f 6f 66 5f 31 34 25 32 36 73 6f 75   =yahoof_14%26sou
04d0  72 63 65 49 44 3d 79 61 68 6f                     rceID=yaho

Frame 21 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906339823
    Next sequence number: 906341183
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x9248 (correct)
Hypertext Transfer Protocol
```

```
    Data (1360 bytes)

0000  6f 66 5f 31 34 22 20 74 61 72 67 65 74 3d 22 5f   of_14" target="_
0010  74 6f 70 22 3e 3c 69 6d 67 20 77 69 64 74 68 3d   top"><img width=
(blah, blah, blah...)
0530  38 33 3b 0a 3c 61 20 68 72 65 66 3d 72 2f 67 63   83;.<a href=r/gc
0540  3e 47 65 6f 43 69 74 69 65 73 3c 2f 61 3e 20 26   >GeoCities</a> &
```

And we start seeing my acknowledgement packets as well:

```
Frame 22 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906341183
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x63fa)

Frame 23 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906341183
    Next sequence number: 906342543
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x59c0 (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  23 31 38 33 3b 0a 3c 61 20 68 72 65 66 3d 72 2f   #183;.<a href=r/
0010  67 72 3e 47 72 65 65 74 69 6e 67 73 3c 2f 61 3e   gr>Greetings</a>
(blah, blah, blah...)
0530  3b 20 3c 61 20 68 72 65 66 3d 73 2f 32 30 38 35   ; <a href=s/2085
0540  3e 4d 69 63 68 61 65 6c 20 4a 6f 72 64 61 6e 3c   >Michael Jordan<

Frame 24 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906342543
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x5eaa)

Frame 25 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906342543
```

```
    Next sequence number: 906343903
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x49e9 (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  2f 61 3e 3c 62 72 3e 0a 26 6e 62 73 70 3b 20 26   /a><br>.  &
0010  23 31 38 33 3b 20 3c 61 20 68 72 65 66 3d 73 2f   #183; <a href=s/
(blah, blah, blah...)
0530  70 61 64 64 69 6e 67 3d 34 3e 3c 74 72 3e 3c 74   padding=4><tr><t
0540  64 20 76 61 6c 69 67 6e 3d 74 6f 70 20 6e 6f 77   d valign=top now

Frame 26 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906343903
    Next sequence number: 906345263
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x7d26 (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  72 61 70 3e 3c 73 6d 61 6c 6c 3e 3c 66 6f 6e 74   rap><small><font
0010  20 73 69 7a 65 3d 33 20 66 61 63 65 3d 61 72 69    size=3 face=ari
(blah, blah, blah...)
0530  6c 6c 20 43 6f 76 65 72 61 67 65 3c 2f 61 3e 2c   ll Coverage</a>,
0540  0a 3c 61 20 68 72 65 66 3d 72 2f 6e 77 3e 4e 65   .<a href=r/nw>Ne

Frame 27 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906345263
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x540a)

Frame 28 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906345263
    Next sequence number: 906346623
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x8e4b (correct)
Hypertext Transfer Protocol
```

```
    Data (1360 bytes)

0000  77 73 70 61 70 65 72 73 3c 2f 61 3e 2c 0a 3c 61   wspapers</a>,.<a
0010  20 68 72 65 66 3d 72 2f 74 76 3e 54 56 3c 2f 61    href=r/tv>TV</a
(blah, blah, blah...)
0530  30 33 30 33 31 36 32 34 2b 68 74 74 70 3a 2f 2f   03031624+http://
0540  75 73 2e 72 6d 69 2e 79 61 68 6f 6f 2e 63 6f 6d   us.rmi.yahoo.com

Frame 29 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906346623
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x4eba)

Frame 30 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906346623
    Next sequence number: 906347983
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0xf66f (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  2f 72 6d 69 2f 68 74 74 70 3a 2f 2f 77 77 77 2e   /rmi/http://www.
0010  63 6f 6d 70 61 71 2e 63 6f 6d 2f 72 6d 69 2d 66   compaq.com/rmi-f
0020  72 61 6d 65 64 2d 75 72 6c 2f 68 74 74 70 3a 2f   ramed-url/http:/
0030  2f 77 77 77 2e 63 6f 6d 70 61 71 2e 63 6f 6d 2f   /www.compaq.com/
0040  79 61 68 6f 6f 2f 22 3e 3c 69 6d 67 20 73 72 63   yahoo/"><img src
0050  3d 22 68 74 74 70 3a 2f 2f 75 73 2e 61 31 2e 79   ="http://us.a1.y
0060  69 6d 67 2e 63 6f 6d 2f 75 73 2e 79 69 6d 67 2e   img.com/us.yimg.
0070  63 6f 6d 2f 61 2f 63 6f 2f 63 6f 6d 70 61 71 5f   com/a/co/compaq_
0080  63 6f 6d 70 5f 63 6f 72 70 2f 70 6f 77 65 72 65   comp_corp/powere
0090  64 5f 62 79 5f 77 68 69 74 65 5f 39 35 78 33 30   d_by_white_95x30
00a0  2e 67 69 66 22 20 61 6c 74 3d 22 22 20 62 6f 72   .gif" alt="" bor
00b0  64 65 72 3d 22 30 22 20 77 69 64 74 68 3d 22 39   der="0" width="9
00c0  35 22 20 68 65 69 67 68 74 3d 22 33 30 22 3e 3c   5" height="30"><
(blah, blah, blah...)
0530  61 72 63 68 3c 2f 61 3e 3c 2f 73 6d 61 6c 6c 3e   arch</a></small>
0540  3c 2f 74 64 3e 3c 2f 74 72 3e 3c 74 72 3e 3c 74   </td></tr><tr><t
```

In the previous packet, notice the reference to an image from us.al.yimg.com in the HTML data.

```
Frame 31 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906347983
    Next sequence number: 906349343
    Acknowledgement number: 1166525632
```

```
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0xc485 (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  64 20 76 61 6c 69 67 6e 3d 74 6f 70 3e 3c 62 3e   d valign=top><b>
0010  26 6e 62 73 70 3b 26 23 31 38 33 3b 26 6e 62 73    &#183;&nbs
(blah, blah, blah...)
0530  62 6f 72 64 65 72 3d 30 3e 3c 74 72 3e 3c 74 64   border=0><tr><td
0540  20 76 61 6c 69 67 6e 3d 74 6f 70 3e 3c 62 3e 26    valign=top><b>&

Frame 32 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906349343
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x441a)
```

The browser, having noticed the reference to the image at us.al.yimg.com in the HTML, prepares to download the image by initiating a DNS lookup on us.al.yimg.com:

```
Frame 33 (74 on wire, 74 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: kscymo1-mls1.rdc
User Datagram Protocol, Src Port: 1079 (1079), Dst Port: domain (53)
Domain Name System (query)
    Transaction ID: 0x000a
    Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        us.a1.yimg.com: type A, class inet

Frame 34 (434 on wire, 434 captured)
Ethernet II
Internet Protocol, Src Addr: kscymo1-mls1.rdc-kc.rr.com (24.94.163.165), Dst Addr
User Datagram Protocol, Src Port: domain (53), Dst Port: 1079 (1079)
Domain Name System (response)
    Transaction ID: 0x000a
    Flags: 0x8180 (Standard query response, No error)
    Questions: 1
    Answer RRs: 3
    Authority RRs: 9
    Additional RRs: 9
    Queries
        us.a1.yimg.com: type A, class inet
            Name: us.a1.yimg.com
            Type: Host address
            Class: inet
    Answers
        us.a1.yimg.com: type CNAME, class inet, cname a32.g.a.yimg.com
        a32.g.a.yimg.com: type A, class inet, addr 24.94.162.91
        a32.g.a.yimg.com: type A, class inet, addr 24.94.162.90
    Authoritative nameservers
        g.a.yimg.com: type NS, class inet, ns n0g.a.yimg.com
```

```
        g.a.yimg.com: type NS, class inet, ns n1g.a.yimg.com
        g.a.yimg.com: type NS, class inet, ns n6g.a.yimg.com
        g.a.yimg.com: type NS, class inet, ns n2g.a.yimg.com
        g.a.yimg.com: type NS, class inet, ns n3g.a.yimg.com
        g.a.yimg.com: type NS, class inet, ns n7g.a.yimg.com
        g.a.yimg.com: type NS, class inet, ns n4g.a.yimg.com
        g.a.yimg.com: type NS, class inet, ns n5g.a.yimg.com
        g.a.yimg.com: type NS, class inet, ns n8g.a.yimg.com
    Additional records
        n0g.a.yimg.com: type A, class inet, addr 24.94.162.85
        n1g.a.yimg.com: type A, class inet, addr 24.94.162.86
        n6g.a.yimg.com: type A, class inet, addr 164.113.247.89
        n2g.a.yimg.com: type A, class inet, addr 24.94.162.85
        n3g.a.yimg.com: type A, class inet, addr 24.94.162.85
        n7g.a.yimg.com: type A, class inet, addr 18.7.20.66
        n4g.a.yimg.com: type A, class inet, addr 24.94.162.85
        n5g.a.yimg.com: type A, class inet, addr 24.94.162.85
        n8g.a.yimg.com: type A, class inet, addr 24.94.162.85
```

Having retrieved an IP address, the browser begins loading the image, while we're still in the process of downloading the HTML from Yahoo:

```
Frame 35 (62 on wire, 62 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: a32.g.a.yimg.com
Transmission Control Protocol, Src Port: 1080 (1080), Dst Port: http (80), Seq: 1
    Source port: 1080 (1080)
    Destination port: http (80)
    Sequence number: 1166630283
    Header length: 28 bytes
    Flags: 0x0002 (SYN)
    Window size: 16384
    Checksum: 0x578f (correct)
    Options: (8 bytes)
        Maximum segment size: 1360 bytes
        NOP
        NOP
        SACK permitted

Frame 36 (62 on wire, 62 captured)
Ethernet II
Internet Protocol, Src Addr: a32.g.a.yimg.com (24.94.162.91), Dst Addr: DBANTTARI
Transmission Control Protocol, Src Port: http (80), Dst Port: 1080 (1080), Seq: 3
    Source port: http (80)
    Destination port: 1080 (1080)
    Sequence number: 3443607970
    Acknowledgement number: 1166630284
    Header length: 28 bytes
    Flags: 0x0012 (SYN, ACK)
    Window size: 32640
    Checksum: 0x011a (correct)
    Options: (8 bytes)
        Maximum segment size: 1360 bytes
        NOP
        NOP
        SACK permitted

Frame 37 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: a32.g.a.yimg.com
Transmission Control Protocol, Src Port: 1080 (1080), Dst Port: http (80), Seq: 1
    Source port: 1080 (1080)
    Destination port: http (80)
    Sequence number: 1166630284
    Acknowledgement number: 3443607971
```

```
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x4472 (incorrect, should be 0x67ea)

Frame 38 (318 on wire, 318 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: a32.g.a.yimg.com
Transmission Control Protocol, Src Port: 1080 (1080), Dst Port: http (80), Seq: 1
    Source port: 1080 (1080)
    Destination port: http (80)
    Sequence number: 1166630284
    Next sequence number: 1166630548
    Acknowledgement number: 3443607971
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 17680
    Checksum: 0x457a (incorrect, should be 0x59d2)
Socks Protocol
```

Socks protocol?? It would seem our packet analyzer is confused by the fact that the source TCP port number for this connection is 1080, which is commonly used by the Socks protocol. So, we won't get to see the HTTP GET request for one of the images used on the page. Note that Ethereal does allow you to override the decode, but I didn't do that here.

Watch the destinations and port numbers carefully; we have two different TCP connections active at the same time. It's easy to get confused, if you're not a computer.

```
Frame 39 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906349343
    Next sequence number: 906350703
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0xce3d (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  6e 62 73 70 3b 26 23 31 38 33 3b 26 6e 62 73 70   nbsp;&#183; 
0010  3b 3c 2f 62 3e 3c 2f 74 64 3e 3c 74 64 20 77 69   ;</b></td><td wi
(blah, blah, blah...)
0530  6f 72 6b 79 20 52 6f 6d 61 6e 6f 3c 2f 61 3e 2c   orky Romano</a>,
0540  20 3c 61 20 68 72 65 66 3d 73 2f 32 31 38 39 3e    <a href=s/2189>

Frame 40 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906350703
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x3eca)

Frame 41 (1414 on wire, 1414 captured)
```

```
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906350703
    Next sequence number: 906352063
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x4b3c (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  54 72 61 69 6e 69 6e 67 20 44 61 79 3c 2f 61 3e   Training Day</a>
0010  3c 2f 73 6d 61 6c 6c 3e 3c 2f 74 64 3e 3c 2f 74   </small></td></t
(blah, blah, blah...)
0530  0a 3c 61 20 68 72 65 66 3d 72 2f 61 74 3e 41 74   .<a href=r/at>At
0540  6c 61 6e 74 61 3c 2f 61 3e 20 2d 0a 3c 61 20 68   lanta</a> -.<a h

Frame 42 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906352063
    Next sequence number: 906353423
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0xa86d (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  72 65 66 3d 72 2f 62 6f 3e 42 6f 73 74 6f 6e 3c   ref=r/bo>Boston<
0010  2f 61 3e 20 2d 0a 3c 61 20 68 72 65 66 3d 72 2f   /a> -.<a href=r/
(blah, blah, blah...)
0530  65 64 73 3c 2f 61 3e 20 2d 0a 3c 61 20 68 72 65   eds</a> -.<a hre
0540  66 3d 72 2f 6c 65 3e 45 76 65 6e 74 73 3c 2f 61   f=r/le>Events</a

Frame 43 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906353423
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x342a)

Frame 44 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906353423
    Next sequence number: 906354783
```

```
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x8df1 (correct)
Hypertext Transfer Protocol
    Data (1360 bytes)

0000  3e 20 2d 0a 3c 61 20 68 72 65 66 3d 72 2f 6c 64   > -.<a href=r/ld
0010  3e 4c 6f 64 67 69 6e 67 3c 2f 61 3e 20 2d 0a 3c   >Lodging</a> -.<
(blah, blah, blah...)
0530  3c 61 20 68 72 65 66 3d 72 2f 63 70 3e 43 6f 6d   <a href=r/cp>Com
0540  70 61 6e 79 20 49 6e 66 6f 3c 2f 61 3e 20 2d 0a   pany Info</a> -.

Frame 45 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906354783
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x16d2 (incorrect, should be 0x2eda)
```

Finally, we get the last packet of the HTML data from Yahoo:

```
Frame 46 (341 on wire, 341 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906354783
    Next sequence number: 906355070
    Acknowledgement number: 1166525632
    Header length: 20 bytes
    Flags: 0x0019 (FIN, PSH, ACK)
    Window size: 17680
    Checksum: 0x05b9 (correct)
Hypertext Transfer Protocol
    Data (287 bytes)

0000  3c 61 20 68 72 65 66 3d 72 2f 63 79 3e 43 6f 70   <a href=r/cy>Cop
0010  79 72 69 67 68 74 20 50 6f 6c 69 63 79 3c 2f 61   yright Policy</a
(blah, blah, blah...)
0100  3c 2f 66 6f 72 6d 3e 3c 2f 63 65 6e 74 65 72 3e   </form></center>
0110  3c 2f 62 6f 64 79 3e 3c 2f 68 74 6d 6c 3e 0a      </body></html>.
```

The PSH and ACK flags are familiar; the FIN flag indicates the destination's intent to close the TCP connection. So, we ACK the last packet from Yahoo; we'll respond to their FIN in a moment.

```
Frame 47 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906355071
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17393
    Checksum: 0x16d2 (incorrect, should be 0x2ed9)
```

Meanwhile, life on our connection to a32.g.a.yimg.com continues...

```
Frame 48 (60 on wire, 60 captured)
Ethernet II
Internet Protocol, Src Addr: a32.g.a.yimg.com (24.94.162.91), Dst Addr: DBANTTARI
Transmission Control Protocol, Src Port: http (80), Dst Port: 1080 (1080), Seq: 3
    Source port: http (80)
    Destination port: 1080 (1080)
    Sequence number: 3443607971
    Acknowledgement number: 1166630548
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 32376
    Checksum: 0x2d7a (correct)

Frame 49 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: a32.g.a.yimg.com (24.94.162.91), Dst Addr: DBANTTARI
Transmission Control Protocol, Src Port: http (80), Dst Port: 1080 (1080), Seq: 3
    Source port: http (80)
    Destination port: 1080 (1080)
    Sequence number: 3443607971
    Next sequence number: 3443609331
    Acknowledgement number: 1166630548
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 32640
    Checksum: 0x701a (correct)
Socks Protocol

Frame 50 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: a32.g.a.yimg.com (24.94.162.91), Dst Addr: DBANTTARI
Transmission Control Protocol, Src Port: http (80), Dst Port: 1080 (1080), Seq: 3
    Source port: http (80)
    Destination port: 1080 (1080)
    Sequence number: 3443609331
    Next sequence number: 3443610691
    Acknowledgement number: 1166630548
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 32640
    Checksum: 0xb0b1 (correct)
Socks Protocol

Frame 51 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: a32.g.a.yimg.com
Transmission Control Protocol, Src Port: 1080 (1080), Dst Port: http (80), Seq: 1
    Source port: 1080 (1080)
    Destination port: http (80)
    Sequence number: 1166630548
    Acknowledgement number: 3443610691
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x4472 (incorrect, should be 0x5c42)

Frame 52 (1414 on wire, 1414 captured)
Ethernet II
Internet Protocol, Src Addr: a32.g.a.yimg.com (24.94.162.91), Dst Addr: DBANTTARI
Transmission Control Protocol, Src Port: http (80), Dst Port: 1080 (1080), Seq: 3
    Source port: http (80)
    Destination port: 1080 (1080)
    Sequence number: 3443610691
```

```
    Next sequence number: 3443612051
    Acknowledgement number: 1166630548
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 32640
    Checksum: 0x7430 (correct)
Socks Protocol
```

Looks like our image is done, we got a packet with a PSH flag.

In other news, we now respond to the server we got the HTML from, with a FIN/ACK flag pair, indicating that we agree to close the connection:

```
Frame 53 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: www.yahoo.akadns
Transmission Control Protocol, Src Port: 1078 (1078), Dst Port: http (80), Seq: 1
    Source port: 1078 (1078)
    Destination port: http (80)
    Sequence number: 1166525632
    Acknowledgement number: 906355071
    Header length: 20 bytes
    Flags: 0x0011 (FIN, ACK)
    Window size: 17393
    Checksum: 0x16d2 (incorrect, should be 0x2ed8)

Frame 54 (683 on wire, 683 captured)
Ethernet II
Internet Protocol, Src Addr: a32.g.a.yimg.com (24.94.162.91), Dst Addr: DBANTTARI
Transmission Control Protocol, Src Port: http (80), Dst Port: 1080 (1080), Seq: 3
    Source port: http (80)
    Destination port: 1080 (1080)
    Sequence number: 3443612051
    Next sequence number: 3443612680
    Acknowledgement number: 1166630548
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
    Window size: 32640
    Checksum: 0xd01e (correct)
Socks Protocol

Frame 55 (54 on wire, 54 captured)
Ethernet II
Internet Protocol, Src Addr: DBANTTARI (65.28.72.130), Dst Addr: a32.g.a.yimg.com
Transmission Control Protocol, Src Port: 1080 (1080), Dst Port: http (80), Seq: 1
    Source port: 1080 (1080)
    Destination port: http (80)
    Sequence number: 1166630548
    Acknowledgement number: 3443612680
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
    Window size: 17680
    Checksum: 0x4472 (incorrect, should be 0x547d)
```

And finally, we get the last ACK from Yahoo, officially closing the connection.

```
Frame 56 (60 on wire, 60 captured)
Ethernet II
Internet Protocol, Src Addr: www.yahoo.akadns.net (64.58.76.223), Dst Addr: DBANT
Transmission Control Protocol, Src Port: http (80), Dst Port: 1078 (1078), Seq: 9
    Source port: http (80)
    Destination port: 1078 (1078)
    Sequence number: 906355071
    Acknowledgement number: 1166525633
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
```

```
    Window size: 17680
    Checksum: 0x2db9 (correct)
```

In this trace, we don't see the image connections to the image server being closed, but they're probably reused anyway (HTTP 1.1 allows this.)

Packet analysis is often the best way to troubleshoot errors; you can see, byte by byte, exactly what is being communicated from client to server and back. In some cases, packet analysis is the *only* way of diagnosing problems.

More information on Ethereal and packet sniffing, see the Ethereal Links page, at http://www.ethereal.com/links.html.

---

16. WAN Connectivity

Wide area networking is actually fairly simple conceptually, but can also be one of the most difficult aspects of networking. On the Near Side of the 'Net, however, things rarely get exceedingly difficult, and with a proper understanding, can be quite easy and simple.

It's quite likely you're reading this over a Wide Area Network (WAN) connection-- your dial-up connection to the Internet! What you've done is run a serial cable all the way across town, over streets, under bridges, to your Internet Service Provider (ISP), right?

No?

Oh, I see. You have a serial cable that connects to a modem, that connects (through the phone system) to your ISP's modem, that connects (serially) to a device that allows PPP Internet connections.

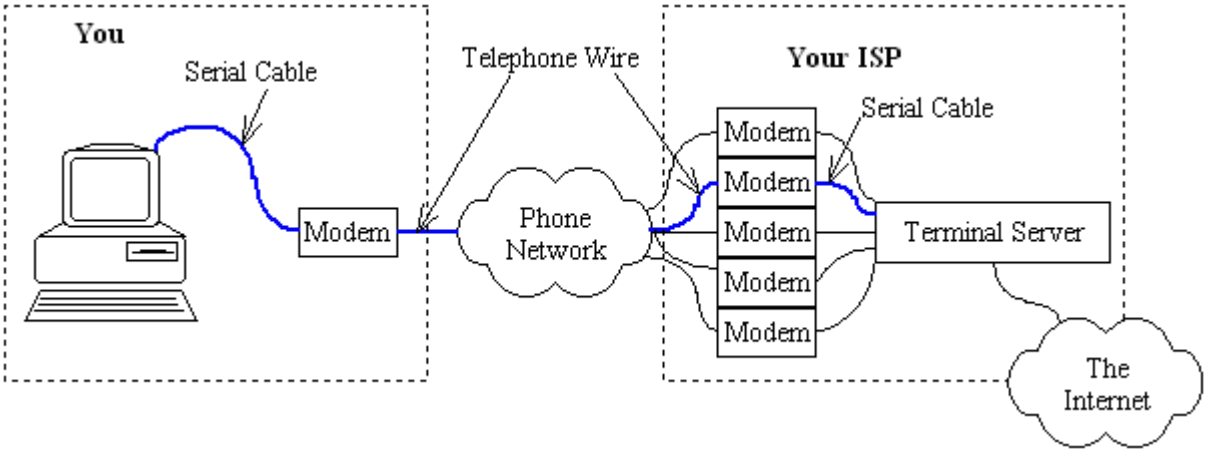


Image created using SmartDraw. Click Here for a free trial copy.

Follow the transition: Serial Cable, [Modem], Phone System, [Modem], Serial Cable. **From this perspective, a modem is nothing more than a serial cable extender, that allows you to run a serial cable through the phone network.** And that's all it is-- a serial cable extender. From a layer 1/layer 2 perspective, the sole function of a modem is to allow you to extend your serial connection through a phone system. **Most WAN links are simply some method of serially connecting two routers through the public telephone network.** The only real differences are in speed and flexibility.

Ok, let's cover some terminology:

Point-To-Multipoint Networking

Networking where one device may be physically connected to multiple devices, such as when using Ethernet or Token-Ring. A layer two address (typically, a MAC address) is required to indicate to the network which device you're talking to. Typically used for LAN connectivity.

Point-To-Point Networking
: Networking where one device is physically connected to one device, such as when using a serial cable (or extended serial cable) and the PPP protocol. There is no concept of MAC address in this case (which can present some difficulties when routing IPX over WAN links, but that's outside the scope of this document.) Typically used for WAN connections.

PPP
: The Point to Point Protocol. Provides a standard way of running multiple protocols simultaneously over a WAN link.

SLIP
: The Serial Line Internet Protocol. Provides a way of running IP over a dial-up WAN link. As far as I know, no one still uses it; SLIP has been replaced by the more flexible PPP.

Modem
: Provides a means of extending a [digital] serial link over an [analog] voice network.

ISDN
: Integrated Services Digital Network. Originally designed to replace the Plain Old Telephone System (POTS), high price and restricted availability have restricted it's adoption primarily to medium-speed WAN connections. More on ISDN in a bit.

Frame Relay
: A point-to-point, point-to-multipoint hybrid that allows multiple "virtual" connections, or circuits, to exist on a single physical connection. A frame-relay "cloud" in the center, managed by the intermediary telco(s), manages the frame-relay network so you don't have to. Or, that's the way it's supposed to work, anyway. :-)

Frame Relay PVC
: A **P**ermanent **V**irtual point-to-point **C**ircuit through the frame-relay cloud.

POP
: Point of Presence. Typically used to describe the a location from which a service is provided. For example, a ISP modem bank can be referred to as a modem POP, or a frame-relay switch can be referred to as a frame-relay POP.

56k
: A digital WAN circuit leased from the phone company. Allows communication at 56kbps bidirectionally. Can be connected directly to another office location, or to the nearest frame-relay POP. Requires a 56k CSU/DSU to be useful as a digital WAN link.

T1
: A digital WAN circuit leased from the phone company. Originally designed to reduce the need for copper under streets (they were running out of room,) a T1 is configured into 24 digital channels, each of which can carry one digitally encoded voice conversation. For use as a serial cable extender (WAN link), a T1 CSU/DSU is required.

CSU/DSU
: Converts the digital signaling of a serial cable into the digital signaling of the telco network; functionally, the same role as a modem. See "T1 Connections", later in this section, for more information.

ISDN

ISDN, or Integrated Services Digital Network, was the digital technology that was supposed to replace analog telephones. However, lackluster (and 'lackluster' is being generous) support from US phone companies have hobbled ISDN's chances of ever replacing the current analog networks. Phone conversations are typcally analog between you and the local phone switch; digital from switch to switch; then analog from the destination switch to the other person you're talking to. This analog-

digital-analog conversion makes the engineers of modem manufacturers lose sleep. Since ISDN is end-to-end digital, it is well suited to carrying data as well as voice. The basic consumer ISDN connection is a Basic Rate Interface, or BRI, circuit. A BRI is physically installed as a single pair of copper wire, but has three logical "channels" (think TV channels.) These channels are referred to as "B" or "D" channels. BRI "D" channels are 16kbps and are used by IDSN equipment for talking to the telco switch ("You have and incoming call" or "I want to call this number"). ISDN "B" channels are 64kbps and a BRI circuit contains two of them. For this reason, people often refer to BRI as "2B+D". Each "B" channel is considered to be a seperate phone line by the phone company, which becomes important if you want to use both of them simultaneously for dial-up connectivity, or when the per-minute bill arrives from the phone company.

That connection from a single pair of copper is known in ISDN circles as a "U" interface, and the phone company expects you to attach an "NT1" to it. An NT1 then provides two-pair "ST" interfaces to the various ISDN devices around your house. In practical use, most people don't use ISDN for voice. Hardware manufactures have picked up on that fact and will usually build the NT1 right into the device-- the device, then, is said to have a "built-in NT1" or have a "U Interface". Devices that expect an external NT1 are usually described as having an "ST" interface and are less expensive than their NT1 Interface counterparts. In most cases, when using ISDN for networking purposes, you will want to purchase a device with a built-in NT1.

Submitted by Tony F.:
In Europe connection to the ISDN network is via the 'S' interface. The difference being (in no technical terms) is that the conversion from the signals on the coppper...to digital format is done by the service provider. In the US the ISDN device that you buy does this bit as well. ie the customer pays [for and owns] the conversion [hardware].

Multilink PPP and BACP

Although ISDN is split into two channels, dialing two seperate [regular] PPP connections to an ISP is not desirable; you would have two different IP addresses, and the best throughput possible in either direction is 64kbps (sending data on one channel while receiving data on the other.) Since most "near side of the 'net" connections are primarily receiving data, having the ability to mostly receive data is important. Enter Multilink PPP (MLPPP). Simply put, Multilink PPP allows a single logical PPP connection to span multiple physical connections. A newer protocol, the Bandwidth Allocation Control Protocol (BACP), allows channels to be added and dropped dynamically, typically in response to higher utilization. Typically, MLPPP asks for two phone numbers to dial, but the two phone numbers are usually identical. BACP will usually ask for the minimum and maximum number of channels to connect. The minimum is generally 0 for outbound-only Internet connections, and 1 for "listening" connections such as mail or Web *servers*; the maximum number of channels is almost always 2.
See Dan Kegel's ISDN Page for much more ISDN information.
http://www.alumni.caltech.edu/~dank/isdn/

56k Connections, Analog

Ahh, the wonderful, ubiquitous 56k dialup connection. It's often all that's required for a small LAN to send and receive e-mail and do some light Web browsing. Many ISPs only offer one POP e-mail account for a dialup connection, but there are other services you can use to add POP accounts for free: Hotmail and Yahoo come to mind. Because you generally get one (varying) IP address, some means of "hiding", or proxying, several "fake" IP addresses behind your single "real" IP address. See the section on NAT for more information. Personally, I use WinRoute for this purpose. One of these days, when I have the spare time (yeah-- right), I intend to get my Linux machine doing this via IP Masquerading. However, my current configuration is working quite well, and fixing what's not broken tends to rank fairly low on the priority list.

56k Connections, Digital

A 56k digital connection is specially ordered from the phone company. Unlike regular phone lines, digital lines understand the digital communication traversing them; the signals are repeated, rather than amplified (which will amplify noise along with the signal.) So, the noise is removed, and the signal is regenerated. Remember, analog signals are *amplified,* increasing noise along with signal, and digital signals are *repeated,* removing the noise and pushing clean, new data down the wire. So, true digital lines have almost no difficulty with signal degradation or data corruption, compared to analog communication. In Europe, the equivalent to a 56k line is a 64k digital line.

T1 Connections

T1 lines were developed because the under-street tunnels in cities were getting clogged with copper wires carrying single conversations. T1 lines digitallencoded 24 voice channels were digitally encoded using time-division multiplexing, and sent over two pair of copper, thereby increasing the ability of copper to carry voice communications by a factor of 12. When two voice telephone switches are connected by a T1 line, the voice communications encoded must be alternately encoded into channels, then decoded from T1 channels back into discrete voice connections. The device that does this is called a CSU, or Channel Service Unit. Data communications specialists quickly found T1 lines to be perfect carriers of data, as well. Since all data must be converted to digital "channels" before being placed on the T1, a device called a Data Service Unit was designed to take a serial cable, and split its data into channels usable by a CSU. So, a router is typically connected to a CSU/DSU, which encodes the data and transmits it over the T1 line. More recently, the CSU/DSU is actually built in to the router. Still, a T1 line, from our standpoint, is just a serial cable extender.

Frame-Relay

In a nutshell, a way of creating virtual data links through a shared-bandwidth "cloud" of routers. Think of hard-wired VPN connections through a private data network, without the encryption.

Routing Over WAN Links

Most of this document has covered networking methods that connect one computer directly to many computers, such as Ethernet or Token Ring. WAN links are, however, point-to-point, so they are treated a bit differently. Remember, each hop between communicating devices has the sole responsiblity of getting each packet to the next hop that is closer to the packet's destination. Routing tables on Ethernet-connected machines typically include the IP address of the next router, since simply specifying "the Ethernet interface" is not nearly specific enough. However, when routing over point-to-point links, next-hop specifications often are, simply, "send packets for these destinations over the Serial0 interface." We don't *need* to specify IP addresses for point-to-point connections (though many admins do), since the routing function can be completed simply by placing the packet on the wire that will bring it to the next hop toward its destination. If this seems confusing, remember: neither the source nor the destination IP address will change on the packet at any point between sending and receiving, though the layer 2 information will be repeatedly stripped off and replaced, as needed, by the intermediate routers. (Network Address Translation is, of course, an exception to this rule, since NAT alters both the IP and the (TCP or UDP) headers of all packets traversing a NAT router.)

---

17. Questions and Answers

The following are questions submitted to me via e-mail. The answers may not always be complete, and quite often there are unmentioned exceptions (that, of course, prove the rule :-)

As usual, use any information here at your own risk; I am not responsible if any errors or omissions that adversely affect you.

If you submit a question to me, please include whatever details you can to help me answer. I don't guarantee a response; if I do respond, I may post the response here, without your full name, edited for brevity, and after altering any IP addresses to preserve your anonymity.

Question added 6/8/2001, submitted by Manoj

Q: Can you please tell the differences between logical port and physical port?

A: Think of a logical port as a "software" port and a physical port as a "hardware" port. For example, if you have a Frame Relay T1 line installed so that you can commumicate with 5 branch offices, each of the 5 virtual circuits are terminated at a logical port, yet there is only one physical T1 port on the router.

Every physical port has at least one logical port associated with it (unless it's disabled.)

Question added 12/1/2000, submitted by Suraj

Q: Can you explain to me the concept of "SUPERNETTING" with a good example?

A: "Supernetting" is really a term that predates RFC 1812, and refers to the practice of combining multiple contiguous "classfull" networks into a larger subnet than RFC 950 would allow.

For example, you could combine the networks 192.168.2.0/24 and 192.168.3.0/24 into one network 192.168.2.0/23 (note the 23), but RFC 950 does not allow for "Class C" addresses to have a subnet mask of less than 24 bits. RFC 1812, however, removes the class restrictions from networks, and therefore effectively obsoleted the term "supernetting."

Technically, RFC 1812 also obsoletes the term "subnet mask" and replaces it with the term "network prefix", but it's a distinction that is rarely made.

Question added 10/15/2000, submitted by Shaowu

Q: I was wondering if you would like to give me some ideas about the following question?

Consider that a host, say A, of IP address in class B and the subnet mask "255.255.0.0", and a host, say B, of IP address in class C and the subnet mask "255.255.255.0". Assuming that the host A and host B can "ping" each other. After changing the subnet mask of the node B to "255.255.0.0" (just done on the host B), could host A and host B "ping" each other? If not, how to let them "ping" each other with the codition that they keep their original IP address and use the same subnet mask "255.255.0.0"?

A: As long as the subnets don't overlap, and the routing is properly configured (which it is if you can ping each other), then you're fine. Remember, you only know (or care) what the destination subnet mask is if the destination subnet also happens to be the source subnet.

The first routing decision is made by the sender, and that decision is whether to send the packet to the router, or to send it directly to the destination host (if the sender determines that both machines are on the same subnet.) As long as the routing is properly configured on both sides, you'll be fine. In fact, many misconfigurations will work fine, too, if a touch inefficiently. If you use a subnet mask that is too specific, you may wind up sending many packets destined for local hosts to the router, which then will put the packets right back on the same wire back to the local destination. If your subnet mask is too broad, then you won't be able to reach any hosts on non-local subnets included in the overly broad mask (since the packets won't be sent to the router for delivery.)

The goal of routing is to eventually get the packet to a router that is on the same subnet as the destination.

Play with the subnet calculator a bit to see that first routing decision in action.

Question added 6/13/2000, submitted by Victor

Q: What does a company like yahoo do when they receive too many hits on their single IP address. Do they just get a bigger router? i.e. Two routers on the net cannot have the same IP address which implies they just have to upgrade to a bigger router, they can't just add another. Also, what if the servers behind the router use up all the port numbers? Does this mean that no more servers can hide behind the single IP address? I've just been really curious how the major internet companies scale their networks to handle thousands of transactions and users.

A: First, remember that there can be a one-to-many relationship between names and IP addresses, and with hardware assistance, between IP addresses and servers. For example, www.microsoft.com maps to five different IP addresses:

C:\>nslookup www.microsoft.com
Server: ns.cicom.net
Address: 208.142.64.3

Non-authoritative answer:
Name: microsoft.com
Addresses: **207.46.131.30, 207.46.130.14, 207.46.130.149, 207.46.130.45, 207.46.131.137**
Aliases: www.microsoft.com

Additionally, each of those IP addresses probably represents whole cluster of Web servers hidden behind a hardware device such as Cisco's Local Director. Remember, it's the Web servers doing the bulk of the work; routing is relatively simple by comparison. Building a database cluster to handle that page request load is another feat.

The two flaws in the question: first, a DNS name can (and often does) point to multiple IP addresses, which facilitates running many servers in parallel to handle very large loads; second, it's not the routers that are problematic in handling significant loads-- generally speaking, Web site capacity planning focuses on the database and Web server(s), not the router(s).

Question added 3/21/2000, submitted by Kathy

Q: I have been trying for several months to find information on tracking the origin of an e-mail thru the message ID, I have had no success. Do you know of a way to track if so can you share that information or is this considered a trade secret???????

A: No secrets here :-)

You can find out much about the origin of a message by the headers. The following are the headers from a SPAM letter I got the other day (you may need to find a "view headers" option in your e-mail client to view these on your own messages):

x-recipient: <daryl@windsorcs.com>
Received: from rech1.werbebuero1.de [195.90.0.60] by windsorcs.com (FTGate 2, 2, 2, 1);
Sun, 19 Mar 2000 23:46:50 -0600
Received: from 198.144.76.204 (pt-006-00189.greenapple.com [198.144.76.204])
by rech1.werbebuero1.de (8.9.3/8.9.3) with SMTP id HAA29140;
Mon, 20 Mar 2000 07:26:06 GMT
From: ftr5@aol.com

Message-Id: <200003200726.HAA29140@rech1.werbebuero1.de>
To: grw3@aol.com
Subject: 20% off top quality health products
Date: Mon, 20 Mar 00 00:46:13 Eastern Standard Time
Reply-To: ftr5@aol.com
X-Priority: 3
X-MSMailPriority: Normal
Importance: Normal
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="----=_NextPart_000_018C_01BD9940.715D52A0"
Content-Type: text/html;

What we're seeing here (read the Received: lines from bottom to top) is that someone at 198.144.76.204 (which has a DNS name of pt-006-00189.greenapple.com) used the mail server rech1.werbebuero1.de (195.90.0.60) to relay SPAM to me. I would notify abuse@greenapple.com and postmaster@greenapple.com that a customer was sending SPAM from their network, and I would notify abuse@werbebuero1.de and postmaster@werbebuero1.de that their mail server was misconfigured and being used to relay SPAM, thereby filling up their mail queue and causing mail delays for their important mail, using all their expensive bandwidth, etc...

Note that the From: address is probably forged/fraudulent, and the To: address has nothing to do with who the mail was sent to.

The from, to, and reply-to headers are created by the sender; the commands that indicate who the /real/ recipient is don't get included in the actual message.

Note that it's a trivial matter to forge un-digitally-signed e-mail from and to someone. Fortunately, the intermediate mail servers generally always include the IP address of the source of the e-mail. If that can be tracked (with help from the date/time stamp) to a sender, you can find the exact individual who sent the e-mail.

If you require secure communications with someone, look into PGP at http://web.mit.edu/network/pgp.html. It is a system that can prove a message is from a certain sender and has not been tampered with (digital signature) and, optionally, it can also encrypt the message so that only the sender's designated recipients can read it (though they also need PGP to either verify the signature or decrypt the message.) It's free for personal use, and businesses need to pay a reasonable fee. For an excellent document on how PGP works, see http://www.pgpi.org/doc/pgpintro/.

Question added 3/2/2000, submitted by Luis

Q: If host A wants to PING host B which is 3 routers away, host A must ARP for his gateway (router). Once he gets the MAC for the gateway, host A sends the packet to the gateway (we know it did a Boolean "AND" and determined that the "pinged" IP address was not local. My question is this:

When the router closest to host A (his gateway) receives the packet and realizes that he does have a route to host B, does the packet leaving router have as a source IP that of the router itself or the IP of the host A????

A: The layer 3 source and destination IP address do not change between the source and destination, unless you have a firewall in between performing Network Address Translation. **The layer 2 information, however, is stripped off and replaced at every hop.** If a packet traverses an

Ethernet network to the first router, a PPP link to another router, and reaches a destination on a Token-Ring network, then the packet will have a destination MAC address of the first router for the first hop, no destination MAC address for the second hop (since PPP is point-to-point, it doesn't need to specify which machine is supposed to get the packet), and the third hop will have a Token-Ring header with a source MAC address of the *router*, and destination MAC address of the target machine. At no point do the source and destination machines know anything about the layer 2 configuration of each other (unless they're on the same subnet.)

Question added 3/1/2000, submitted by Sid

Q: This is a great and very useful source of information. *[Thanks!]* I still don't completely understand TCP/IP subnets and submasking, but this really helps. Hopefully, after I've read it a few more times, I'll find the answer to my question: do all hosts on a LAN need to have the same submask setting?

A: All hosts on a given LAN subnet **must** share the same mask. The subnet mask, when applied to the host's IP address, indicates which other IP addresses should be on the same network. Packets sent to destinations not believed to be on the same network are sent via the best available route--usually to the default router.

An easy way to theoretically disconnect yourself from the world is to set a subnet mask for your workstation that suggests that your default router is on a different network. If you can't send packets to your default router, then you can't talk to anyone not on your subnet. (I've noticed, however, that Windows machines will attempt to ARP the Ethernet address for any IP address given to Windows as the default router's IP address. So, in this case, if the default router is physically on the same network, even if it's logically seperate, it will work.)

To see the subnet mask in action, refer to Daryl's Subnet Calculator (heh.. someday I'll come up with a more creative name), at
http://ipprimer.windsorcs.com/subnet.html

Question added 6/5/1999, submitted by Kent

Q: This is a great site. Thank You. [*You're welcome.*] I do have one question concerning subnetting and when to do so. How many nodes can you put on one TCP/IP subnet before it requires segmenting your network? I am referring to a Lan with approx. 300 users. Is there a reason why I can't use a standard 255.255.000.000 subnet. I will only be assigning addresses in my DHCP scope as the network requires them.

A: This is a good question, and really is more of a layer two question than a TCP/IP question. I would not run a 300 user lan on a single 10Mbps Ethernet segment; however, I wouldn't balk at a 300 user network segmented into 12 or 24 switched partitions using a centralized Ethernet switch. So the real question here is, "will my current layer two network topology support 300 users on one segment?" You can put as many nodes as you want on one TCP/IP segment; however, that lack of limitation does not apply to Ethernet. (I would ensure no Windows boxes are running NetBEUI, though.)

Remember, a switch "segments" networks on layer two, and a router "segments" on layer three. The main difference, from a topology planning standpoint, is that switches forward broadcast packets and routers don't. Thus, switching becomes a problem quickly with "loud" protocols like NetBEUI, since switching doesn't reduce or segment broadcast traffic.

You can use a subnet mask of 255.255.0.0 to put up to 65,534 hosts on a single routed network segment; or you can use a subnet mask of 255.255.254.0 to put up to 512 hosts on a network segment. I'm assuming you're using "reserved" addresses (such as 10.1.x.x) behind a NAT firewall or proxy, so the choice of subnet mask is yours. The choice of whether or not to segment by switching or routing is also yours; I tend to prefer switching, since it tends to keep things simpler.

Question added 1/23/1999, submitted by NBK

Q: How vulnerable is Linux against Net attacks compared to NT??? Damn NT has to many holes....

A: In both cases, it depends on the administrator :-)

a good packet filter or (better yet) firewall, good knowledge of the security issues of the services the box is providing, and keeping current on the security updates/mailing lists for the OS'es and running services makes for a pretty strong box. Any badly installed service can present the opportunity for a full breach; be sure to read the security FAQ's (and I'll often scan cracker websites) for the OS and the services you're making available to the public.

Question added 12/3/1998, submitted by David

Q: This is to request from you a tutorial on TCP/IP.
Thank you very much.

[Answer: can you be more specific? Platform, etc?]

Actually I'm looking for an overview on the internet network. How the providers build their network...
How do they get inteconnections...
What are the critical economical issues for internet on the next years...etc

A: Hm... That's intentionally outside the scope of the Primer (hence the subtitle, "...the near side of the 'net.") For the information you're looking for, search for "BGP4" re: interconnections, and regarding economic issues (etc) try any of the Internet trade rags for the professional pundits :-)

Doing generic dialup and hosting does not (IMHO) have an entry level any more; the services are very commoditized and the economies of scale involved will squeeze out the smaller non-value-added providers. But (apologies to Dennis Miller) that's just my opinion, I could be wrong.

Question added 10/12/1998, submitted by Joanne

Q: The part I don't understand is: what is the reason to subnet? You can't possibly get more destinations that way, I mean, 32 bits are 32 bits. There's only 4 billion possible internet destinations, no matter how you split it up. So what does subnetting do for ya?

A: Subnetting does two things, depending on what context you're in:

If you're a workstation (or server), the subnet mask is used to determine whether the destination IP address is on your same subnet; if so, the workstation will attempt to ARP the destination's Ethernet card address and deliver the data directly; remember, **the first routing decision is made by the workstation,** and the decision is: whether or not to send the packet to a router.

Routers keep their routing tables managable by clumping large blocks of addresses together using broad subnet masks ("Network Prefixes"). In the old days of classful routing, routers would have to keep track of each "Class C" address individually, which was causing extreme growth of routing tables; CIDR routing allows you to clump as many "Class C" networks together as you want (in powers of 2.)

So, you may ask, what about servers that also act as routers? In which category to they fall? Well, I lied when I said that subnetting does different things depending on context; it's just that most IP end stations (workstations) don't bother trying to keep track of the whole network; they just know that "these addresses are local, and I'll send anything else to my default gateway/router."

Question added 10/6/1998, submitted by Bob

Q: Is it possible with IE or netscape to address a web server by its MAC address?

A: It sounds like you're asking if you could run HTTP over DLC; the short answer is "no."

The long answer: the HTTP protocol is based on the TCP protocol, which is based on IP; therefore, both the client and server must already be running IP for HTTP to work. You could force client and server IP address into their local ARP caches if they are on the same subnetwork (bounded by routers), but I dont know how well that would work (I doubt the IP stack checks its arp cache before it determines whether or not a given IP is on a locally attached subnet.) If it did work, you could then type the (fake?) IP address of the server into your browser's location line to pull pages. The server would then reply to your (fake?) IP address.

Alternatively, if there is an IP router involved, you could play with its ARP cache; routers are more likely to be forgiving about having multiple IP subnets (or, network prefixes, in RFC 1812 parlance) on the same subnetwork than, say, Win95 workstations.

Note that on any point-to-multipoint network (like Ethernet or Token Ring, but not including serial PPP or HDLC connections), the most basic address (in the layer 2 MAC header) is the MAC address. But you cannot type a MAC address into 'IE or netscape' and connect to a web server; even if you could, the web server would not know what IP address nor TCP socket number to reply to.

Question added 9/30/1998, submitted by Jim

Q: I just have a quick question, its regarding Windows 95 (Yeah, I hear you screaming), when you set the computer to 'disable DNS', and don't set a gateway address (all via control panel) and disable WINS--how is anything assigned to the computer? Is it fair to assume its BOOTP, or something else?

A: Probably DHCP;

BOOTP assigns the IP address, subnet mask, default gateway (route), and (if memory serves) the DNS information. DHCP allows for a bunch of other information to be sent to the workstation, including WINS server addresses. DHCP also has a facility for "lease expiration", where addresses that are not renewed are returned to the pool of available addresses; under BOOTP, IP addresses are permanently associated with the NIC's MAC address, so if you throw out the NIC, the IP address is "lost." Win95 does not support BOOTP.

DHCP and WINS are two very different things, they just seemed to "appear" at the same time (with the introduction and subsequent popularity of Windows 95 and NT Server 3.5x). DHCP is used for automatically configuring workstations with all the information they need to access the TCP/IP resources available to them, including IP address, subnet mask, default gateway, and on Windows NT networks, WINS server addresses. WINS is like DNS for NT networks; WINS is used to "advertise" and locate NT server and (win95|nt) workstation resources on the NT network, such as shared drives and printers. DHCP is a non-Microsoft-specific "upgrade" to BOOTP, WINS can be described as a Microsoft Networking version of DNS. (Novell's version of WINS for distributing SAP information is called DSS, or Domain SAP Server.)

BTW-- Win95 doesn't make me scream, but don't bring any Win3.X machines by unless you're equipped with earplugs :-)

18. Update Notifications/Comment Form

I do update Daryl's TCP/IP Primer on an irregular basis; if you'd like to be notified of these updates, or if you want to send me a comment or suggestion, use the appropriate boxes below. I will not use your name nor email address for any other purpose than to alert you of updates to Daryl's TCP/IP Primer, and those notifications (even for minor updates) don't happen very often; expect an email every 2-6 weeks or so (irregularly) for minor updates; major updates anywhere from quarterly to annually. I add Q&A's as time allots; possibly two in one day and none for the month following. You can always change your notification options later by resubmitting the form (instructions will be

provided in every email.)

If you submit the comment form more than once, you will not receive duplicate notifications; you can, in fact, change your notification option this way.

Of course, if you want to make a donation, that's cool, too. And it makes a very nice incentive toward putting the time in to maintain and expand the site.

Notification and Comment Form	
*Your Name:	
*Your E-Mail Address:	
Notify Options:	☑ Major Updates (e.g., new chapter; rare) ☑ Minor Updates (content added; less rare) ☑ New Q&A Added (every week or so, so far)
Comments:	
Submit Query	

---

19. Other Sources

Other publications by Daryl Banttari

(Including Daryl's ColdFusion Primer) are listed at http://www.windsorcs.com/

On the 'net: (in no particular order)

The Internet.com Webopedia: The top non-search engine referrer of people to Daryl's TCP/IP Primer. It must be a good resource. ;-)
The Information Technology Professional's Resource Center: Well-organized set of links to other TCP/IP resources. Not for the weak of vision-- everything is <font size=1>
The IT Web Links Pages - Protocols: Links to other information on TCP/IP
Uri's TCP/IP Resources List: 'This posting contains a list of various resources (books, web sites, FAQs, newsgroups, and useful net techniques) intended to help a newbie to learn about the TCP/IP suite of protocols.'
Patrick's MCSE Place: Lots of MCSE stuff, links.
Google's Directory Service: the "best" links to networking resources. Too bad they don't acknowledge my requests to fix their link to me. >:-(

Your link here.

In Print: (with convenient links to purchase the books from Fatbrain.com)

This is a look at my bookshelf- I have included all of the books with more than one crease in the binding.

Topic	Recommendation
Setting up UNIX services for the Internet, including details on TCP/IP and TCP/IP services like DNS and SENDMAIL. Helpful, even if you never touch UNIX.	TCP/IP Network Administration by Craig Hunt Published by O'Reilly and Associates, Inc.
Everything you ever needed to know about DNS. Referred to as the "DNS Bible" or simply "The DNS Book" on DNS mailing lists.	DNS and BIND by Paul Albitz, Cricket Liu, Mike Loukides Published by O'Reilly and Associates, Inc.
Packet filtering and general Internet security	Building Internet Firewalls by D. Brent Chapman, Elizabeth D. Zwicky, Deborah Russell Published by O'Reilly and Associates, Inc.
UNIX System Administration. Covers issues with several *nix flavors.	Essential System Administration : Help for Unix System Administrators by AEleen Frisch Published by O'Reilly and Associates, Inc.
My Linux command reference. Always at my elbow when I'm doing anything interesting on my Linux box.	Linux in a Nutshell by Jessica Perry Hekman, Andy Oram (Ed) Published by O'Reilly and Associates, Inc.
High speed Internet connectivity	Getting Connected : The Internet at 56K and Up by Kevin Dowd, Mike Loukides Published by O'Reilly and Associates, Inc.
JavaScript: This book had everything I needed to make the Subnet Calculator work.	Javascript : The Definitive Guide by David Flanagan Published by O'Reilly and Associates, Inc.
Interdomain (Backbone) routing-- how the big boys and girls do it.	Internet Routing Architectures by Bassam Halabi Published by Cisco Press
Or, check out FatBrain's category for Computing & Internet Subjects : Networking & Telecommunications : Protocols & Standards : TCP/IP	

Other Favorites	
The first book I reach for when I have a question about how my favorite DBMS, Microsoft SQL Server, works. I was ready to leave NT behind and switch to ColdFusion/Linux (from ColdFusion/NT) when SQL Server 7.0 came out.	Inside Microsoft SQL Server 2000 By Kalen Delaney,Ron Soukup Published by Microsoft Press
A must-read for any person who manages software development. The source of the oft-quoted "Brooks' Law": '*Adding manpower to a late software project makes it later.*'	The Mythical Man-Month By Frederick P. Brooks Jr. Published by Addison Wesley Longman
Although I prefer ColdFusion for Database-To-Web projects, Java occaisonally fits the bill when heavy lifting is required. At the time of this writing, there are nineteen books on ColdFusion listed at Fatbrain, but I can't personally recommend any of them-- probably because I've been using CF since version 2.0 and simply don't need to be told how CF works.	Java Servlets By Karl Moss Published by Osborne/McGraw-Hill
JDBC and Java: Complete. Consise. What would we do without O'Reilly?	Database Programming with JDBC and Java By George Reese,Andy Oram (Editor) Published by O'Reilly and Associates, Inc.

In fact, every book I've read from O'Reilly and Associates has been very good. If you see one of their books relating to a subject you're interested in, my advice is to buy it.

---

20. Glossary

"A" Record

1. Used by Domain Name Service (DNS) administrators to assign an IP address to a host name.
2. A DNS host record, used to name-resolve all non-email addresses.

Address Lookup

see Domain Name Service.

Address Resolution Protocol (ARP)

1. A protocol used to find the network interface that "owns" a particular IP address.
2. On LANs, this is used to get the Layer 2 address of a host, so that IP transmission can take place over Layer 2 protocols like Ethernet or Token-Ring.

ARP

see Address Resolution Protocol

Banana

1. ~~A tropical fruit.~~
2. *Contributed by Brent G. Gratias:*
   banana (be-nàn¹e), name for a family of tropical herbs (the Musacae), for a genus (Musa) of herbaceous plants, and for the fruits they produce. Bananas are probably native to tropical Asia, but are widely cultivated. They are related to the economically valuable MANILA HEMP and to the BIRD-OF-PARADISE FLOWER. Banana plants have a palmlike aspect and large leaves, the overlapping bases of which form the so-called false trunk. Only female flowers develop into the

banana fruit (botanically, a berry), each plant bearing fruit only once. The seeds are sterile; propagation is through shoots from the rhizomes. Bananas are an important food staple in the tropics. [some people take humor so *seriously!*]

Class

1. [Common usage] "Class C Subnet" is a convenient shorthand for a network prefix of 255.255.255.0
2. The "old" way of allocating sections of the TCP/IP address space

see "Class A", etc. in "IP Addresses, Subnet Masks, and Subnetting, Part A," above.

DNS

see Domain Name Service.

Domain Name Service (DNS)

1. The means by which computer names (e.g. "ipprimer.windsorcs.com") are converted into IP addresses.
2. All communication on the Internet is done based on IP addresses. The Domain Name Service allows "us humans" to use names for computers, which, for us, are much easier to understand and remember. The "address lookup" process converts address names like "www.microsoft.com" to IP addresses like "207.68.137.9".

Host

1. For purposes of this discussion, a Host is an IP-capable machine connected to an IP network.
2. A computing device attached to an IP network by only one interface. Although a host can have more than one interface, those hosts usually perform routing functions, and are therefore called routers.

Interface

1. A physical connection to a network.
2. An Ethernet card is an example of a Network **Interface** Card (NIC).
3. Any connection endpoint capable of sending and receiving packets over some medium.

Internet Protocol (the IP in TCP/IP)

1. The address system of the Internet.
2. The Internet Protocol is a four-byte addressing system, where each byte is expressed in decimal numbers and separated by a period, like "168.192.1.1".

Internet Service Provider

1. The people you pay to get Internet access. (Besides the phone company.)
2. Any person or company that provides services related to the Internet, esp. those that provide connections for home and office users.

Internetwork Packet eXchange (IPX)

1. The address system of Novell networks.
2. A protocol stack typically used on Novell networks. Useful for its simplicity of configuration, this protocol (in its current implementation) does not scale real well to large networks, for reasons touched on in the section "IP vs. IPX".

IP

see Internet Protocol.

IPX

see Internetwork Packet eXchange (IPX).

ISP

see Internet Service Provider.

Layer 1

1. In OSI Model terms, the conceptual networking layer that defines electrical signaling on a wire.

Layer 2

1. In OSI Model terms, the conceptual networking layer that defines physical addressing and packetizing over a given wire.

Layer 3

1. In OSI Model terms, the conceptual networking layer that defines logical addressing and routing.

"MX" Record

1. A type of Domain Name System address, used specifically for Internet e-mail servers.
2. A DNS Mail eXchanger record, used to name-resolve email servers for purposes of mail delivery.

Name Resolution

see Domain Name Service.

Network

see Segment

Network Mask

1. The fixed part of the subnet mask determined by the RFC 950 "Class" of address.

Network Prefix

1. A special "bit mask" used to determine which computers are on the local network, and which aren't.
2. Synonymous with Subnet Mask in common usage.
3. Under RFC 1812, "Network Prefix" combines and replaces "Network Mask" and "Subnet Mask," both depreciated RFC 950 concepts.

Protocol Stack

1. Networking software.
2. The software used to communicate on a network using a given protocol, e.g., "I'm running Windows for Workgroups and using the Microsoft TCPIP-32 implementation of the TCPIP protocol stack." The word "stack" is derived from the layered nature of networking.

Router

1. An Internet traffic director, responsible for sending packets to their correct destinations.
2. A host, with multiple interfaces, that performs routing.
3. A device with multiple interfaces that forwards packets based on their Layer 3 address.

Routing

1. The process of sending packets to their correct destination across multiple networks.
2. The process of intelligently forwarding packets from network segment to network segment based on their Layer 3 address.

Segment

1. A network segment.
2. A subnet.
3. In this discussion, a "network segment" is a collection of hosts and/or layer 2 networking devices bounded by routers.

TCP/IP

1. Transmission Control Protocol/Internet Protocol

2. The "TCP/IP Protocol Suite" describes all of the various communications protocols of the Internet.
3. Commonly used when people actually mean "IP". (e.g., "What's the TCP/IP address of that workstation?")

WAN

see Wide Area Network

Wide Area Network (WAN)

1. A network that connects multiple non-contiguous sites.
2. Technically, the Internet can be considered to be the world's largest WAN.
3. A term generally used to describe any network that includes at least one dedicated link that involves paying the phone company.
4. Some people prefer to use the term Metropolitan Area Network (MAN) when a WAN doesn't connect sites in multiple cities, but the distinction is generally only made by the same people who get upset when they hear someone refer to a network card as a "NIC Card", since that would technically be redundant (as in, Network Interface Card Card.)

Top: Daryl's TCP/IP Primer
