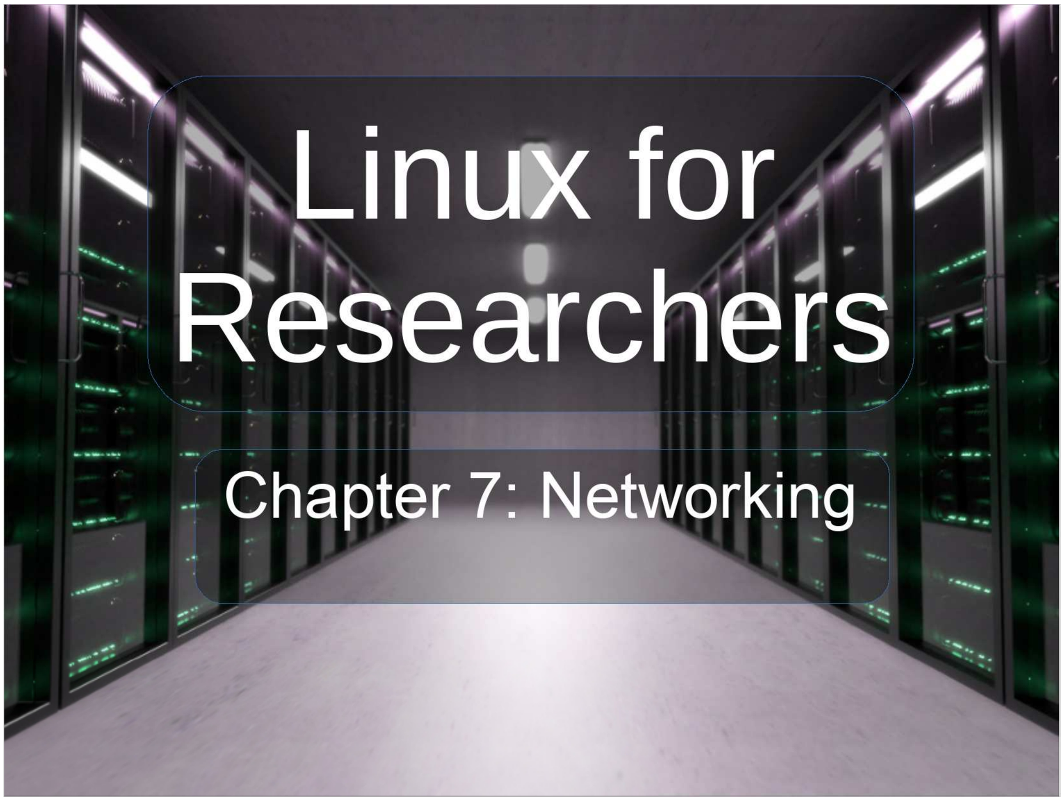


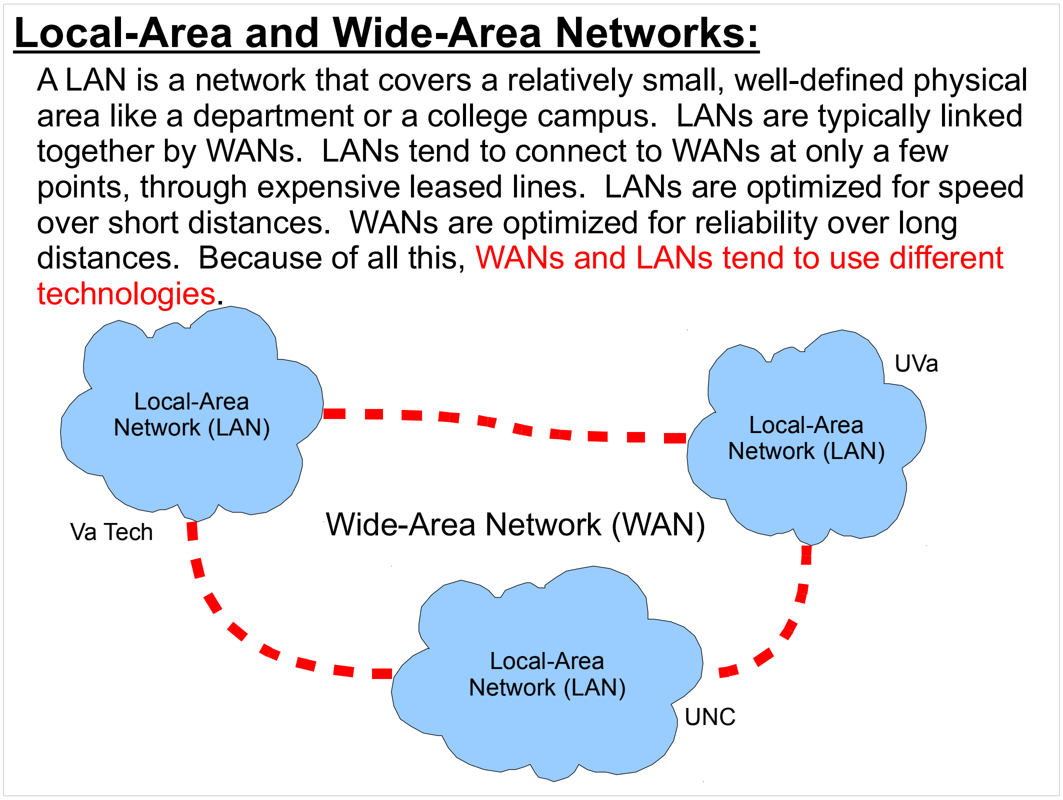
Linux for Researchers

Chapter 7: Networking

Up until now, we've taken networking for granted. Now let's take a look at how networking works in general, and some tools available under Linux for doing network-specific things.

Part 1: Network Hardware

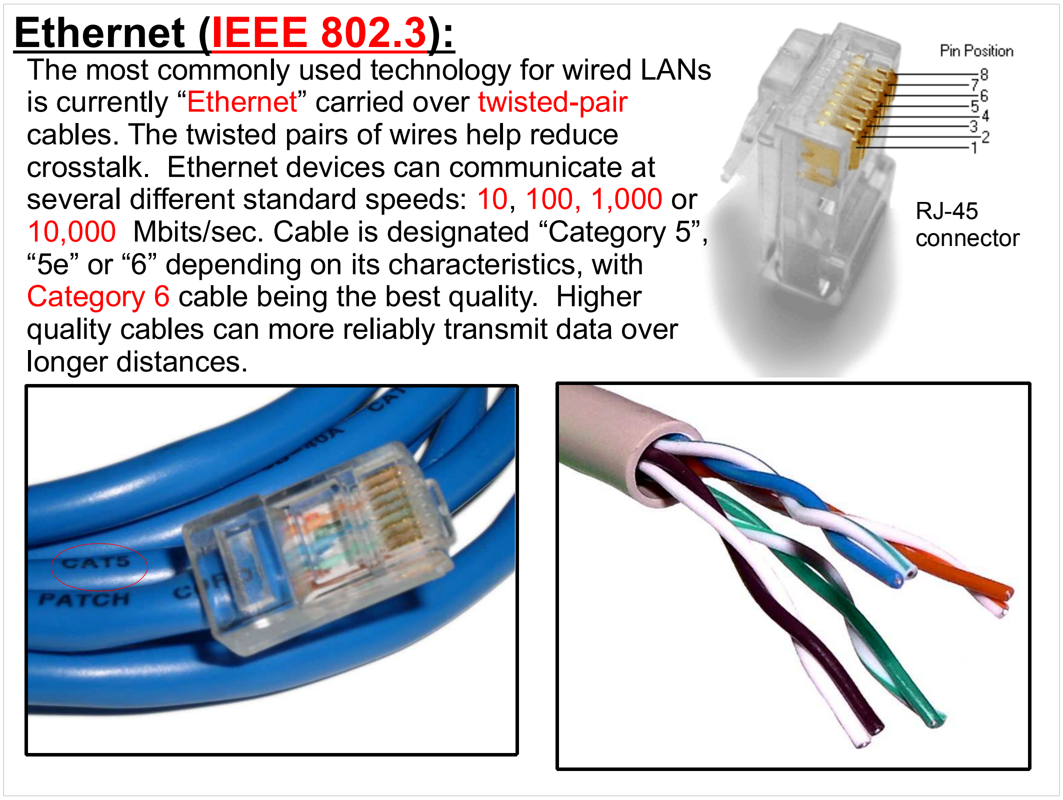
This is a broad topic, but I'll really only be talking about the most common type of network hardware, Ethernet over twisted-pair copper wire.



Local-Area and Wide-Area Networks:

A LAN is a network that covers a relatively small, well-defined physical area like a department or a college campus. LANs are typically linked together by WANs. LANs tend to connect to WANs at only a few points, through expensive leased lines. LANs are optimized for speed over short distances. WANs are optimized for reliability over long distances. Because of all this, WANs and LANs tend to use different technologies.

Local networks are created by local IT staff, or by hired contractors, who string cables and put in other network hardware as necessary. You can't usually connect two widely-separated networks this way, though. It would be unreasonable to send your IT people walking down the road from UVa to Va Tech, reeling out cable behind them, to create a link between the two universities. Instead, local networks are typically linked by leased lines, rented from phone companies or other providers. These companies have the infrastructure already in place to connect distant locations and maintain that connection.



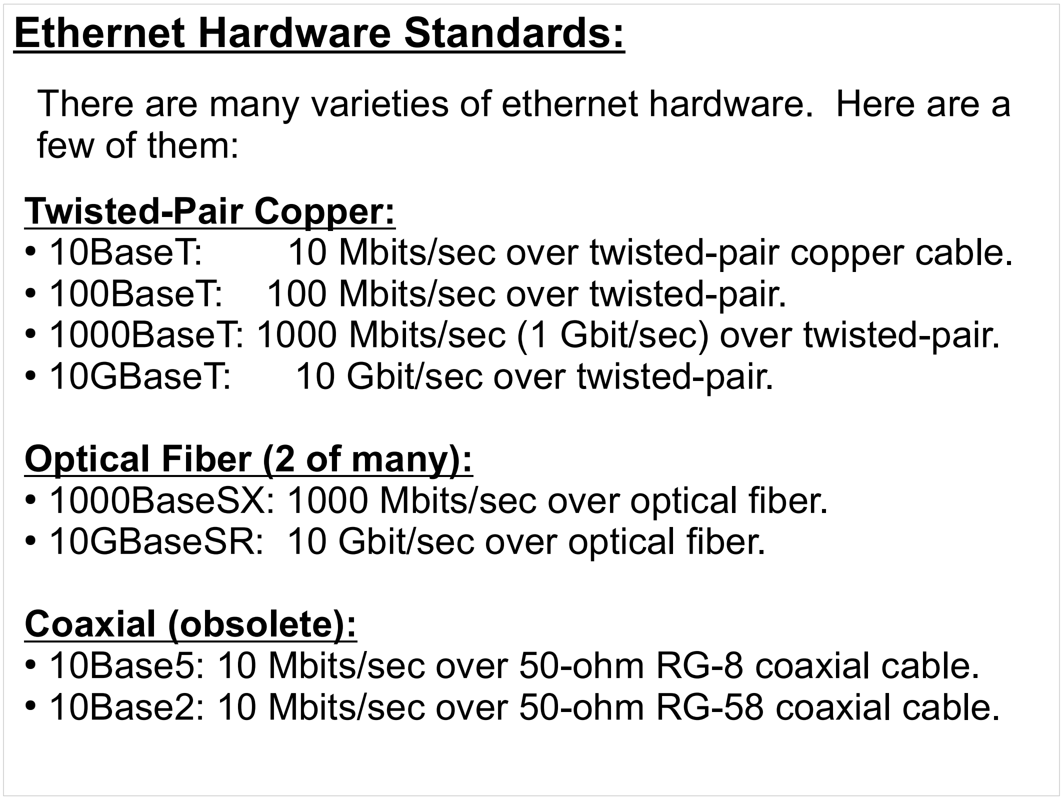
Ethernet (IEEE 802.3):

The most commonly used technology for wired LANs is currently "Ethernet" carried over twisted-pair cables. The twisted pairs of wires help reduce crosstalk. Ethernet devices can communicate at several different standard speeds: 10, 100, 1,000 or 10,000 Mbits/sec. Cable is designated "Category 5", "5e" or "6" depending on its characteristics, with Category 6 cable being the best quality. Higher quality cables can more reliably transmit data over longer distances.

The connectors at the ends of the cables are called RJ-45 connectors. They're a type of modular connector similar to modular telephone connectors (called RJ-11), but wider.

Each pair of wires carries signals. At the end of the line, the voltage difference between the two wires in the pair is measured. By measuring this differential voltage, noise that affects both wires equally is eliminated. The twists in the wires help ensure that sources of noise along the way do affect both wires equally. To help reduce crosstalk between adjacent twisted pairs, each pair in a cable has a different number of twists per meter.

This twisted-pair scheme has been in use since the early days of the phone system. It's well-understood technology, and cheap to produce.



Ethernet Hardware Standards:

There are many varieties of ethernet hardware. Here are a few of them:

Twisted-Pair Copper:

- 10BaseT: 10 Mbits/sec over twisted-pair copper cable.
- 100BaseT: 100 Mbits/sec over twisted-pair.
- 1000BaseT: 1000 Mbits/sec (1 Gbit/sec) over twisted-pair.
- 10GBaseT: 10 Gbit/sec over twisted-pair.

Optical Fiber (2 of many):

- 1000BaseSX: 1000 Mbits/sec over optical fiber.
- 10GBaseSR: 10 Gbit/sec over optical fiber.

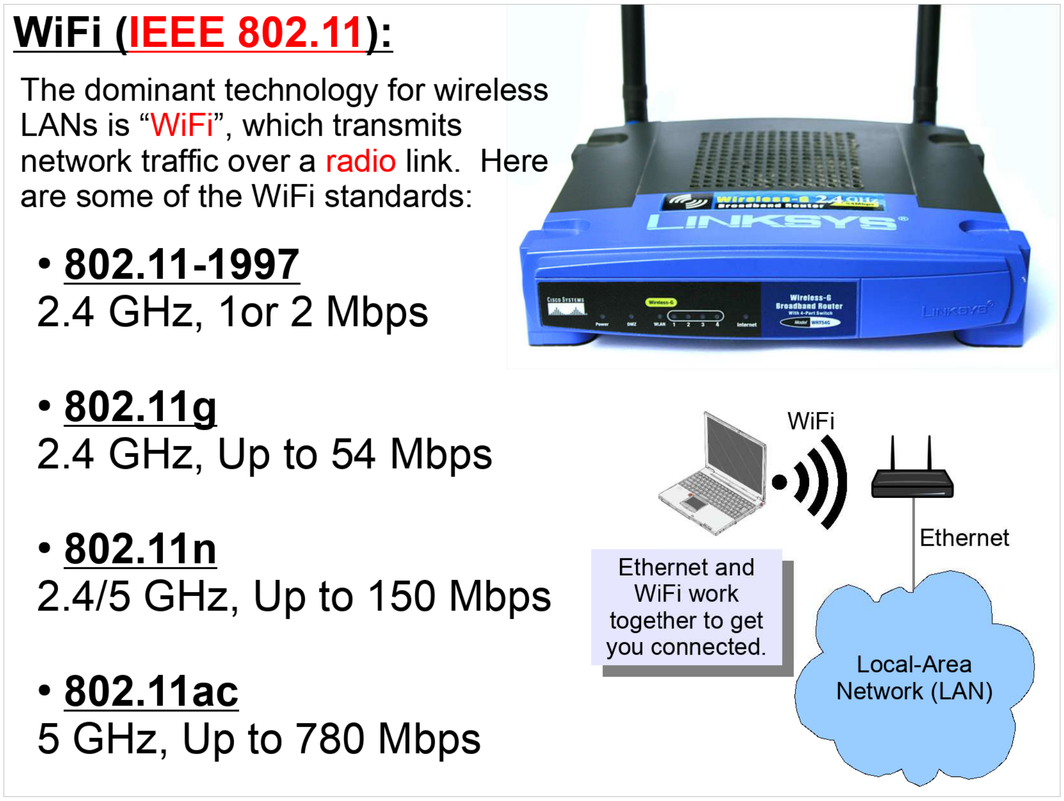
Coaxial (obsolete):

- 10Base5: 10 Mbits/sec over 50-ohm RG-8 coaxial cable.
- 10Base2: 10 Mbits/sec over 50-ohm RG-58 coaxial cable.

A note on nomenclature: The “base” in names like “10BaseT” comes from “baseband”, meaning that these transmission standards use a band of frequencies starting at zero and going up to some maximum, cutoff, frequency.

Optical fibers are better for long distances, but they're more expensive to deploy and maintain. They're usually used for connecting buildings together.

Typical network connections to desktop computers currently use either 10BaseT or 100BaseT Ethernet. 1000 Mbit/sec (Gigabit) Ethernet is often used between buildings or as the backplane of clusters of computers, but “Gigabit to the desktop” is becoming more common.



WiFi (IEEE 802.11):

The dominant technology for wireless LANs is "WiFi", which transmits network traffic over a radio link. Here are some of the WiFi standards:

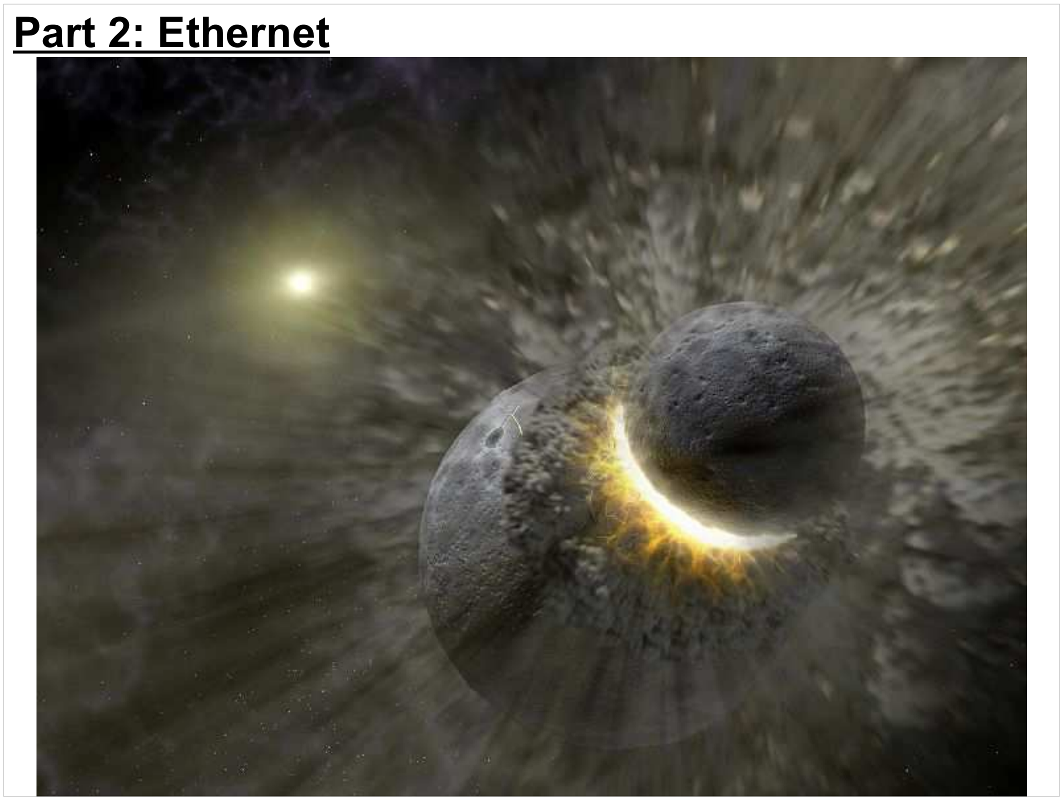
- **802.11-1997**
  2.4 GHz, 1or 2 Mbps
- **802.11g**
  2.4 GHz, Up to 54 Mbps
- **802.11n**
  2.4/5 GHz, Up to 150 Mbps
- **802.11ac**
  5 GHz, Up to 780 Mbps

Ethernet and WiFi work together to get you connected.

WiFi provides wireless connections to devices called Access Points. These access points are usually connected via Ethernet to the LAN. You could think of the wireless link as a way of wirelessly getting your laptop connected to the local Ethernet.

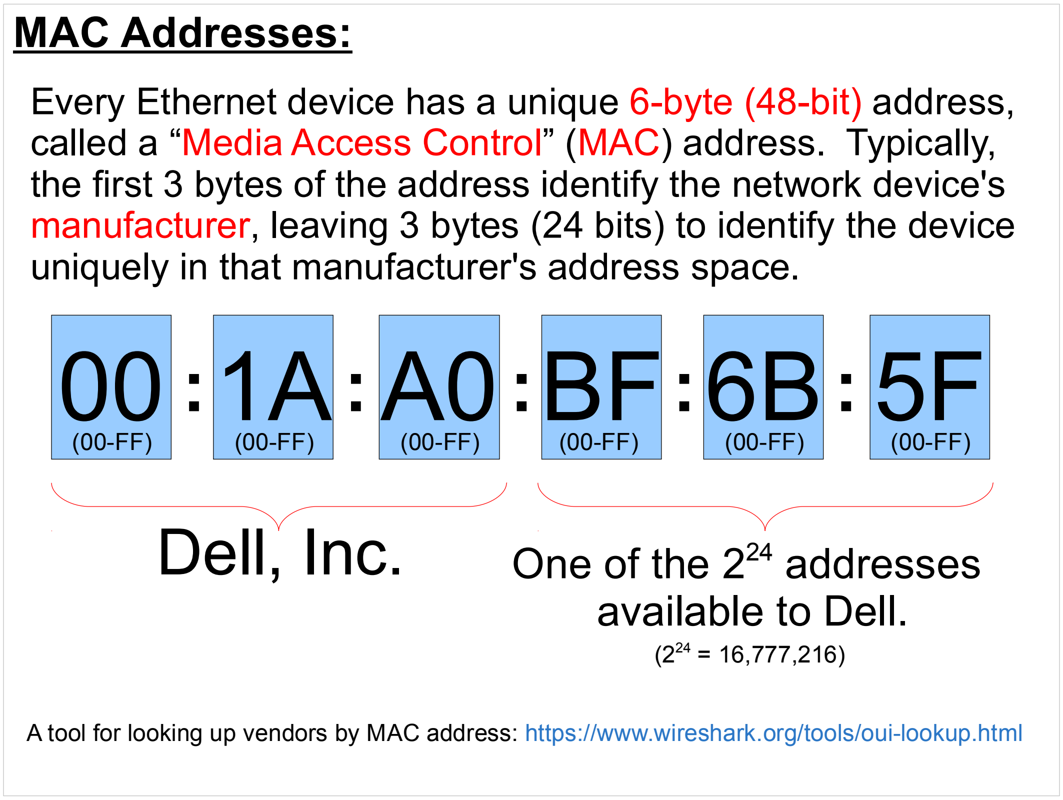
The wireless connection between your laptop and the access point is a replacement for a cable between the laptop and a wall outlet.

The WiFi and Ethernet protocols share a lot of features. In the following, I'm only going to discuss Ethernet specifically, but the concepts also apply to WiFi.



Part 2: Ethernet

Now that we've seen the hardware, let's look at how Ethernet works. As we'll see, collisions are an integral part of it.



MAC Addresses:

Every Ethernet device has a unique 6-byte (48-bit) address, called a "Media Access Control" (MAC) address. Typically, the first 3 bytes of the address identify the network device's manufacturer, leaving 3 bytes (24 bits) to identify the device uniquely in that manufacturer's address space.

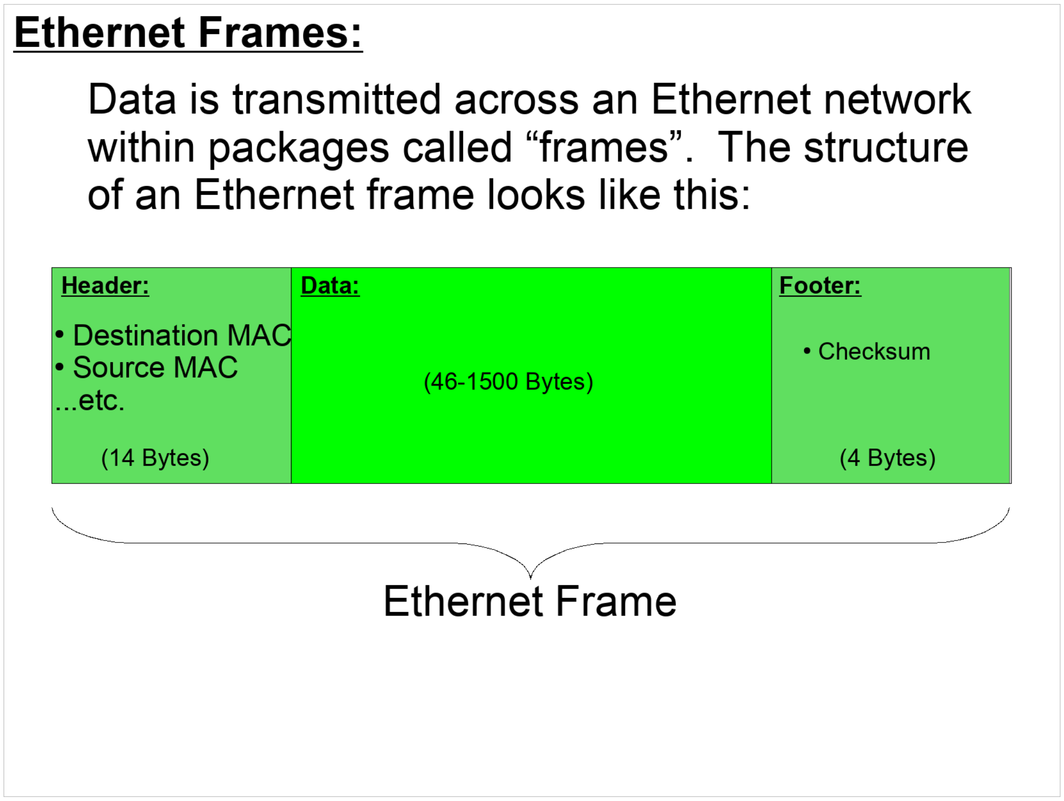
One of the 2^{24} addresses available to Dell.

(2^{24} = 16,777,216)

A tool for looking up vendors by MAC address: https://www.wireshark.org/tools/oui-lookup.html

A single manufacturer may actually be allocated several different 3-byte prefixes, especially a large manufacturer like Dell. There are plenty to go around: almost 17 million.

There's no expectation that we'll run short of MAC addresses anytime soon. The total number of them is about 280 trillion. This isn't the case with some other network addresses, as we'll see.



Ethernet Frames:

Data is transmitted across an Ethernet network within packages called “frames”. The structure of an Ethernet frame looks like this:

Header:

• Destination MAC
• Source MAC
...etc.

(14 Bytes)

Data:

(46-1500 Bytes)

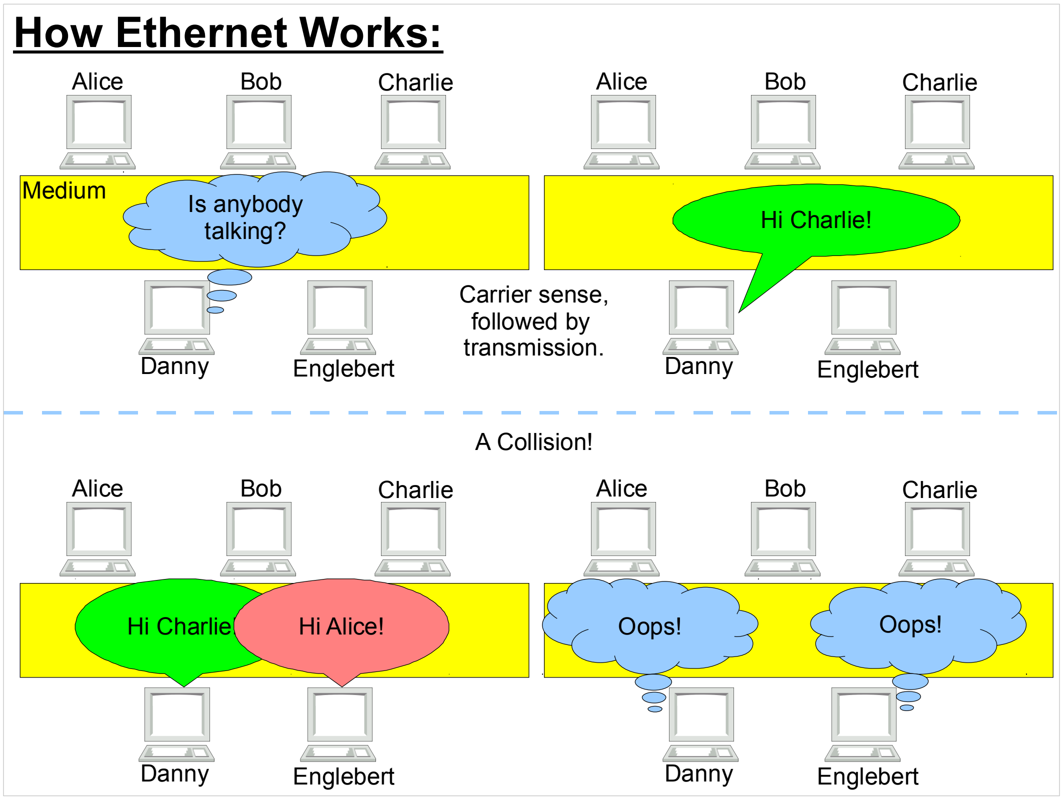
Footer:

• Checksum

(4 Bytes)

Ethernet Frame

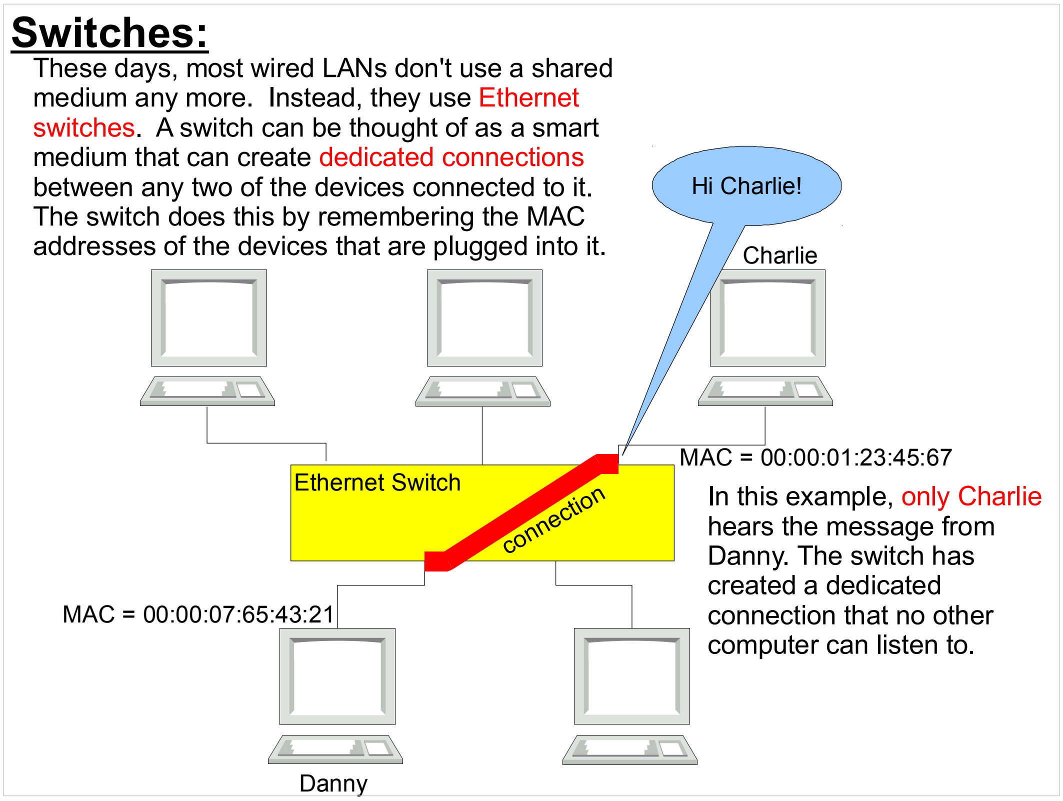
The checksum at the end of the frame is just a number computed from the frame's data. When the frame is composed, some function is applied to the data that produces a “hash”. The function is chosen so that any small change in the data will produce a different hash. This hash is stored as the checksum value in the frame's footer. When the frame arrives, the same function is applied again to the data. If the resulting hash doesn't match the hash stored in the checksum, then the receiver knows that the frame has been mangled during transmission.



You can think of the original Ethernet design as a bunch of rooms along a corridor. Whenever one of the occupants (a computer) wants to communicate with another, it shouts the message down the corridor (a "shared medium"). Everyone except the intended recipient ignores the message.

Originally, Ethernet used a design called "Carrier-Sense, Multiple Access with Collision Detection" (CSMA/CD). When an Ethernet device wants to start talking, it first listens to see if the shared medium is free, then it transmits its message. If two devices talk at the same time, that's a "collision", and each device shuts up for some random time, before trying again. Early on, this simple-minded system was (amazingly) shown to perform much better than more sophisticated networking schemes.

It's important to remember that collisions are a natural part of the way Ethernet works. There's nothing wrong with a few of them.



Switches:

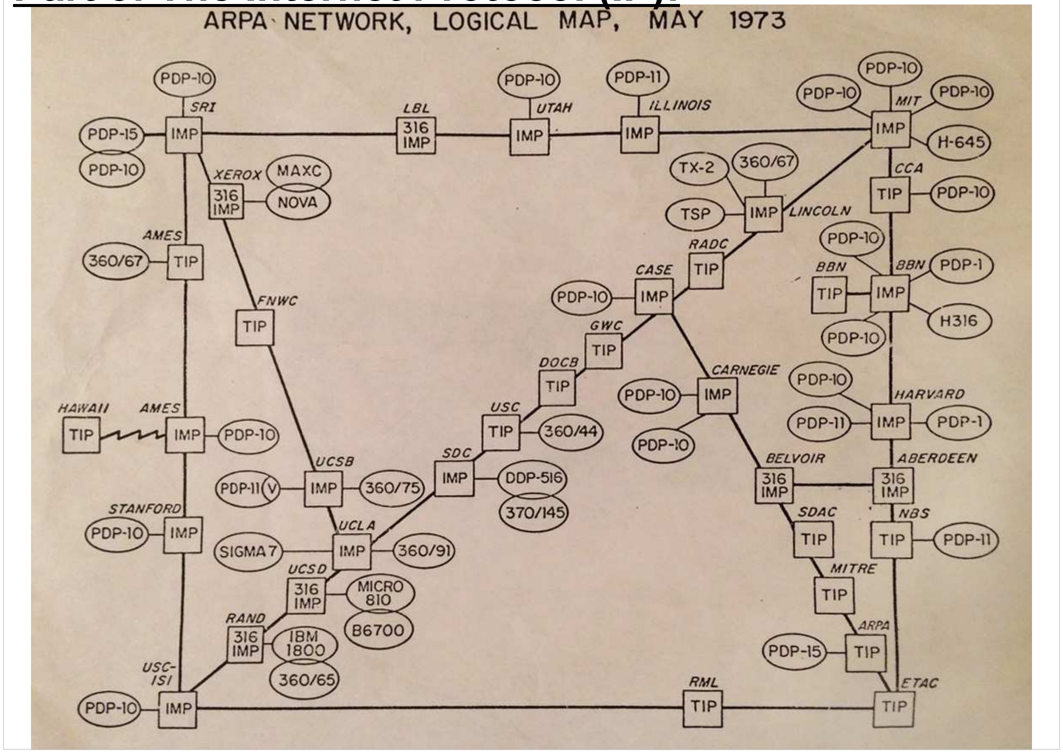
These days, most wired LANs don't use a shared medium any more. Instead, they use Ethernet switches. A switch can be thought of as a smart medium that can create dedicated connections between any two of the devices connected to it. The switch does this by remembering the MAC addresses of the devices that are plugged into it.

In this example, only Charlie hears the message from Danny. The switch has created a dedicated connection that no other computer can listen to.

The switch does its work by remembering the MAC addresses of the devices that are plugged into it. For each connector, the switch maintains a list of the MAC addresses it has recently seen talking on that connection.

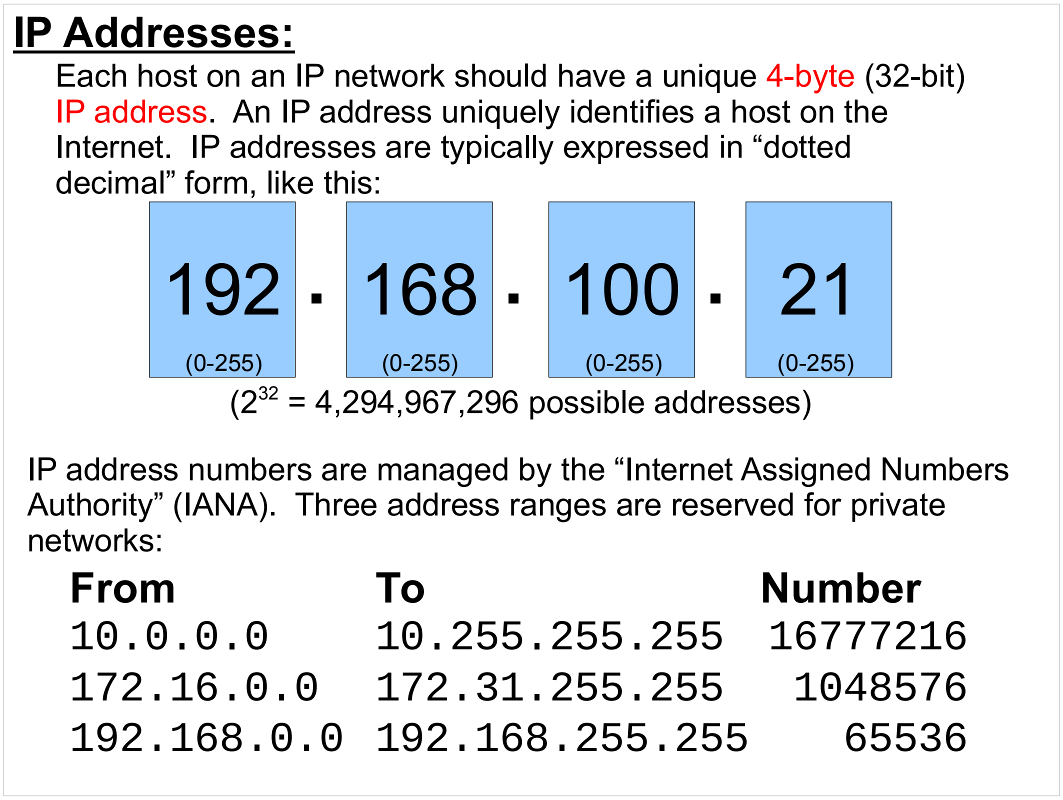
Some traffic still needs to be broadcast, though. (ARP packets, for example, as we'll see.) The switch will send broadcast traffic to all devices connected to it.

Part 3: The Internet Protocol (IP):



In the 1960s the Advanced Research Projects Agency (ARPA) of the Department of Defense began building a nationwide network called “ARPANET”. The first link (at 50 Kbits/sec) was created between UCLA and Stanford in 1969. The first message sent was “lo”! It was supposed to be “login”, but the network crashed after the first two letters! ARPANET grew into the Internet we know now.

A new protocol, called “Internet Protocol” (IP) was invented for transmitting data on the Internet. This protocol was intended to be layered on top of a variety of underlying protocols. Thus, IP was independent of the details of a site's local network hardware. The Internet Protocol was capable of binding together a heterogeneous collection of computers around the world.



IP Addresses:

Each host on an IP network should have a unique 4-byte (32-bit) IP address. An IP address uniquely identifies a host on the Internet. IP addresses are typically expressed in “dotted decimal” form, like this:

192 . 168 . 100 . 21
(0-255)
(0-255)
(0-255)
(0-255)

(2^{32} = 4,294,967,296 possible addresses)

IP address numbers are managed by the “Internet Assigned Numbers Authority” (IANA). Three address ranges are reserved for private networks:

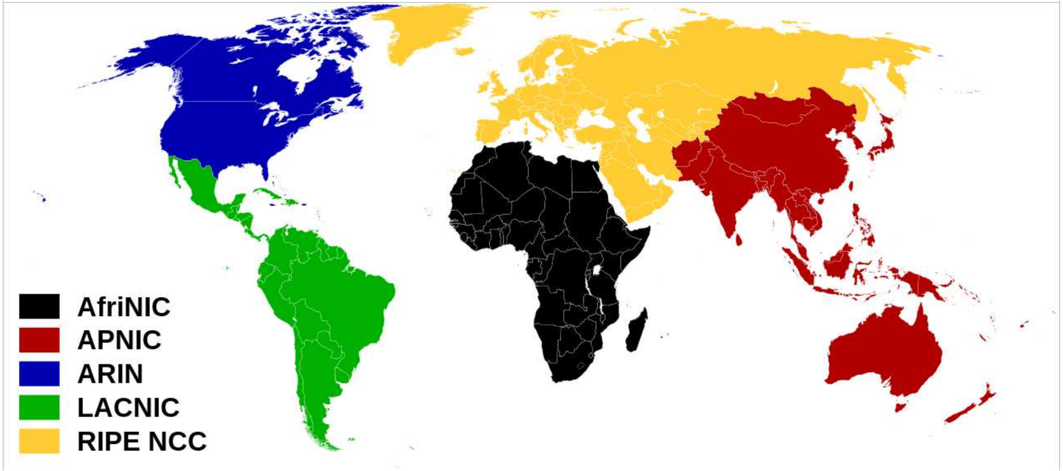
From	To	Number
10.0.0.0	10.255.255.255	16777216
172.16.0.0	172.31.255.255	1048576
192.168.0.0	192.168.255.255	65536

Note that I use the term “host” here instead of computer. By “host”, I mean “a thing with an IP address”. Usually, there will be a one-to-one mapping between IP addresses and computers, but not always by any means.

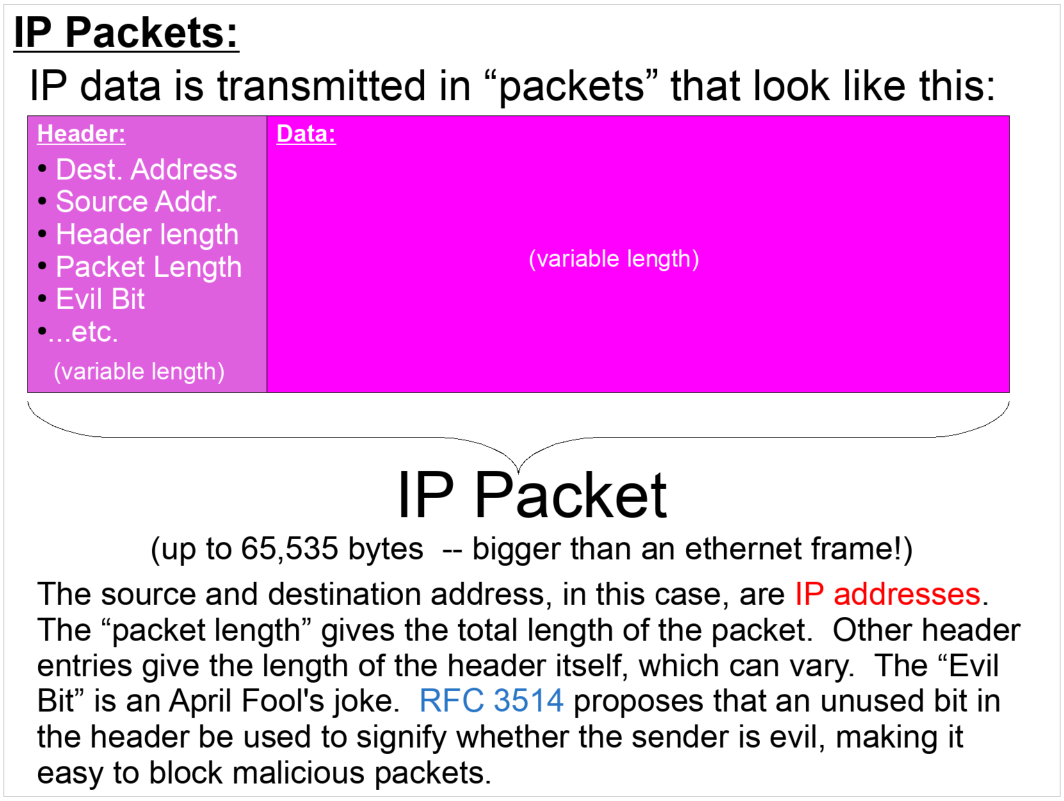
The IANA hands out blocks of addresses through regional registrars to internet service providers (ISPs). Each ISP is approved by the regional registrar, and must pay an annual fee to retain its IP addresses. UVa owns several address ranges: 128.143.*.*, 137.54.*.* and portions of the 199.111.*.* address space. The annual fee for a two-byte address range (called a /16 or a “Class B” network) like 128.143.*.* is $4,500.

Finally, note that everything I'm going to say about IP applies to the current version, IPv4. The successor, IPv6, is on its way, but probably won't affect you for a few more years. IPv6 has a much larger address space and a different notation.

IANA Regional Internet Registries:



ARIN = “American Registry for Internet Numbers”



IP Packets:

IP data is transmitted in "packets" that look like this:

Header:

- Dest. Address
- Source Addr.
- Header length
- Packet Length
- Evil Bit
- ...etc.

(variable length)

Data:

(variable length)

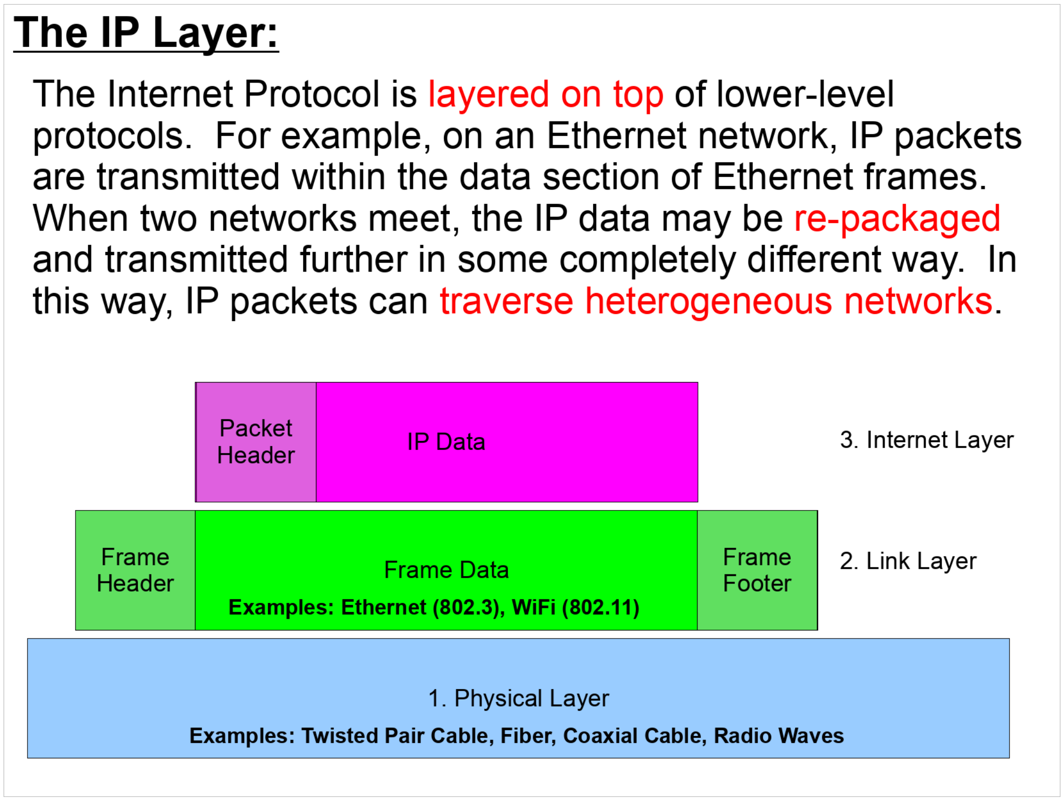
IP Packet

(up to 65,535 bytes -- bigger than an ethernet frame!)

The source and destination address, in this case, are IP addresses. The "packet length" gives the total length of the packet. Other header entries give the length of the header itself, which can vary. The "Evil Bit" is an April Fool's joke. RFC 3514 proposes that an unused bit in the header be used to signify whether the sender is evil, making it easy to block malicious packets.

Notice that an IP packet can have a maximum size of 65,535 bytes, but an Ethernet frame can only carry up to 1,500 bytes of data. When a large IP packet arrives at an Ethernet network, it may be broken up into smaller packets ("fragments") which are sent in separate Ethernet frames. The IP headers of these fragments will contain the information necessary to reassemble them later.

The maximum size of the underlying layer's "chunks" is called the "Maximum Transmission Unit" (MTU).



The IP Layer:

The Internet Protocol is layered on top of lower-level protocols. For example, on an Ethernet network, IP packets are transmitted within the data section of Ethernet frames. When two networks meet, the IP data may be re-packaged and transmitted further in some completely different way. In this way, IP packets can traverse heterogeneous networks.

Packet Header

IP Data

3. Internet Layer

Frame Header

Frame Data

Examples: Ethernet (802.3), WiFi (802.11)

Frame Footer

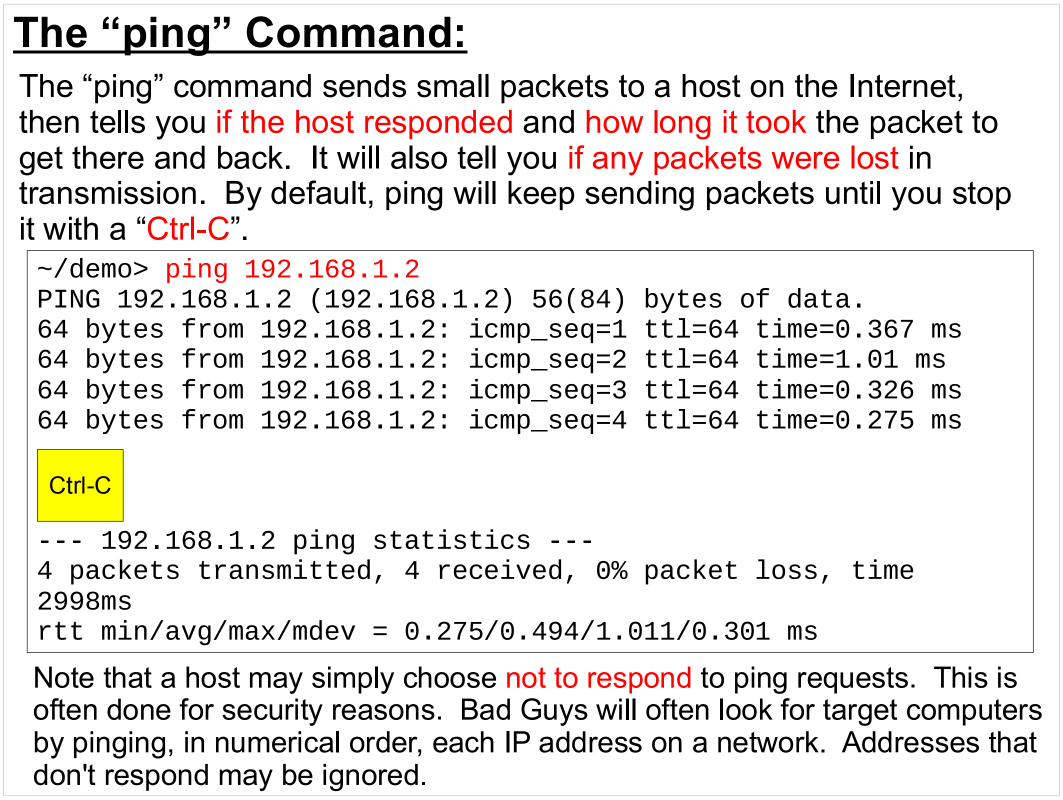
2. Link Layer

1. Physical Layer

Examples: Twisted Pair Cable, Fiber, Coaxial Cable, Radio Waves

We're starting to build up a stack of protocols, each of which adds features unavailable at lower levels. The physical layer gives us a mechanism for transmitting zeros and ones. The “link” layer (e.g., Ethernet) gives us a way to send a set of zeros and ones to a particular computer. The Internet layer gives us a way to transmit data across a heterogeneous network. We'll add two more layers before we're done.

You can imagine IP packets travelling across the network the way you'd take a plane trip. First, you get in your car and drive to the airport. There, you board a small plane and fly to another airport, where you get on a big plane and fly somewhere else. Once there, you rent a car and drive to your final destination. Just like you, IP packets can travel in a variety of vehicles on their way from one computer to another. Sometimes they'll be contained in Ethernet packets, sometimes they'll travel over other types of network. But eventually they'll arrive a their destination.



The “ping” Command:

The “ping” command sends small packets to a host on the Internet, then tells you if the host responded and how long it took the packet to get there and back. It will also tell you if any packets were lost in transmission. By default, ping will keep sending packets until you stop it with a “Ctrl-C”.

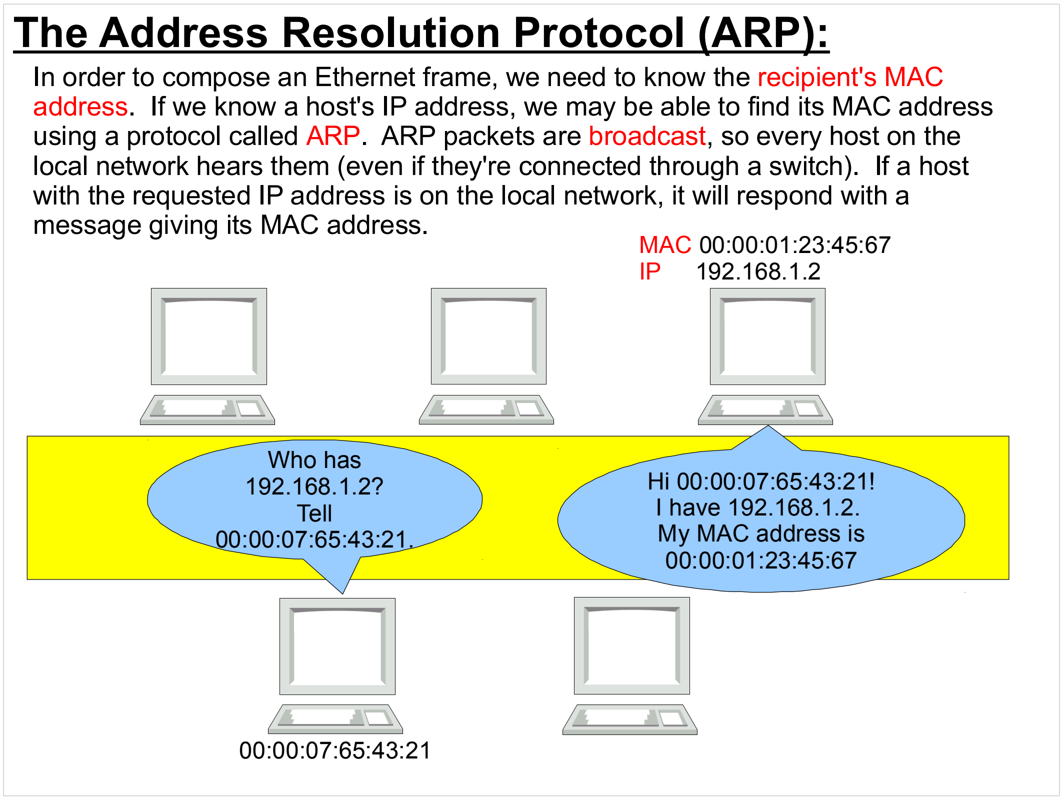
```
~/demo> ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.367 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=1.01 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.326 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.275 ms

Ctrl-C

--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time
2998ms
rtt min/avg/max/mdev = 0.275/0.494/1.011/0.301 ms
```

Note that a host may simply choose not to respond to ping requests. This is often done for security reasons. Bad Guys will often look for target computers by pinging, in numerical order, each IP address on a network. Addresses that don't respond may be ignored.

One way to tell your computer to ignore pings is through a command like this:

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

We'll talk about the “/proc” filesystem more later. In addition to giving you inormation about processes (as we've already seen), It also gives you access to many tunable parameters of the Linux kernel.

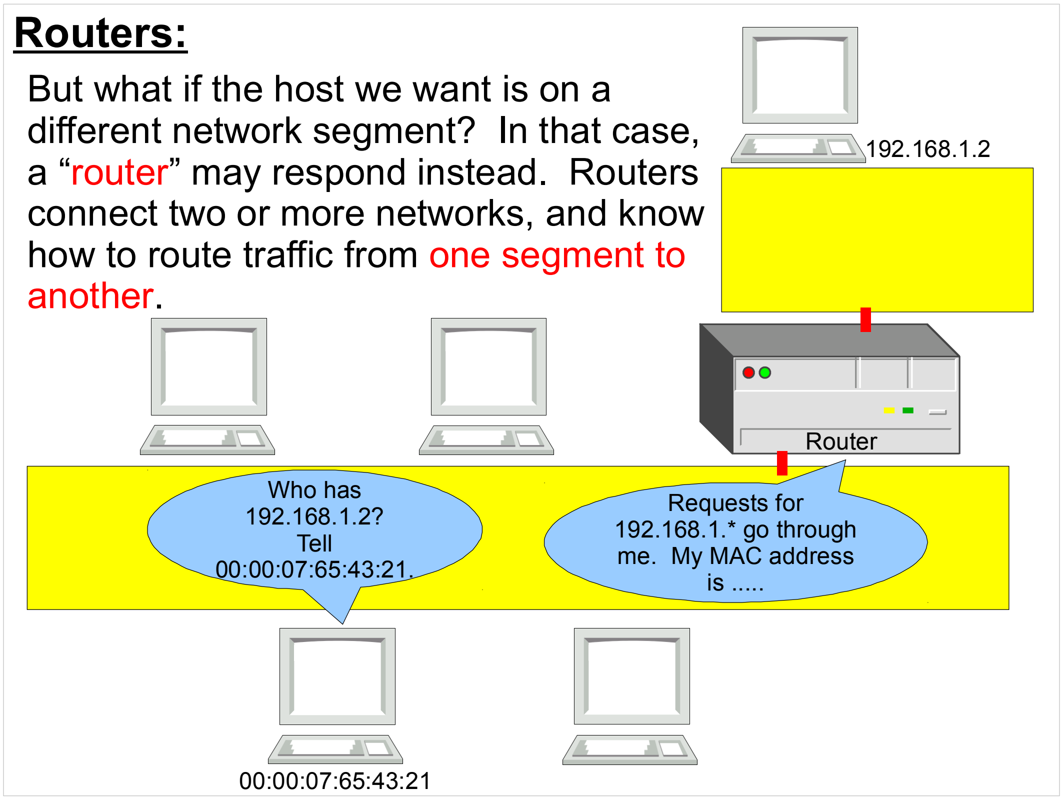


The Address Resolution Protocol (ARP):

In order to compose an Ethernet frame, we need to know the recipient's MAC address. If we know a host's IP address, we may be able to find its MAC address using a protocol called ARP. ARP packets are broadcast, so every host on the local network hears them (even if they're connected through a switch). If a host with the requested IP address is on the local network, it will respond with a message giving its MAC address.

MAC 00:00:01:23:45:67
IP 192.168.1.2
Who has 192.168.1.2? Tell 00:00:07:65:43:21.
Hi 00:00:07:65:43:21! I have 192.168.1.2. My MAC address is 00:00:01:23:45:67
00:00:07:65:43:21

To broadcast a request on an Ethernet network, you send it to the special MAC address "FF:FF:FF:FF:FF:FF".



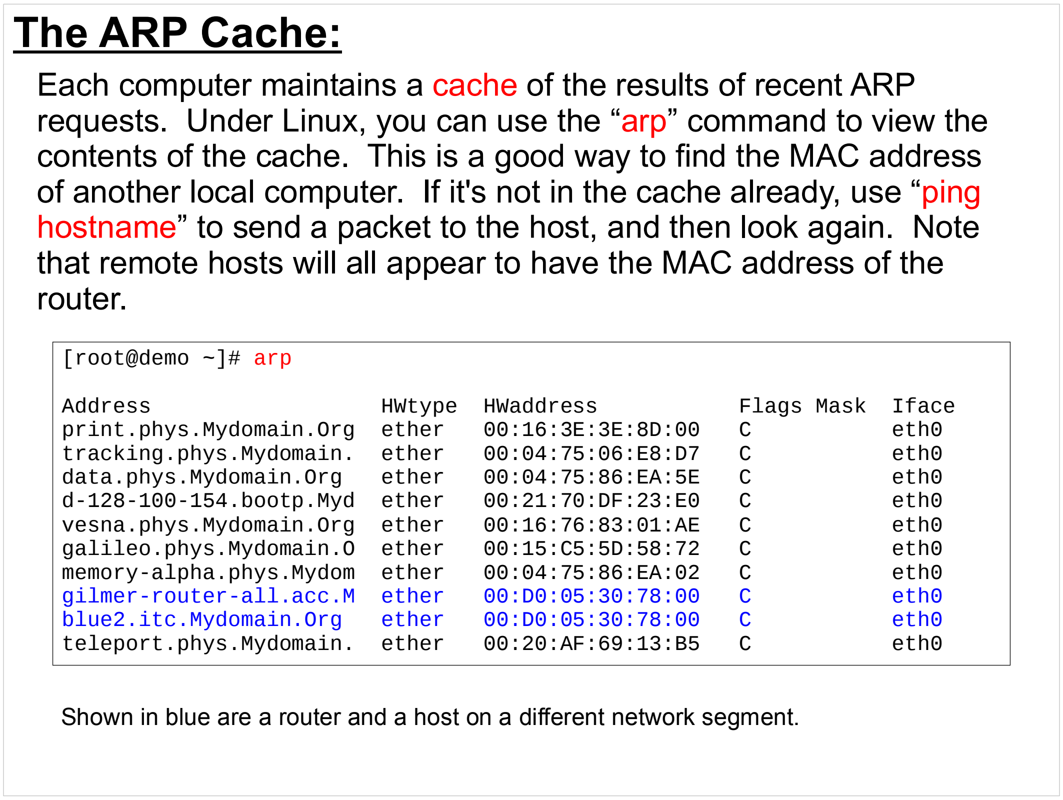
Routers:

But what if the host we want is on a different network segment? In that case, a “router” may respond instead. Routers connect two or more networks, and know how to route traffic from one segment to another.

192.168.1.2
Router
Who has 192.168.1.2? Tell 00:00:07:65:43:21.
Requests for 192.168.1.* go through me. My MAC address is .....
00:00:07:65:43:21

A network “segment” is a section of a network that is connected at the physical layer. Traffic can travel between all computers on the same segment without the help of routers or other intermediate devices that understand higher-level protocols.

When a router hears an ARP request for a host on one of the other segments it's connected to, the router responds with its own MAC address. The sending computer then knows to send Ethernet frames to the router, and the router will pass them along to the other network segment, doing its own ARPs there to find the destination host.



The ARP Cache:

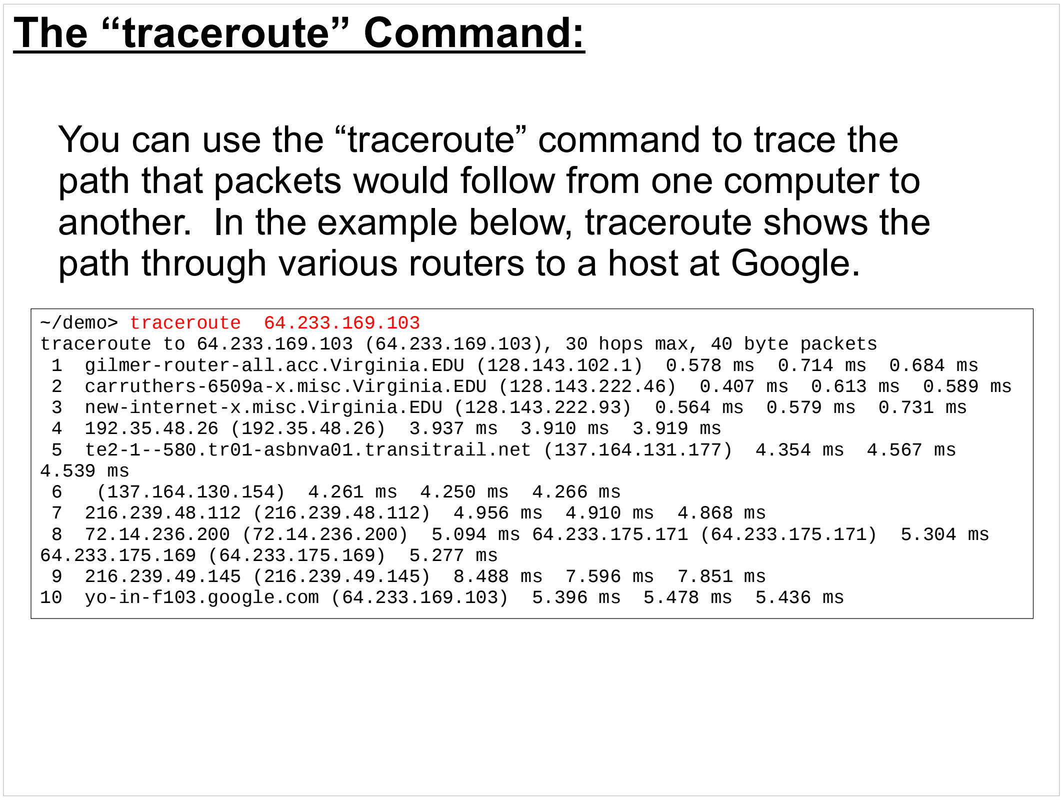
Each computer maintains a cache of the results of recent ARP requests. Under Linux, you can use the “arp” command to view the contents of the cache. This is a good way to find the MAC address of another local computer. If it's not in the cache already, use “ping hostname” to send a packet to the host, and then look again. Note that remote hosts will all appear to have the MAC address of the router.

```
[root@demo ~]# arp

Address                  HWtype  HWaddress           Flags Mask  Iface
print.phys.Mydomain.Org  ether   00:16:3E:3E:8D:00   C           eth0
tracking.phys.Mydomain.  ether   00:04:75:06:E8:D7   C           eth0
data.phys.Mydomain.Org   ether   00:04:75:86:EA:5E   C           eth0
d-128-100-154.bootp.Myd  ether   00:21:70:DF:23:E0   C           eth0
vesna.phys.Mydomain.Org  ether   00:16:76:83:01:AE   C           eth0
galileo.phys.Mydomain.O  ether   00:15:C5:5D:58:72   C           eth0
memory-alpha.phys.Mydom  ether   00:04:75:86:EA:02   C           eth0
gilmer-router-all.acc.M  ether   00:D0:05:30:78:00   C           eth0
blue2.itc.Mydomain.Org   ether   00:D0:05:30:78:00   C           eth0
teleport.phys.Mydomain.  ether   00:20:AF:69:13:B5   C           eth0
```

Shown in blue are a router and a host on a different network segment.

One of those things you'll probably never need to do, but it's possible anyway:

You can also use the “arp” command to manually manipulate the ARP cache. This is sometimes necessary for configuring network devices that are only accessible through the local network segment (i.e., they don't have a keyboard or any other way of configuring them locally). Using the appropriate arp command, you can manually enter a MAC address into the ARP cache and associate it with an IP address. You can then use that IP address to communicate with the remote device.

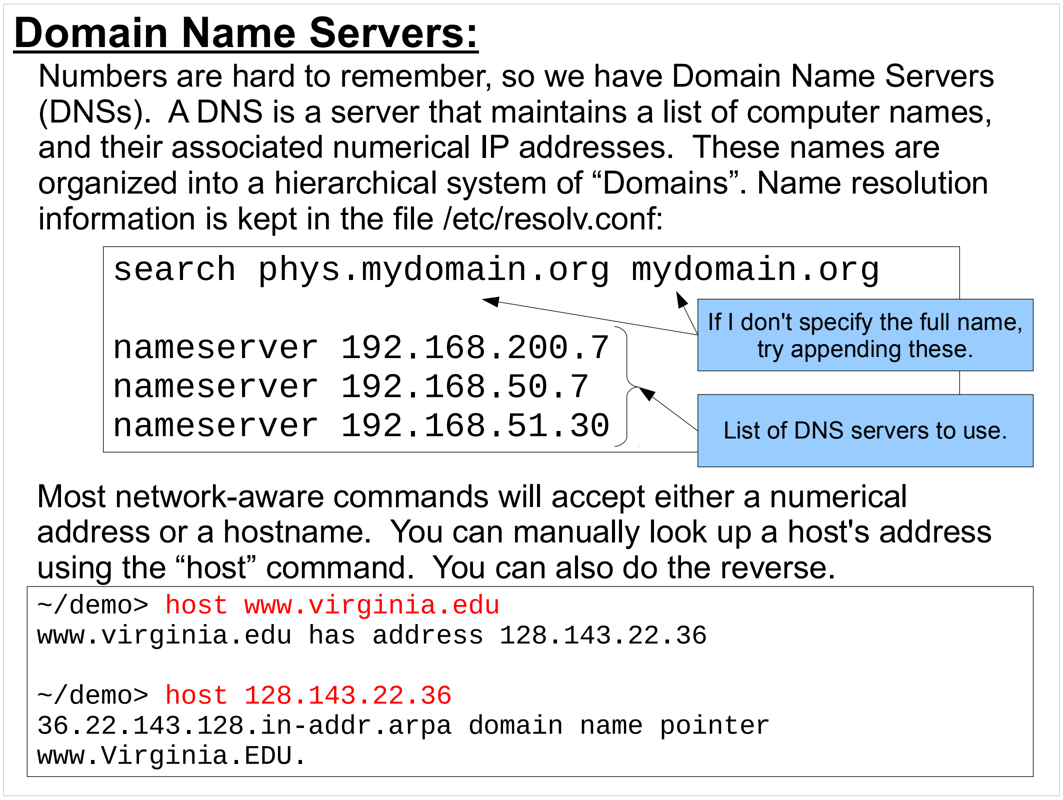


The “traceroute” Command:

You can use the “traceroute” command to trace the path that packets would follow from one computer to another. In the example below, traceroute shows the path through various routers to a host at Google.

```
~/demo> traceroute  64.233.169.103
traceroute to 64.233.169.103 (64.233.169.103), 30 hops max, 40 byte packets
 1  gilmer-router-all.acc.Virginia.EDU (128.143.102.1)  0.578 ms  0.714 ms  0.684 ms
 2  carruthers-6509a-x.misc.Virginia.EDU (128.143.222.46)  0.407 ms  0.613 ms  0.589 ms
 3  new-internet-x.misc.Virginia.EDU (128.143.222.93)  0.564 ms  0.579 ms  0.731 ms
 4  192.35.48.26 (192.35.48.26)  3.937 ms  3.910 ms  3.919 ms
 5  te2-1--580.tr01-asbnva01.transitrail.net (137.164.131.177)  4.354 ms  4.567 ms  
4.539 ms
 6   (137.164.130.154)  4.261 ms  4.250 ms  4.266 ms
 7  216.239.48.112 (216.239.48.112)  4.956 ms  4.910 ms  4.868 ms
 8  72.14.236.200 (72.14.236.200)  5.094 ms 64.233.175.171 (64.233.175.171)  5.304 ms  
64.233.175.169 (64.233.175.169)  5.277 ms
 9  216.239.49.145 (216.239.49.145)  8.488 ms  7.596 ms  7.851 ms
10  yo-in-f103.google.com (64.233.169.103)  5.396 ms  5.478 ms  5.436 ms
```

Note that traceroute can't always identify all of the routers along the way. Firewall rules may prevent some intermediate hosts from responding to traceroute's queries.



Domain Name Servers:

Numbers are hard to remember, so we have Domain Name Servers (DNSs). A DNS is a server that maintains a list of computer names, and their associated numerical IP addresses. These names are organized into a hierarchical system of “Domains”. Name resolution information is kept in the file /etc/resolv.conf:

```
search phys.mydomain.org mydomain.org

nameserver 192.168.200.7
nameserver 192.168.50.7
nameserver 192.168.51.30
```

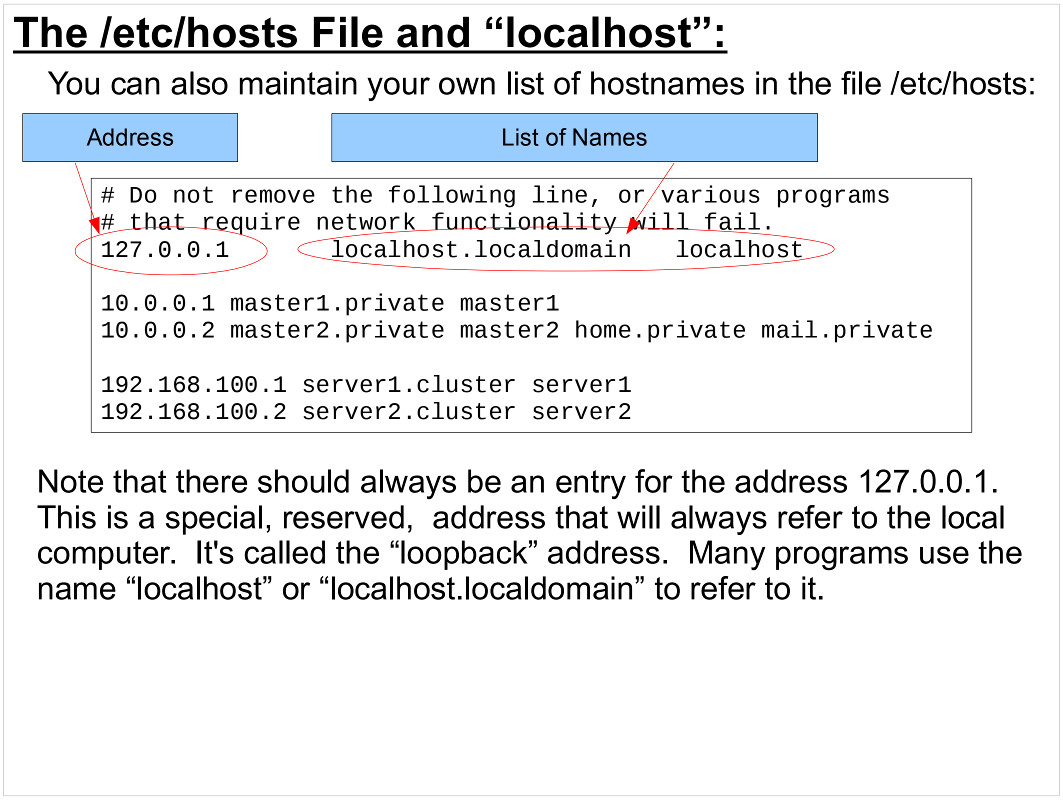
If I don't specify the full name, try appending these.

List of DNS servers to use.

Most network-aware commands will accept either a numerical address or a hostname. You can manually look up a host's address using the “host” command. You can also do the reverse.

```
~/demo> host www.virginia.edu
www.virginia.edu has address 128.143.22.36

~/demo> host 128.143.22.36
36.22.143.128.in-addr.arpa domain name pointer
www.Virginia.EDU.
```

Underneath a few top-level domains (TLDs) like “.com” and “.edu”, secondary domain names are given out to individuals or institutions, on a first-come-first-serve basis, through domain registrars. Registrars typically charge an annual fee for retaining a domain name. The annual fee for a domain name is typically only a few dollars.



The /etc/hosts File and “localhost”:

You can also maintain your own list of hostnames in the file /etc/hosts:

Address
List of Names

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost

10.0.0.1 master1.private master1
10.0.0.2 master2.private master2 home.private mail.private

192.168.100.1 server1.cluster server1
192.168.100.2 server2.cluster server2
```

Note that there should always be an entry for the address 127.0.0.1. This is a special, reserved, address that will always refer to the local computer. It's called the “loopback” address. Many programs use the name “localhost” or “localhost.localdomain” to refer to it.

Applications will usually check the /etc/hosts file first, and then go to the DNS servers when trying to resolve a name. The order is configurable in /etc/nsswitch.conf. Type “man nsswitch.conf” for more information.

Part 4: Network Interfaces

A “network interface” is the device inside your computer that is actually plugged into the local network. This may be a card plugged into an expansion slot inside your computer, but typically these days the network interface is built into the computer's motherboard. For most computers, the network interface will be the thing in the back of your computer into which you plug an RJ-45 connector.

The picture above shows an old acoustic coupler, of the type I used when I was in high school in the 1970s. This was an early way of connecting computers together. Data was sent over a phone line as a series of tones, and these were decoded by devices like the on above and turned into digital signals.

Network Interfaces:

To get a list of your computer's network interfaces, use the “ifconfig” command. Normally, you'll see at least two interfaces:

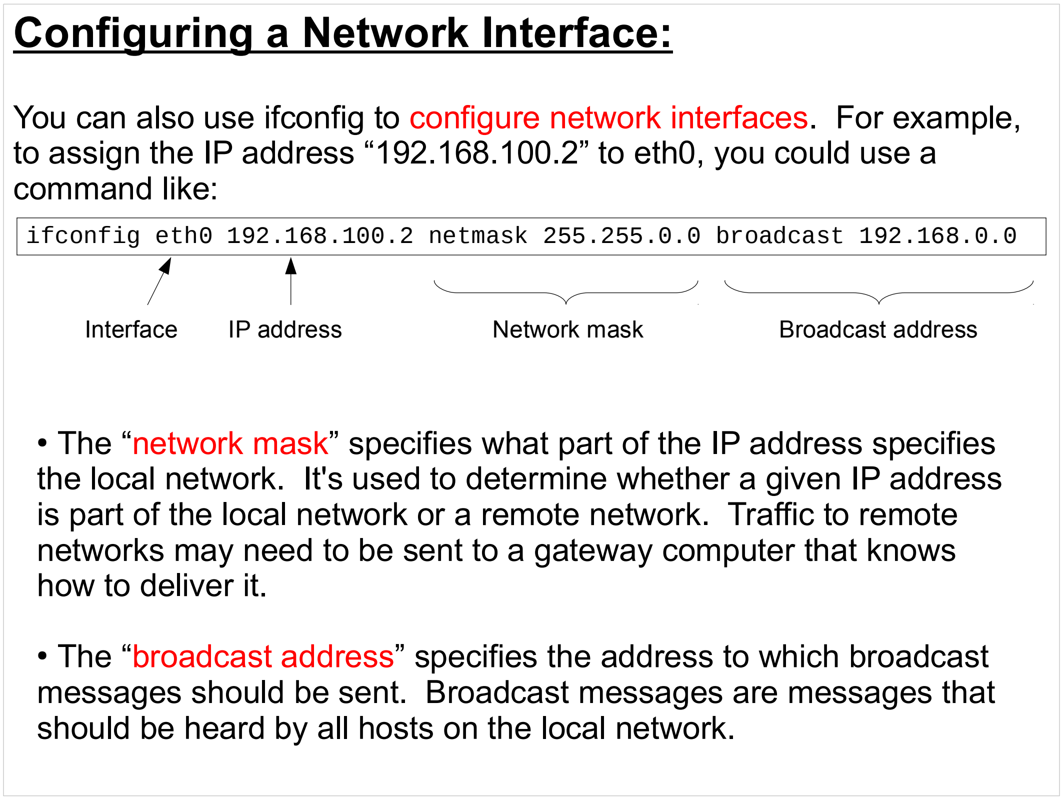
```
[root@demo ~]# ifconfig
eth0  Link encap:Ethernet  HWaddr 00:1A:A0:BF:6B:5F
      inet addr:192.168.100.2  Bcast:192.168.255.255  Mask:255.255.0.0
      inet6 addr: fe80::21a:a0ff:febf:6b5f/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:78617868 errors:0 dropped:0 overruns:0 frame:0
      TX packets:25924911 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:725202911 (691.6 MiB)  TX bytes:1837757879 (1.7 GiB)

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:10774315 errors:0 dropped:0 overruns:0 frame:0
      TX packets:10774315 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:4152450841 (3.8 GiB)  TX bytes:4152450841 (3.8 GiB)
```

The interface “eth0” is an ethernet interface. Ifconfig shows its MAC address and IP address. The interface called “lo” is the “loopback” interface. It's used when a network-aware program wants to talk to the same computer.

Programs could talk to the local computer through eth0, too, but there are some disadvantages:

- There's more overhead involved. To talk on eth0 the system needs to go through the whole process of composing a message on the network, sending it out on the network, then hearing it and interpreting it.

- There may be firewall rules applied to eth0 that you don't want to apply to purely internal communications.



Configuring a Network Interface:

You can also use ifconfig to configure network interfaces. For example, to assign the IP address “192.168.100.2” to eth0, you could use a command like:

```
ifconfig eth0 192.168.100.2 netmask 255.255.0.0 broadcast 192.168.0.0
```

Interface

IP address

Network mask

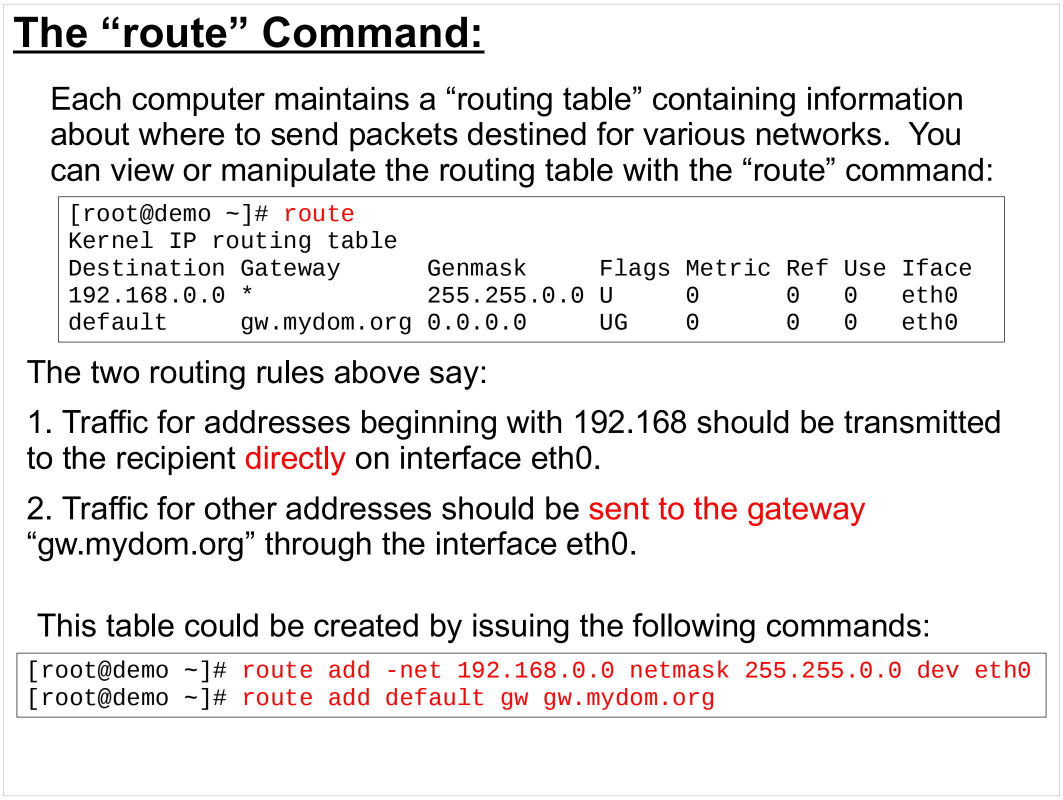
Broadcast address

- The “network mask” specifies what part of the IP address specifies the local network. It's used to determine whether a given IP address is part of the local network or a remote network. Traffic to remote networks may need to be sent to a gateway computer that knows how to deliver it.

- The “broadcast address” specifies the address to which broadcast messages should be sent. Broadcast messages are messages that should be heard by all hosts on the local network.

You can actually have more than one IP address on a single interface. You can use ifconfig to add extra addresses to an interface by specifying “aliases” for the interface. These have names like “eth0:0”, “eth0:1”, “eth0:2”, etc. Once you've configured the interface with its first address, as above, you can add others like this:

```
ifconfig eth0:0 192.168.100.50
ifconfig eth0:1 192.168.100.38
ifconfig eth0:2 192.168.100.5
```



The “route” Command:

Each computer maintains a “routing table” containing information about where to send packets destined for various networks. You can view or manipulate the routing table with the “route” command:

```
[root@demo ~]# route
Kernel IP routing table
Destination Gateway      Genmask     Flags Metric Ref Use Iface
192.168.0.0 *            255.255.0.0 U     0      0   0   eth0
default     gw.mydom.org 0.0.0.0     UG    0      0   0   eth0
```

The two routing rules above say:

1. Traffic for addresses beginning with 192.168 should be transmitted to the recipient directly on interface eth0.

2. Traffic for other addresses should be sent to the gateway “gw.mydom.org” through the interface eth0.

This table could be created by issuing the following commands:

```
[root@demo ~]# route add -net 192.168.0.0 netmask 255.255.0.0 dev eth0
[root@demo ~]# route add default gw gw.mydom.org
```

Note that we could just send out every packet as though the destination were on the local network and let ARP figure out how to get the packets to their destination. If we know *a priori,* though, that a given host isn't on our local network segment, we can save time and network traffic by sending it directly to a gateway router.

Static versus Dynamically-assigned IP Addresses:

How do you know what IP address to assign to your network interface? Your ISP can provide you with an IP address in two ways:

1. Static IP Addresses:

If your computer always needs to have the same IP address, your ISP may assign a “fixed” or “static” IP address to you. The ISP will keep a list of static IP address assignments, so that they can be sure they don't assign the same address more than once. The ISP will also probably assign the computer a name, and associate this name with the computer's static address in their DNS servers.

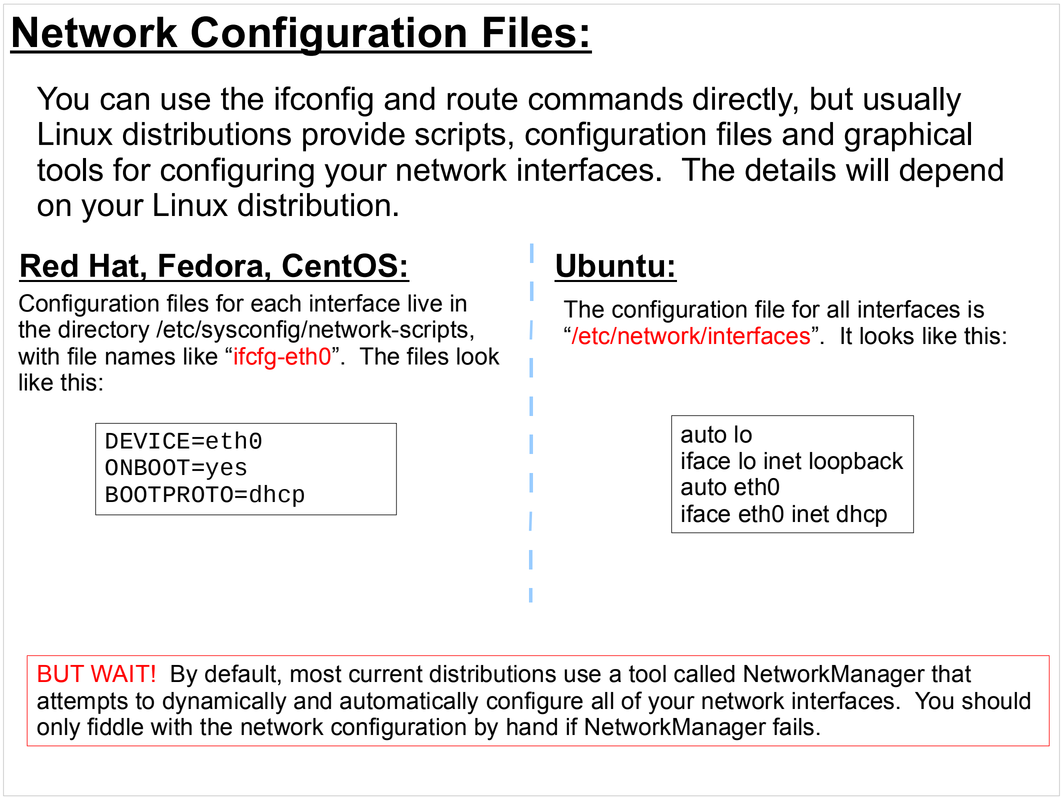
2. Dynamic IP Addresses:

If your computer doesn't need a fixed address, you can obtain a randomly-assigned address through a process called “Dynamic Host Configuration Protocol” (DHCP). Your ISP probably maintains a DHCP server that will, on demand, provide your computer with an IP address and other configuration information, such as a list of DNS servers.

Static addresses are usually appropriate for servers. Dynamic addresses are appropriate for most personal computers.

There are many tools for querying DHCP servers and obtaining an IP address. Some of the common ones are “dhclient” (used in Red Hat-derived distributions and Ubuntu), “pump” (used by KNOPPIX), and “dhcpcd” (not to be confused with “dhcpd” -- one's a client and the other's a server!). All of these will query the DHCP server, obtain an IP address and configure the interface for you.

The syntax for dhclient is just “dhclient eth0”.



Network Configuration Files:

You can use the ifconfig and route commands directly, but usually Linux distributions provide scripts, configuration files and graphical tools for configuring your network interfaces. The details will depend on your Linux distribution.

Red Hat, Fedora, CentOS:

Configuration files for each interface live in the directory /etc/sysconfig/network-scripts, with file names like "ifcfg-eth0". The files look like this:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Ubuntu:

The configuration file for all interfaces is "/etc/network/interfaces". It looks like this:

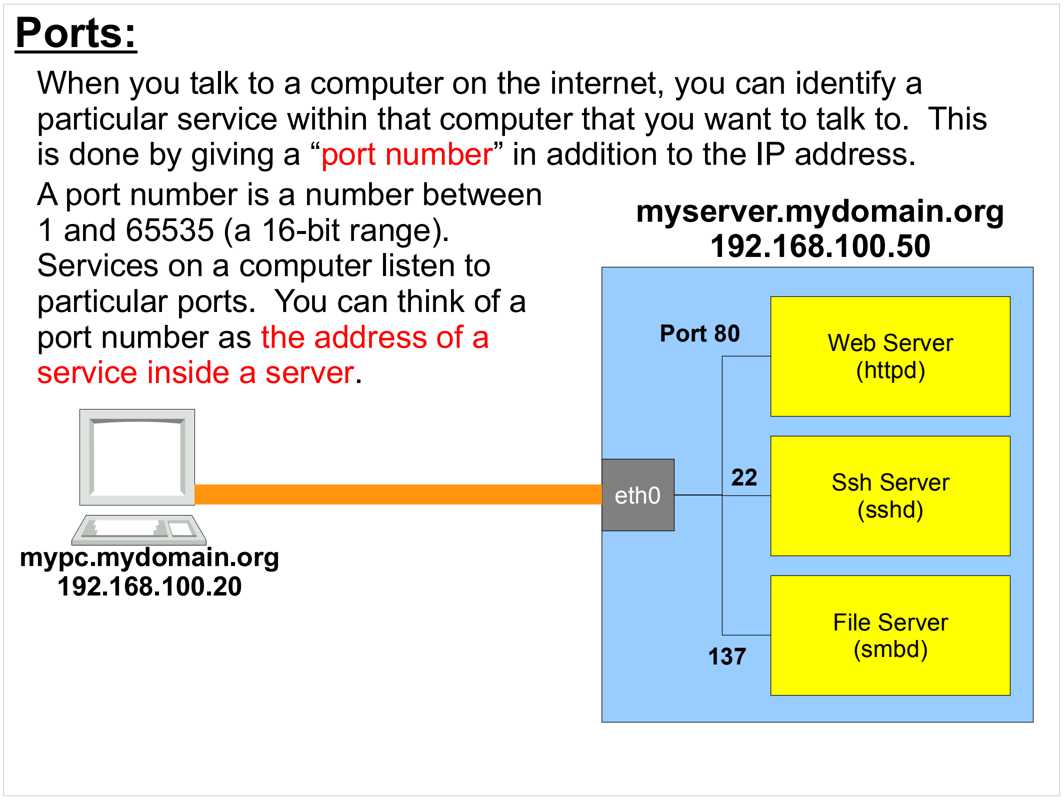
```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

BUT WAIT! By default, most current distributions use a tool called NetworkManager that attempts to dynamically and automatically configure all of your network interfaces. You should only fiddle with the network configuration by hand if NetworkManager fails.

NetworkManager can be a source of problems. My rule of thumb is: use NetworkManager for laptops, but turn it off for desktop computers.

Part 5: Ports

We know how to contact a remote computer, using its IP address, but what about contacting a particular service running on that remote computer? For that, we'll need to introduce an internal address called a “port number”. Sometimes people are confused by the name “port”. It sounds like some physical thing that you would plug a cable into. In the following, remember that ports are just addresses. They're not anything physical.



Ports:

When you talk to a computer on the internet, you can identify a particular service within that computer that you want to talk to. This is done by giving a “port number” in addition to the IP address.

A port number is a number between 1 and 65535 (a 16-bit range). Services on a computer listen to particular ports. You can think of a port number as the address of a service inside a server.

A port number is actually used at both ends of the connection. In the illustration above, the traffic leaving “mypc.mydomain.com” would originate from a particular source port on that computer. If the computer were browsing the web, for example, the traffic would originate from some randomly-assigned port on mypc, and would be address to destination port 80, on the remote computer, where the web server would be listening.

When an application requests a port on the local computer, it can ask for port “0”. In this special case, the computer picks a port at random, from a pool of ports available for that purpose, and gives it to the application.

Port Number Assignment:

The IANA maintains an official list of standard port numbers for various services. This list is purely advisory, but software authors seldom use ports in non-standard ways. The list of ports is divided into three sections:

- **Well-Known Ports (aka Privileged Ports):**
Ports 1-1023 are called "Well-Known" ports, and many of them have been in common use since the beginning of the Internet. These are the ports used by familiar services such as web, ftp, ssh, telnet and smb. Only processes owned by the root user are allowed to bind to these ports.

- **Registered Ports:**
Ports 1024-49151 are "Registered" ports. These are associated with applications that have registered with IANA and been assigned an official port number. Registering a port makes it less likely that someone else will use that port for another purpose.

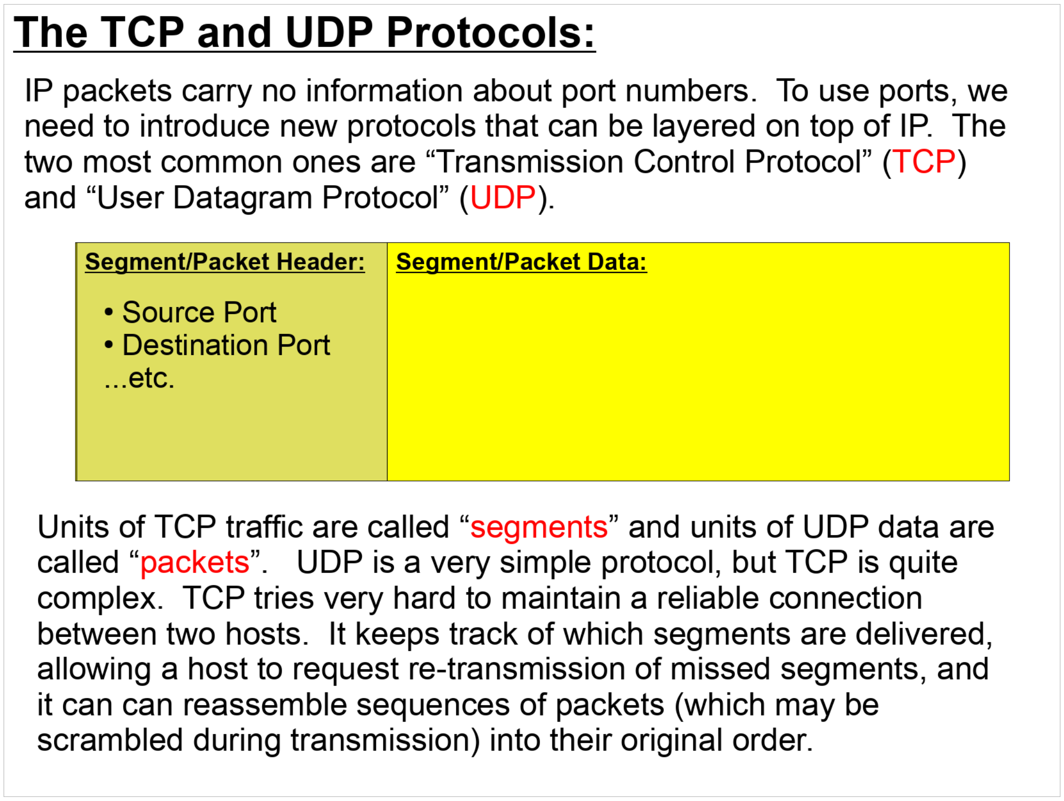
- **Dynamic/Private/Ephemeral ports:**
Ports 49152-65535 are available for temporary use, or for private use.

Some Common Ports:

Here are a few commonly-used ports:

22:	Ssh
80:	Http
443:	Https
25:	Smtp
20/21:	Ftp
53:	DNS
110:	Pop3
143:	Imap
389:	Ldap
993:	Imaps

You don't need to remember the numbers. You can usually refer to them by name. Most applications will look up ports by name, using the file "/etc/services". This file contains a list of port names, their associated numbers, and other information.



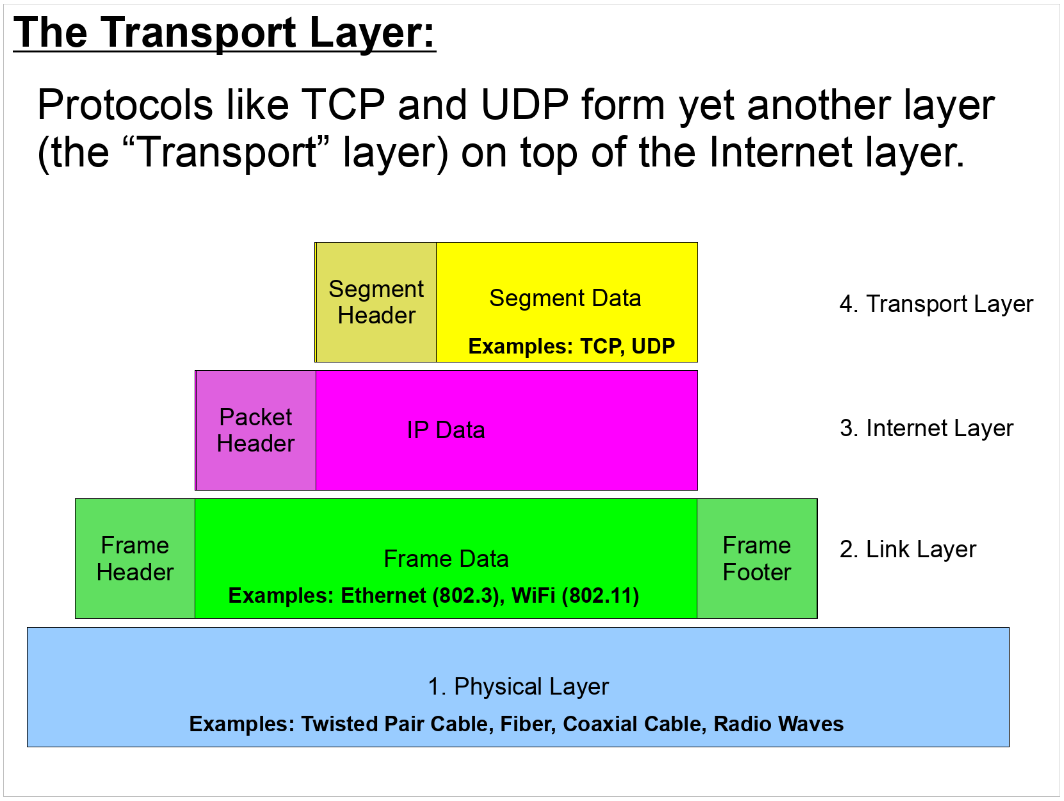
The TCP and UDP Protocols:

IP packets carry no information about port numbers. To use ports, we need to introduce new protocols that can be layered on top of IP. The two most common ones are “Transmission Control Protocol” (TCP) and “User Datagram Protocol” (UDP).

Segment/Packet Header:	**Segment/Packet Data:**
• Source Port • Destination Port ...etc.	

Units of TCP traffic are called “segments” and units of UDP data are called “packets”.   UDP is a very simple protocol, but TCP is quite complex. TCP tries very hard to maintain a reliable connection between two hosts. It keeps track of which segments are delivered, allowing a host to request re-transmission of missed segments, and it can can reassemble sequences of packets (which may be scrambled during transmission) into their original order.

Because it's so much more complex, TCP also has a lot more overhead. In situations where dropped packets can be tolerated, or where the order of packets doesn't matter, UDP is used instead.



The Transport Layer:

Protocols like TCP and UDP form yet another layer (the “Transport” layer) on top of the Internet layer.

Segment Header

Segment Data

Examples: TCP, UDP

4. Transport Layer

Packet Header

IP Data

3. Internet Layer

Frame Header

Frame Data

Examples: Ethernet (802.3), WiFi (802.11)

Frame Footer

2. Link Layer

1. Physical Layer

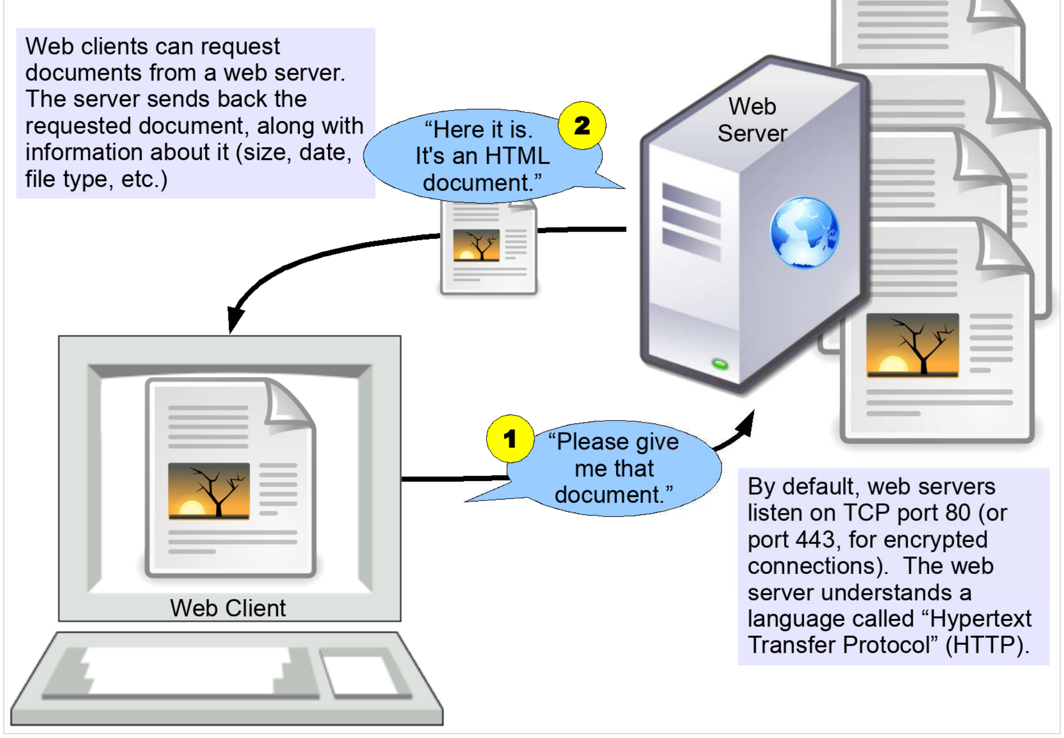
Examples: Twisted Pair Cable, Fiber, Coaxial Cable, Radio Waves

So now we have four layers in our layer-cake. On top of the physical layer, Ethernet frames carry data from computer to computer, guided by MAC addresses. Within those frames are IP packets, guided by IP addresses. Inside the IP packets are TCP or UDP segments/packets, addressed by port number.

Part 6: Applications

This is Tim Berners-Lee's original web server from CERN. Many applications use the Internet, but none is more well-known than the World-Wide Web. Let's take a look at how web servers work, as an example of how an application transmits information across the network.

What's a Web Server?:



Web clients can request documents from a web server. The server sends back the requested document, along with information about it (size, date, file type, etc.)

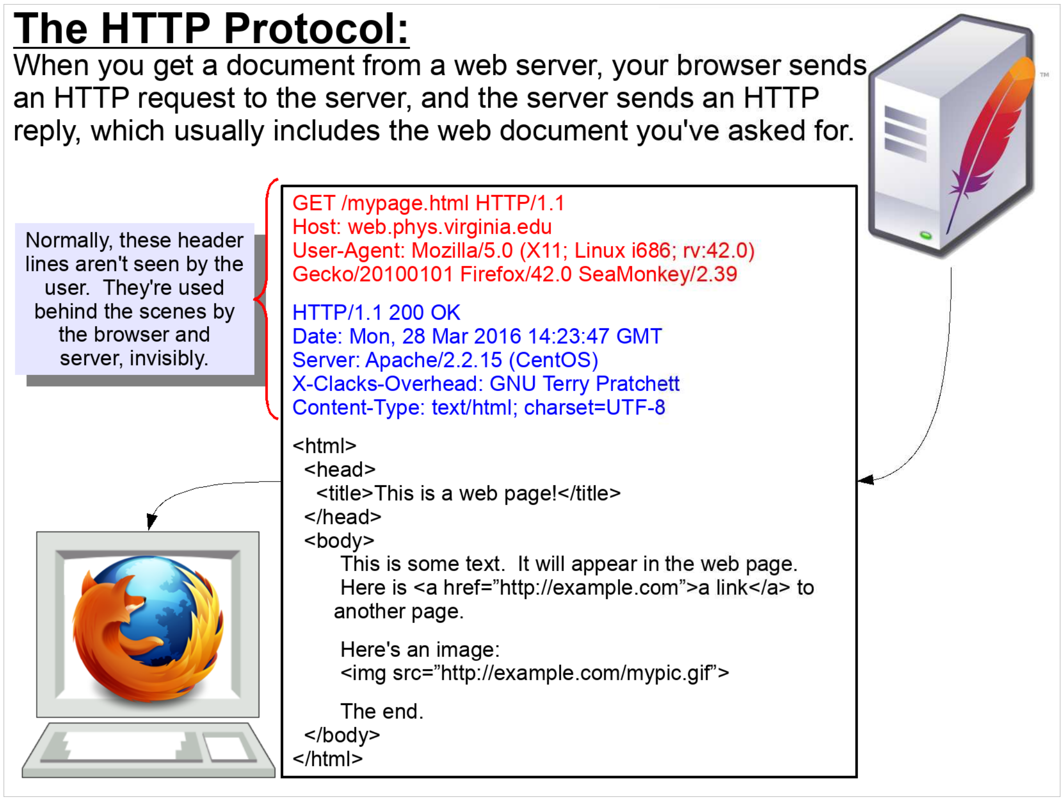
“Here it is. It's an HTML document.”
2
Web Server
1
“Please give me that document.”
Web Client

By default, web servers listen on TCP port 80 (or port 443, for encrypted connections). The web server understands a language called “Hypertext Transfer Protocol” (HTTP).

A web server is a computer that's capable of serving up documents through the “Hypertext Transfer Protocol” (HTTP). A network service running on these computers listens to TCP port 80 and/or port 443, and responds to HTTP requests.

When the web server returns a document, it also returns some metadata about the document: its size, creation time, and file type. The file type is expressed as one of the “Multimedia Internet Mail Extention” (MIME) types. Some common types are “text/html”, “application/pdf”, and “text/plain”. The file type helps the client decide how to deal with the file.

Interactive web clients are usually referred to as “web browsers”.



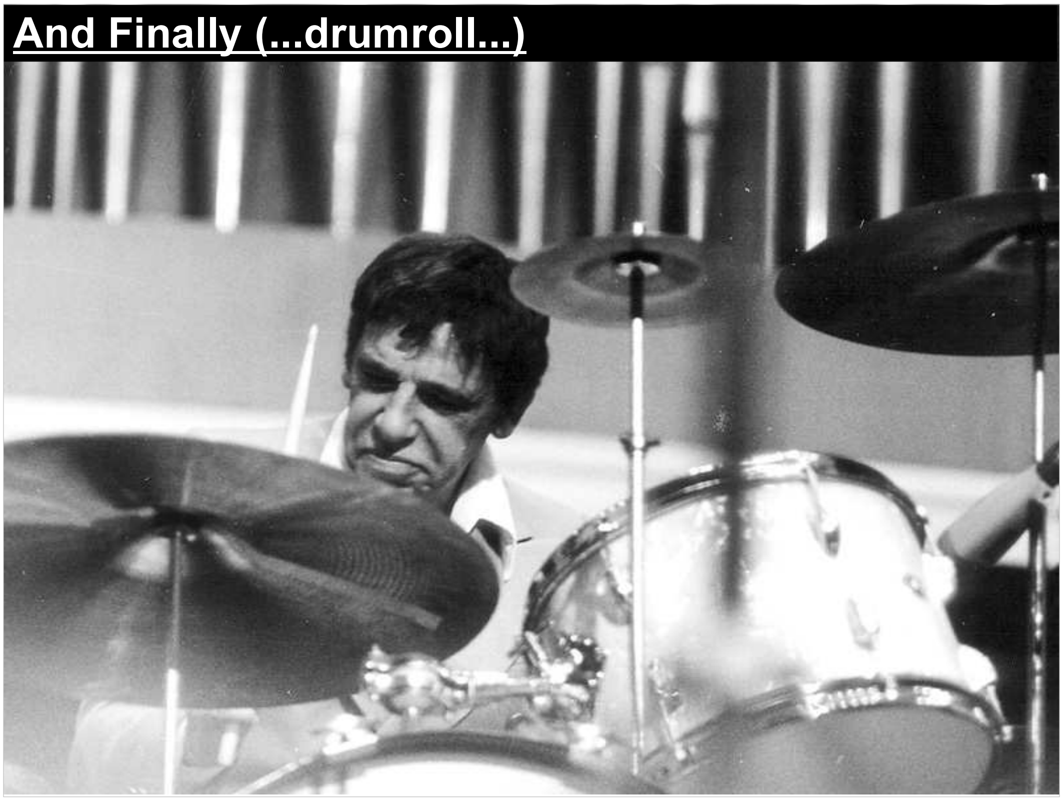
The HTTP Protocol:

When you get a document from a web server, your browser sends an HTTP request to the server, and the server sends an HTTP reply, which usually includes the web document you've asked for.

Normally, these header lines aren't seen by the user. They're used behind the scenes by the browser and server, invisibly.

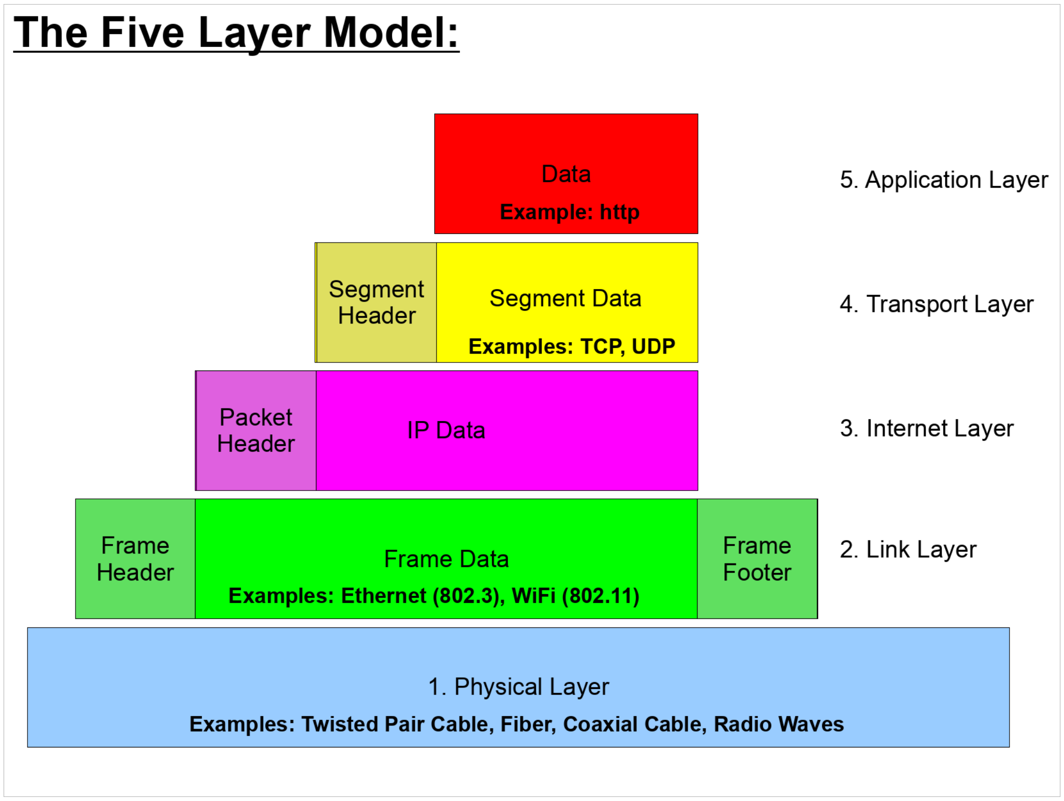
```
GET /mypage.html HTTP/1.1
Host: web.phys.virginia.edu
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:42.0)
Gecko/20100101 Firefox/42.0 SeaMonkey/2.39

HTTP/1.1 200 OK
Date: Mon, 28 Mar 2016 14:23:47 GMT
Server: Apache/2.2.15 (CentOS)
X-Clacks-Overhead: GNU Terry Pratchett
Content-Type: text/html; charset=UTF-8

<html>
  <head>
    <title>This is a web page!</title>
  </head>
  <body>
      This is some text.  It will appear in the web page.
      Here is <a href=”http://example.com”>a link</a> to
     another page.

      Here's an image:
      <img src=”http://example.com/mypic.gif”>

      The end.
  </body>
</html>
```

As we saw last time, you can see these headers if you telnet to port 80 on the web server and manually use the HTTP “GET” command. You can also see them if you use the “-S” flag when you fetch a document with wget.



And Finally (...drumroll...)

Now let's put it all together.



The Five Layer Model:

Data
Example: http

5. Application Layer

Segment Header

Segment Data
Examples: TCP, UDP

4. Transport Layer

Packet Header

IP Data

3. Internet Layer

Frame Header

Frame Data
Examples: Ethernet (802.3), WiFi (802.11)

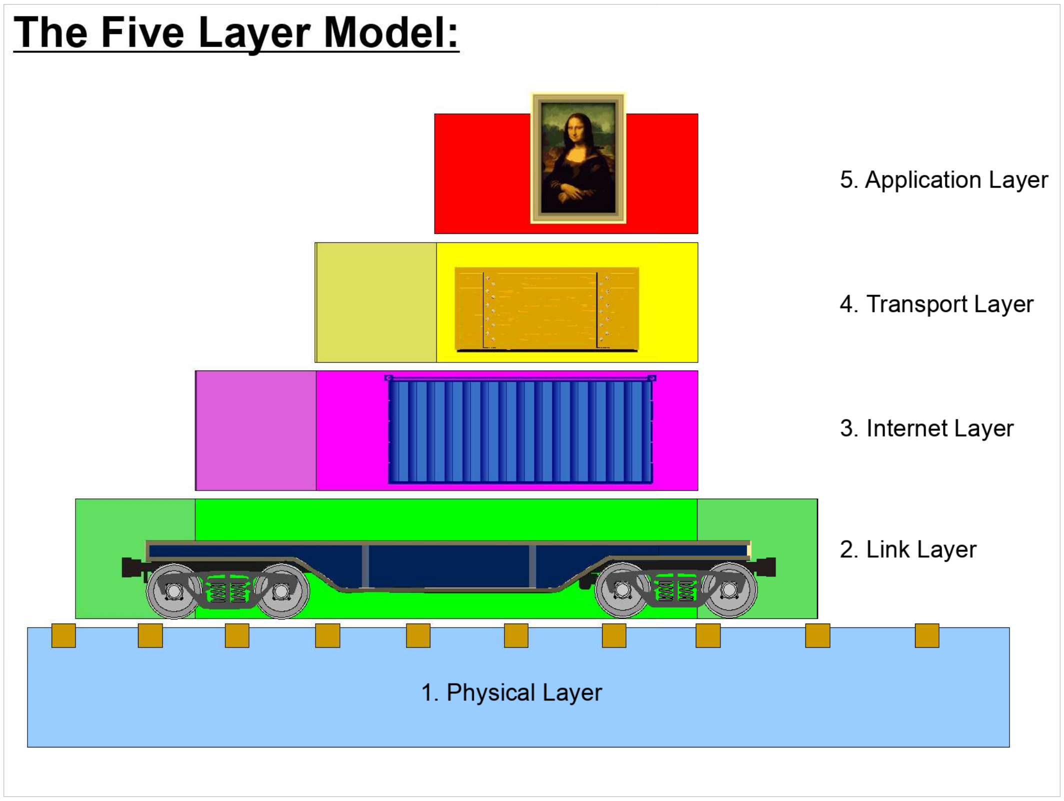
Frame Footer

2. Link Layer

1. Physical Layer
Examples: Twisted Pair Cable, Fiber, Coaxial Cable, Radio Waves

Next time, we'll talk about one more layer: the application layer, where protocols like http live. These layers make up what's called the TCP/IP Five Layer Model of networking.

The Five Layer Model is a simplified version of a more general model of networking called the OSI Seven Layer Model, which you may have heard of. The Five layer model gives a more intuitive picture of how the most common type of networking works.



The Five Layer Model:

5. Application Layer

4. Transport Layer

3. Internet Layer

2. Link Layer

1. Physical Layer

Here's another way of looking at our layers.

Part 7: Monitoring Traffic

Now let's look at some tools for monitoring network activity.

The “netstat” Command:

The “netstat” command shows information about network connections to your computer. It shows the source and destination address and port for each connection, and it can be made to show the process ID and process name of the processes that are bound to these ports.

```
[root@demo ~]# netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address     Foreign Address    State       PID/Program
name
tcp        0      0 0.0.0.0:750       0.0.0.0:*          LISTEN      354/rpc.statd
tcp        0      0 0.0.0.0:111       0.0.0.0:*          LISTEN      3507/portmap
tcp        0      0 0.0.0.0:6000      0.0.0.0:*          LISTEN      13034/X
tcp        0      0 127.0.0.1:631     0.0.0.0:*          LISTEN      5858/cupsd
tcp        0      0 0.0.0.0:3551      0.0.0.0:*          LISTEN      8305/apcupsd
tcp        0      0 10.2.1.43:37218   10.2.2.108:22      ESTABLISHED 7491/ssh
tcp        0      0 10.2.1.43:25      10.9.3.3:50071     TIME_WAIT   -
tcp        0      0 10.2.1.43:38860   10.2.1.159:22      ESTABLISHED 5581/ssh
tcp        0      0 10.2.1.43:54874   10.2.2.107:22      ESTABLISHED 25409/ssh
tcp        0      0 10.2.1.43:57525   10.2.1.57:2200     ESTABLISHED 27818/ssh
tcp        0      0 127.0.0.1:39788   127.0.0.1:783      TIME_WAIT   -
tcp        0      0 10.2.1.43:47548   128.143.100.51:22  ESTABLISHED 11350/ssh
tcp        0      0 10.2.1.43:42177   10.2.1.44:22       ESTABLISHED 15294/ssh
tcp        0      0 10.2.1.43:25      10.2.1.105:53651   TIME_WAIT   -
tcp        0      0 127.0.0.1:49912   127.0.0.1:22       ESTABLISHED 28866/ssh
tcp        0      0 10.2.1.43:37956   10.2.1.114:22      ESTABLISHED 7362/ssh
tcp        0      0 10.2.1.43:47173   10.2.1.113:22      ESTABLISHED 7405/ssh
tcp        0      0 127.0.0.1:60554   127.0.0.1:22       ESTABLISHED 26185/ssh
...etc.
```

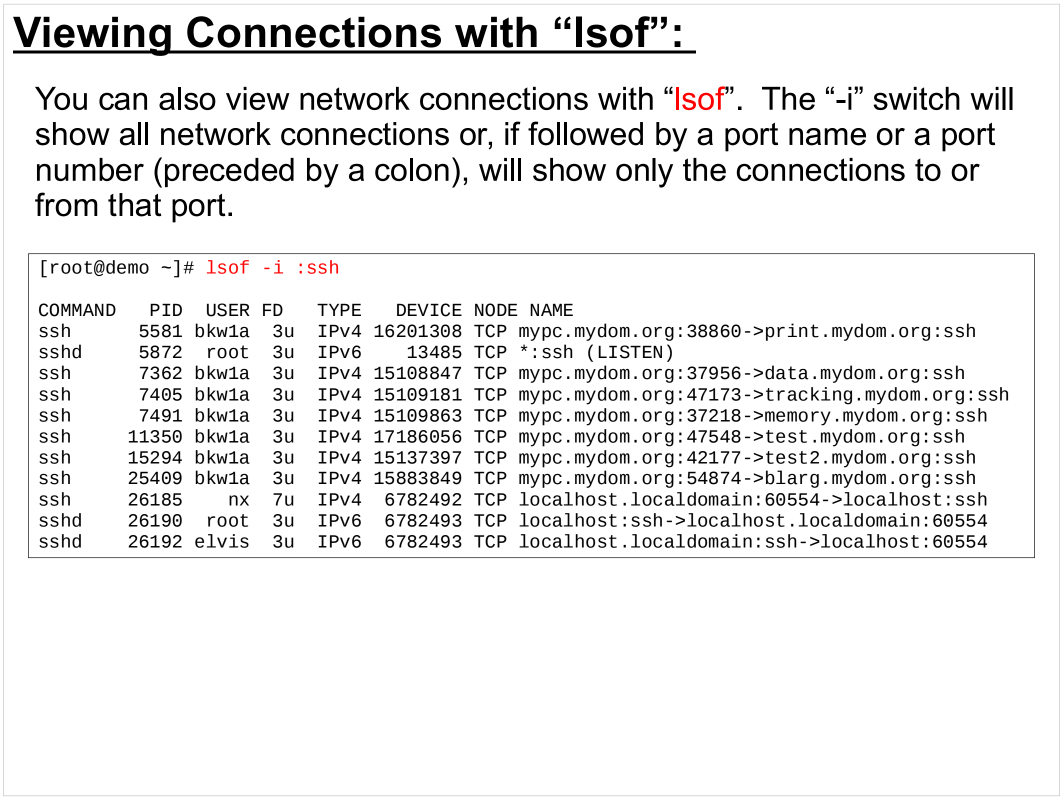
Netstat will also show you information about “Unix domain sockets”. Ignore this for now. These are local connections between processes running on your computer.

The switches shown above are:

-a Show all connections, including servers that are just listening.

-n Don't resolve host or port names. Just show the numbers.

-p Show the process IDs and process names.



Viewing Connections with “lsof”:

You can also view network connections with “lsof”. The “-i” switch will show all network connections or, if followed by a port name or a port number (preceded by a colon), will show only the connections to or from that port.

```
[root@demo ~]# lsof -i :ssh

COMMAND   PID  USER FD   TYPE   DEVICE NODE NAME
ssh      5581 bkw1a  3u  IPv4 16201308 TCP mypc.mydom.org:38860->print.mydom.org:ssh
sshd     5872  root  3u  IPv6    13485 TCP *:ssh (LISTEN)
ssh      7362 bkw1a  3u  IPv4 15108847 TCP mypc.mydom.org:37956->data.mydom.org:ssh
ssh      7405 bkw1a  3u  IPv4 15109181 TCP mypc.mydom.org:47173->tracking.mydom.org:ssh
ssh      7491 bkw1a  3u  IPv4 15109863 TCP mypc.mydom.org:37218->memory.mydom.org:ssh
ssh     11350 bkw1a  3u  IPv4 17186056 TCP mypc.mydom.org:47548->test.mydom.org:ssh
ssh     15294 bkw1a  3u  IPv4 15137397 TCP mypc.mydom.org:42177->test2.mydom.org:ssh
ssh     25409 bkw1a  3u  IPv4 15883849 TCP mypc.mydom.org:54874->blarg.mydom.org:ssh
ssh     26185    nx  7u  IPv4  6782492 TCP localhost.localdomain:60554->localhost:ssh
sshd    26190  root  3u  IPv6  6782493 TCP localhost:ssh->localhost.localdomain:60554
sshd    26192 elvis  3u  IPv6  6782493 TCP localhost.localdomain:ssh->localhost:60554
```

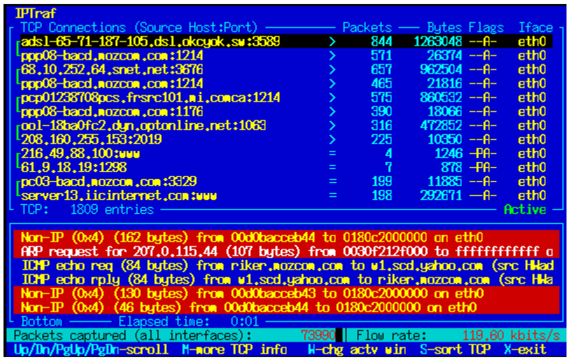
We saw lsof earlier as a tool for seeing which files were currently being used by various processes. It can also show us network connections being used by processes.

Using “iptraf” to Monitor Traffic in Real Time:

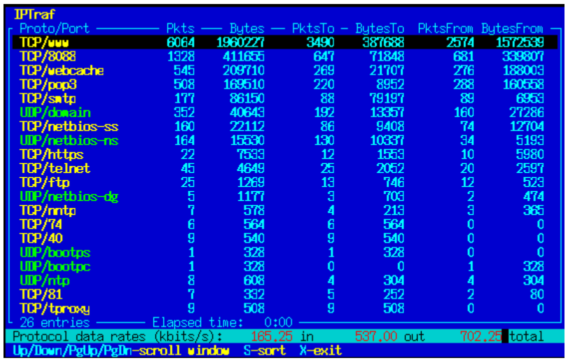
You can use the “iptraf” command to monitor network traffic in real time. Iptraf is menu-driven, and has several modes.

In the mode shown at top, it will show information about each new connection in the top pane, and a running stream of information about incoming packets in the bottom pane.

In the mode shown below, iptraf gathers statistics about traffic on each port.



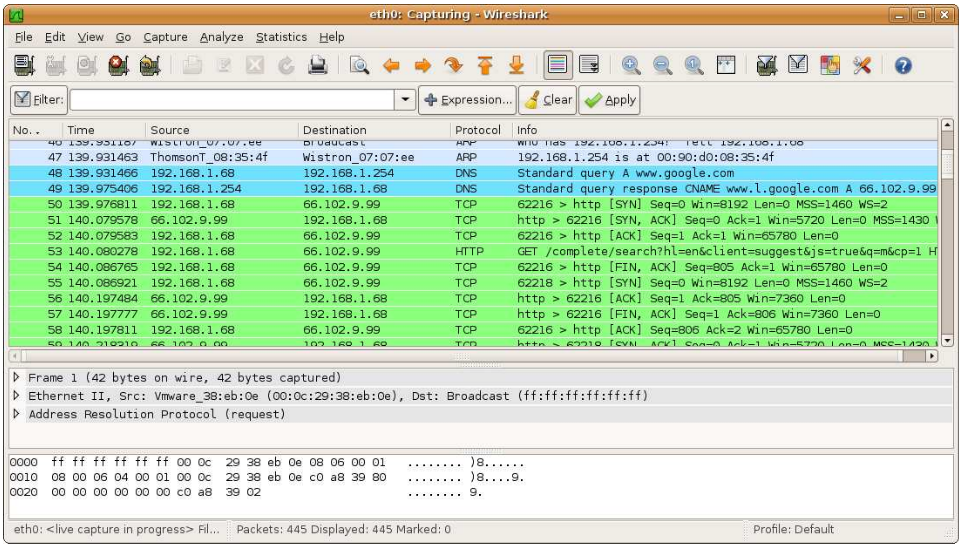
```
IPTraf
 TCP Connections (Source Host:Port) ----------- Packets --- Bytes Flags  Iface
 adsl-65-71-187-105.dsl.okcyok.sw:3589      >      844  1263048 --A-   eth0
 ppp08-bacd.mozcom.com:1214                 >      571    26374 --A-   eth0
 68.10.252.64.snet.net:3676                 >      657   962504 --A-   eth0
 ppp08-bacd.mozcom.com:1214                 >      465    21816 --A-   eth0
 pcp01238708pcs.frsrc101.mi.comca:1214      >      575   860532 --A-   eth0
 ppp08-bacd.mozcom.com:1176                 >      390    18066 --A-   eth0
 ool-18ba0fc2.dyn.optonline.net:1063        >      316   472852 --A-   eth0
 208.160.255.153:2019                       >      225    10350 --A-   eth0
 216.49.88.100:www                          =        4     1246 -PA-   eth0
 61.9.18.19:1298                            =        7      878 -PA-   eth0
 pc03-bacd.mozcom.com:3329                  =      199    11885 --A-   eth0
 server13.iicinternet.com:www               =      198   292671 --A-   eth0
 TCP:   1809 entries -------------------------------------------------- Active

 Non-IP (0x4) (162 bytes) from 00d0bacceb44 to 0180c2000000 on eth0
 ARP request for 207.0.115.44 (107 bytes) from 0030f212f000 to ffffffffffff o
 ICMP echo req (84 bytes) from riker.mozcom.com to w1.scd.yahoo.com (src HWad
 ICMP echo rply (84 bytes) from w1.scd.yahoo.com to riker.mozcom.com (src HWa
 Non-IP (0x4) (130 bytes) from 00d0bacceb43 to 0180c2000000 on eth0
 Non-IP (0x4) (46 bytes) from 00d0bacceb44 to 0180c2000000 on eth0
 Bottom --------- Elapsed time:   0:01 -----------------------------------------
 Packets captured (all interfaces):        73990 | Flow rate:     119.60 kbits/s
 Up/Dn/PgUp/PgDn-scroll  M-more TCP info   W-chg actv win  S-sort TCP  X-exit
```



```
IPTraf
 Proto/Port ----------- Pkts --- Bytes -- PktsTo - BytesTo  PktsFrom BytesFrom -
 TCP/www                6064   1960227     3490    387688      2574   1572539
 TCP/8088               1328    411655      647     71848       681    339807
 TCP/webcache            545    209710      269     21707       276    188003
 TCP/pop3                508    169510      220      8952       288    160558
 TCP/smtp                177     86150       88     79197        89      6953
 UDP/domain              352     40643      192     13357       160     27286
 TCP/netbios-ss          160     22112       86      9408        74     12704
 UDP/netbios-ns          164     15530      130     10337        34      5193
 TCP/https                22      7533       12      1553        10      5980
 TCP/telnet               45      4649       25      2052        20      2597
 TCP/ftp                  25      1269       13       746        12       523
 UDP/netbios-dg            5      1177        3       703         2       474
 TCP/nntp                  7       578        4       213         3       365
 TCP/74                    6       564        6       564         0         0
 TCP/40                    9       540        9       540         0         0
 UDP/bootps                1       328        1       328         0         0
 UDP/bootpc                1       328        0         0         1       328
 UDP/ntp                   8       608        4       304         4       304
 TCP/81                    7       332        5       252         2        80
 TCP/tproxy                9       508        9       508         0         0
 26 entries -------- Elapsed time:   0:00 ------------------------------------
 Protocol data rates (kbits/s):     165.25 in      537.00 out      702.25 total
 Up/Down/PgUp/PgDn-scroll window  S-sort  X-exit
```

Using “wireshark” to Monitor Traffic in Real Time:

Wireshark is an invaluable tool for analyzing network traffic. It allows you to capture (optionally filtered) traffic, dissect it, do offline filtering, and produce graphs and statistics.



No.	Time	Source	Destination	Protocol	Info
46	139.931187	Wistron_07:07:ee	Broadcast	ARP	Who has 192.168.1.254? Tell 192.168.1.68
47	139.931463	ThomsonT_08:35:4f	Wistron_07:07:ee	ARP	192.168.1.254 is at 00:90:d0:08:35:4f
48	139.931466	192.168.1.68	192.168.1.254	DNS	Standard query A www.google.com
49	139.975406	192.168.1.254	192.168.1.68	DNS	Standard query response CNAME www.l.google.com A 66.102.9.99
50	139.976811	192.168.1.68	66.102.9.99	TCP	62216 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
51	140.079578	66.102.9.99	192.168.1.68	TCP	http > 62216 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430
52	140.079583	192.168.1.68	66.102.9.99	TCP	62216 > http [ACK] Seq=1 Ack=1 Win=65780 Len=0
53	140.080278	192.168.1.68	66.102.9.99	HTTP	GET /complete/search?hl=en&client=suggest&js=true&q=m&cp=1 H
54	140.086765	192.168.1.68	66.102.9.99	TCP	62216 > http [FIN, ACK] Seq=805 Ack=1 Win=65780 Len=0
55	140.086921	192.168.1.68	66.102.9.99	TCP	62218 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
56	140.197484	66.102.9.99	192.168.1.68	TCP	http > 62216 [ACK] Seq=1 Ack=805 Win=7360 Len=0
57	140.197777	66.102.9.99	192.168.1.68	TCP	http > 62216 [FIN, ACK] Seq=1 Ack=806 Win=7360 Len=0
58	140.197811	192.168.1.68	66.102.9.99	TCP	62216 > http [ACK] Seq=806 Ack=2 Win=65780 Len=0
59	140.218319	66.102.9.99	192.168.1.68	TCP	http > 62218 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430

```
▷ Frame 1 (42 bytes on wire, 42 bytes captured)
▷ Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 00 0c  29 38 eb 0e 08 06 00 01   ........ )8......
0010  08 00 06 04 00 01 00 0c  29 38 eb 0e c0 a8 39 80   ........ )8....9.
0020  00 00 00 00 00 00 c0 a8  39 02                     ........ 9.
```

eth0: <live capture in progress> Fil... Packets: 445 Displayed: 445 Marked: 0 Profile: Default

The End

Thanks!