Computer Networking

NETWORKING

- Computer Networking
- Computer network is like a phone system for computers, data call
- Basic outline is surprisingly simple, complex details
- Worth knowing .. using all the time

The details of networking -- like anything really -- can be quite complicated. But the basic ideas of how it all works are surprisingly simple, and that's what we're going to study.

The Internet is like a global phone system for computers: a computer can "call" another computer on the internet to get or send a little information. Suppose your laptop is connected to the internet, and you type "http://www.nytimes.com" into your browser -- what happens? Your computer contacts the computer "www.nytimes.com" -- placing a "call" in effect -- and sends a request for the main web page. The request is small, about 1KB (1 kilobyte). The www.nytimes.com machine sends back a large response which is the web page -- maybe 200KB -- and ends the call. Your browser gets back all this data and formats it for your screen so you can read the text, click links etc. We'll look at this fetch-web-page example a few different ways to see how the internet works.

LAN - LOCAL AREA NETWORK

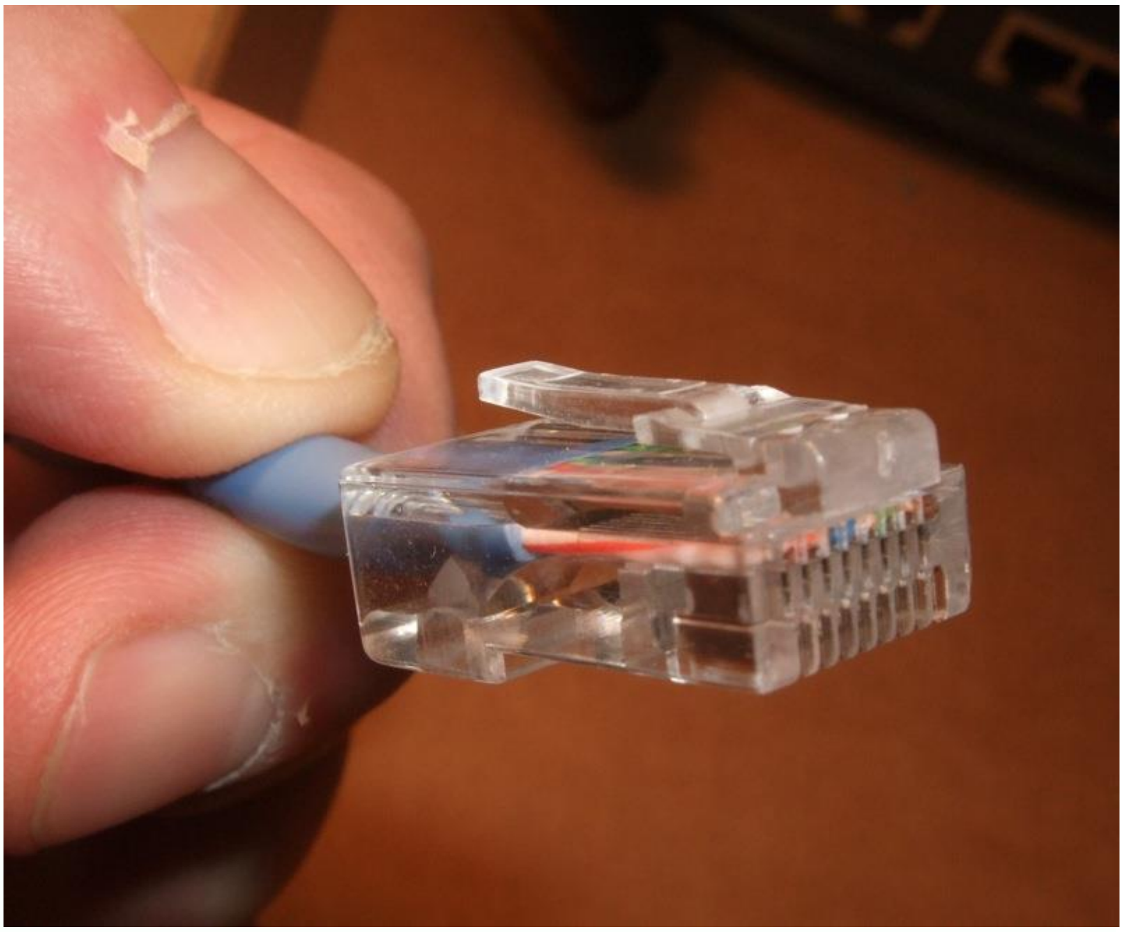
- Start with small scale

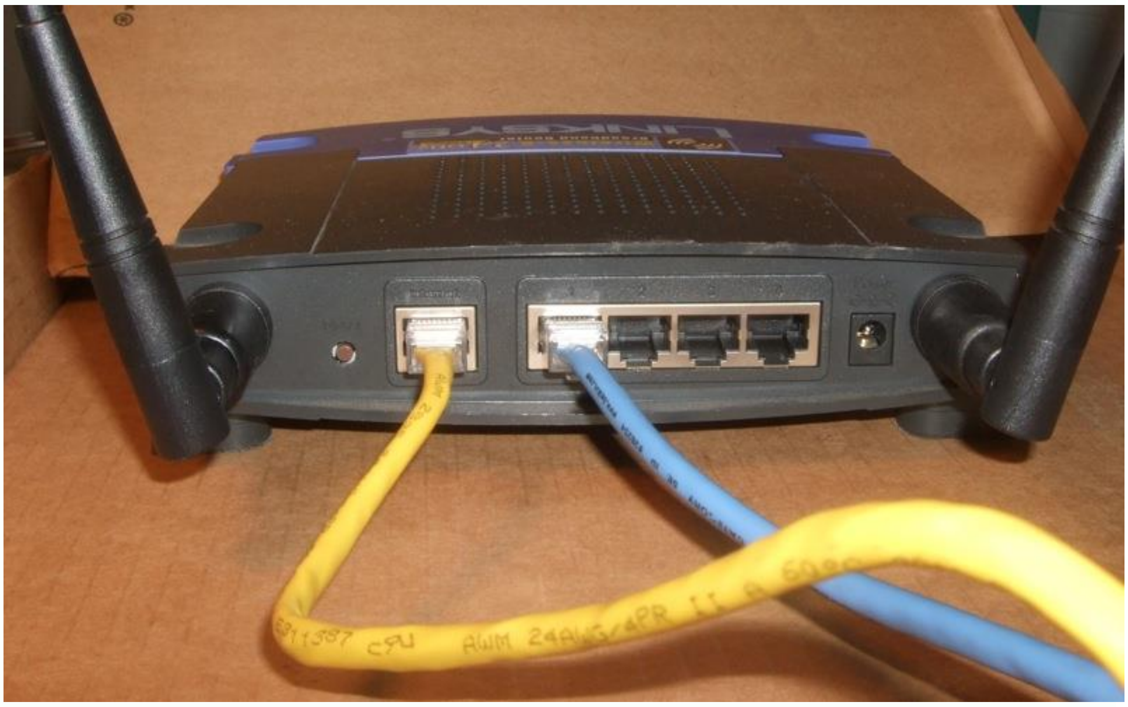
- LAN - Local Area Network
- One house, one floor of a building
- Later, show scale up to world-wide internet
- e.g. Ethernet, wired LAN
- e.g. Wi-Fi, wireless LAN

We'll start by looking at LAN (local area network) technology -- connecting 2-50 computers in a house or on one floor of a building.

ETHERNET LAN

- Ethernet
- Very ubiquitous wired LAN technology
- Wires about as thick as a drinking straw
- 100 meter max wire length -- **local**
- Wires often yellow or blue
- RJ-45 plug, like a big phone plug

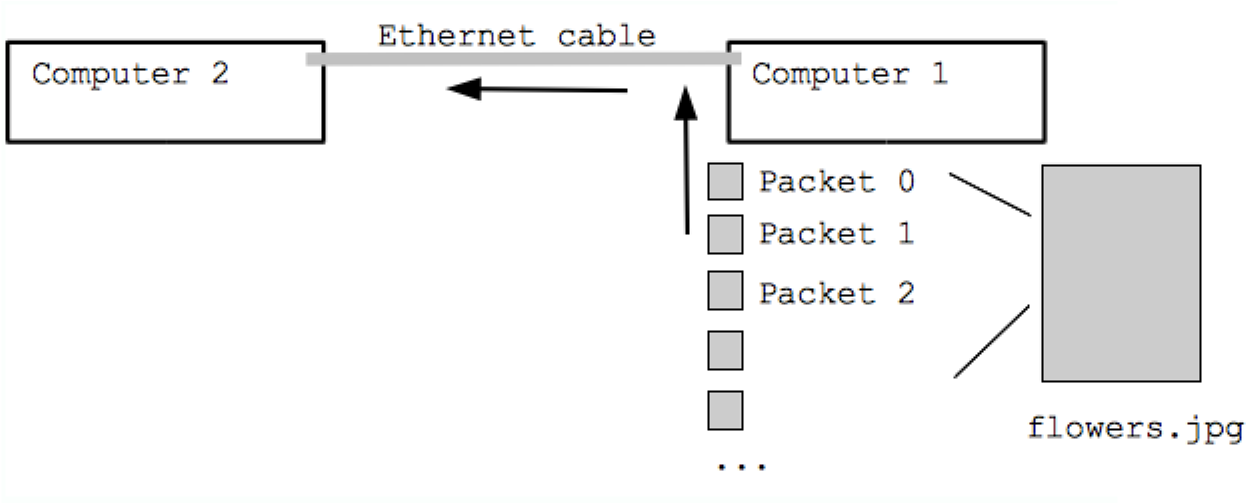

Ethernet RJ45 plug


Ethernet cables plugged into the back of a Wi-Fi router

Ethernet is an extremely common and influential wired LAN standard, so we'll start there. Ethernet cable lengths are typically limited to 100 meters, in keeping with its "local" orientation. A typical LAN application is networking the computers in one room or in one floor of a building. The most common form of ethernet wiring is 100base-T (100 megabit) with "RJ-45" connectors on the ends. An RJ-45 connector is about the size of your pinkie finger, like a wide phone wire plug.

PACKETS - DATA FROM HERE TO THERE

- Send image from one computer to another on ethernet
- This is the "one hop" case (scale up to whole world later)
- e.g. 50KB image.jpg
- 50,000 bytes
- How to send the image.jpg on the wire?
- **Packets**
- Divide bytes of image.jpg into packets
- Say each packet is 1500 bytes (varies)
- Then image.jpg divides into about 32 packets
- Ethernet: transmit one packet between computers



We'll start with the simplest case of two computer connected with an ethernet cable, and we want to send a 50KB jpeg image file from one computer to the other. This is the "one-hop" case .. networking between two computers separated only by an ethernet cable. Below we'll scale this up to the full Internet case of two computers on separate sides of the world. The first question is: how are the bytes of the image file on one computer sent to the other computer over the rather skinny ethernet cable?

For transmission, the 50KB of the image is divided into **packets**. The packet is the natural unit of transmission within networking. In this case, say each packet is about 1500 bytes (a typical ethernet packet size), then the bytes of the 50KB image could be divided into about 32 packets. It is not required that all the packets be the same size, just that every byte of the image is sent in one packet or another.

ETHERNET -- SENDING ONE PACKET

- Look at transmission of one 1500 byte packet
- Each byte is 8 bits, e.g. 0 1 1 0 1 0 1 0
- Send each byte (slight simplification):

  Go through 0/1 bits, left to right

--For each 1, put 3 volts on the wire

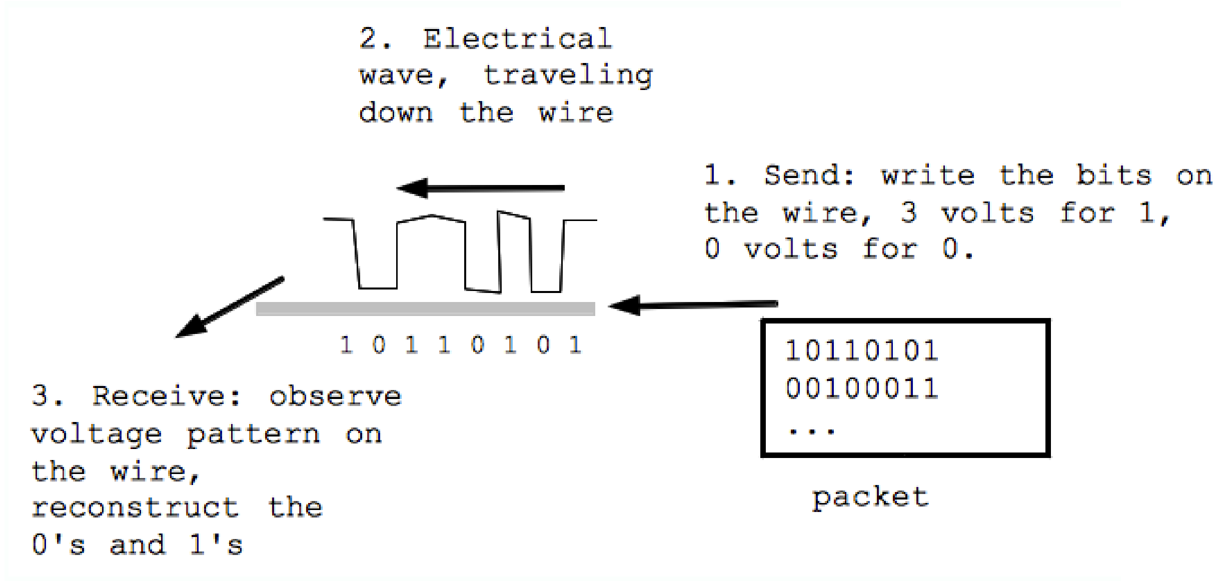
--For each 0, put 0 volts on the wire

- Receive

  --Follow along the pattern of 3 volts / 0 volts coming down the wire

  --Assemble the 0's and 1's to make each byte

- **digital transmission** - just 0's and 1's

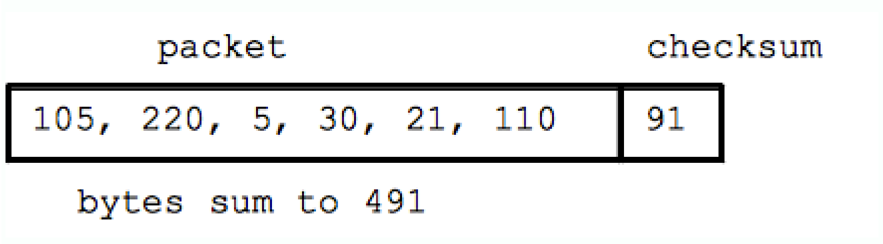

2. Electrical wave, traveling down the wire
1. Send: write the bits on the wire, 3 volts for 1, 0 volts for 0.
1 0 1 1 0 1 0 1
10110101
00100011
...
packet
3. Receive: observe voltage pattern on the wire, reconstruct the 0's and 1's

Ethernet provides a basic facility to transmit a packet between two computers connected by the ethernet cable. Say we have a packet of 1500 bytes of information we want to send. Each byte is 8 bits, so that's 12000 bits to send, where each bit is a 0 or 1. Here's an oversimplification that captures how it works: the ethernet cable contains two wires connecting the computers. The sending computer could read through the 12000 bits in order, and for each 1 bit, put 3 volts between the wires, and for each 0 bit, put 0 volts between the wires. The receiving computer can follow along, noting the 3v/0v pattern on the wires over time and so receive the 12000 bits. In reality the most recent ethernet contains 4 pairs of wires and supports sending information in both directions and with a more complex voltage scheme. However, this pattern of going through the bits and varying the voltage to "send" each bit is essentially how it all works.

PACKET ERRORS -- CHECKSUM RE-SEND

- 1 Gigabit - 1 billion bits per second
- Detect the occasional error
- **Checksum** - receiver can detect that the packet was received correctly
- Example checksum scheme:
  --Sender adds up all packet's bytes, e.g. sum 157231
  --For illustration, say the last 2 digits, 31, is the "checksum" of the packet
  --(Actual checksum scheme is more sophisticated)
  --Sender sends the checksum at the end of the packet
  --Receiver: add up the bytes received, check that the checksum matches
  Checksum not matching indicates data corruption
  Receiver asks the sender to re-send that packet
- Very likely to detect errors
- Not perfect - 2 errors could cancel out

- Checksums are widely used in computer systems
- How your file transfers get every bit correct



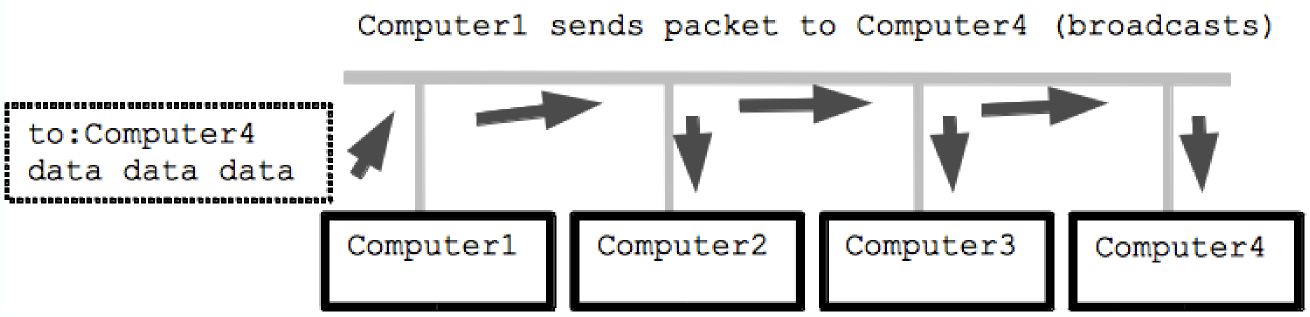
Each packet includes extra **checksum** bytes, so the receiver of the packet can detect if some of the bits in the packet got corrupted in transmission. A simple example checksum scheme would be: go through all the bytes, and add them all up. The checksum is the last 2 digits of the sum of all the bytes; send that checksum as an extra byte along with the rest of the packet data. The receiver can do the same computation -- adding up all the bytes -- to check that they get the same checksum. The actual checksum algorithm is more complex than just adding up the bytes, and is more capable of detecting errors. The checksum is probabilistic, not detecting 100% of errors; there is a microscopic chance that an error occurs but the checksum does not catch it.

The checksum allows the receiver to notice that a packet did not come through right, and get the sender to re-send that packet. Most packets get through fine, but re-sending a few packets happens all the time in your life. In this way, when you send a JPEG file from one place to another, it comes through correctly, down to every last bit.

MULTIPLE COMPUTERS -- ETHERNET DESIGN

- Ethernet -- elegant design
- Layout: one wire, shared by all the computers
- **No central control** -- distributed, collaborative

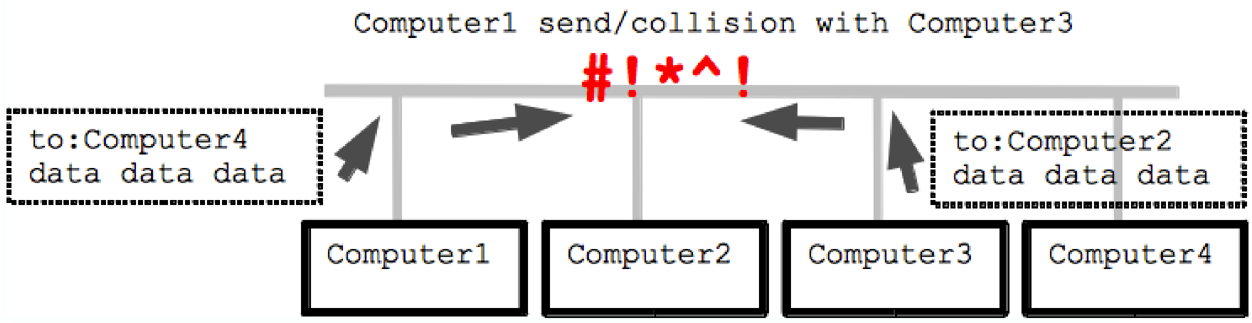
- Like talking at a party
- Suppose one computer wants to send to another on the wire
- Send:
  --Every computer has a unique address on the wire
  --Packet includes to:addr of recipient
  --Sender waits for period of silence on the wire, sends packet
  --Packet spreads out on wire, reaching all computers
  --More "broadcast" than "send"
- Receive:
  --All computers listen to the wire all the time
  --Pick out packets addressed to them, ignore other packets



ETHERNET COLLISION

- "Collision" happens in rare cases

- Two computers transmit at the same time
- --Data collides on the wire, hopelessly garbled
- --Senders notice the collision, stop sending
- --Both senders wait a **random** amount of time, then try again
- --What if both senders tried to send again immediately?

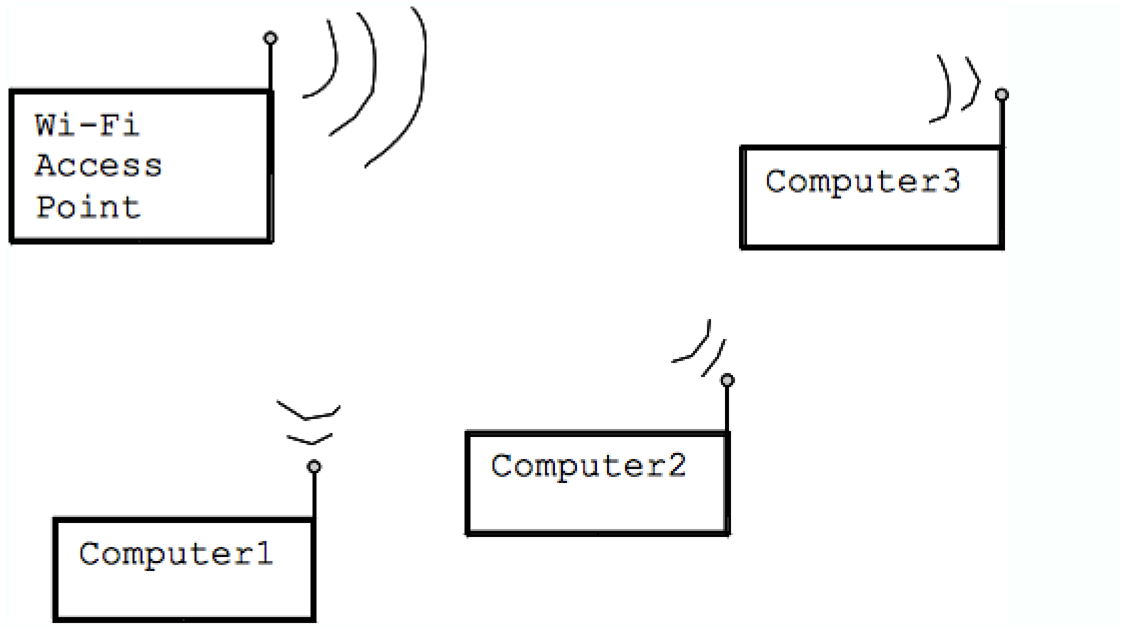


Ethernet, even in its original simple form, is an interesting and elegant design for a LAN network, connecting a bunch of local computers together with one shared wire. It demonstrates the paradigm of getting many actors to collaborate without any central control. Ethernet was created by Bob Metcalfe at Xerox PARC in 1975. Here is how the original Ethernet worked, although newer versions are more complex and have higher performance.

- There is one wire, and all the computers are connected to it -- they share the wire -- this is what makes it "broadband" as they all share the one medium. This is a simple and cheap configuration; the computers just connect to a dumb, shared wire. To add a computer, just connect it to the wire.
- Each computers on the LAN has a unique address. This is know as its **MAC** address (Media Access Control). The MAC address is burned in at the factory, thankfully not something you need to set or maintain manually.

- Only one computer should transmit at a time
- All computers listen to the wire all the time, picking up packets addressed to them and ignoring packets not for them
- To send data, the sender divides their message into small "packets" of, say, around 1500 bytes. Every packet begins with the address of the recipient.
- To send, the sender listens, waiting for a period of silence on the wire. When there's a period of silence, the sender sends their packet on the wire, effectively broadcasting it over the whole wire.
- Sometimes two senders send at the same time, and so their packets "collide" on the wire and get garbled. The network hardware can usually detect this "collision" and so know to stop transmitting, as those sends are ruined.
- The senders follow a "wait/re-transmit" protocol to re-send packets -- wait a random amount of time -- one of Metcalfe's breakthrough ideas -- and then try again when the wire is quiet. If the senders each tried to re-send immediately .. the sends would just collide again! By waiting a random amount of time, the two senders coordinate that one goes first, then the other.
- If we have multiple computers using the network at the same time .. this all makes ethernet a little unpredictable in terms of performance; it's hard to say precisely how long it's going to take to get a packet through. In practice, it works incredibly well, getting great performance with very little networking hardware.

WI-FI -- SAME STRATEGY



- Wi-Fi wireless networking
- Similar strategy to Ethernet (simplifying)

  --Every computer has a radio

  --The "air" is the shared medium

  --One computer transmits at a time

  --Everyone listens

ETHERNET DESIGN SUMMARY

Ethernet is a nice example of getting multiple distributed actors to collaborate without relying on a central authority. This theme re-appears at the larger, world wide internet scale.

- **Shared** -- there's just the one wire and everybody uses it (cheap)

- **Distributed and Collaborative** -- no central control, depends on each computer following the collaborative protocol in good faith
- **Insecure** -- not to hard to listen and pick up packets not intended for you (shared)
- **Performance degrades** but does not break as more computers use the shared medium
- **Incredibly successful design strategy** -- getting great performance out of minimal hardware
- Wi-Fi is very similar

The Internet - TCP/IP

INTERNET - TCP/IP STANDARDS

- Previously .. LAN, e.g. ethernet, Wi-Fi, one house
- Internet - world-wide network
- Like a phone system for computers
- --Every computer has a unique address
- --Every computer can try to "call" any other computer

- TCP/IP Standards, 1974, government sponsored research
- Free and open standards, vendor neutral -- successful pattern for infrastructure

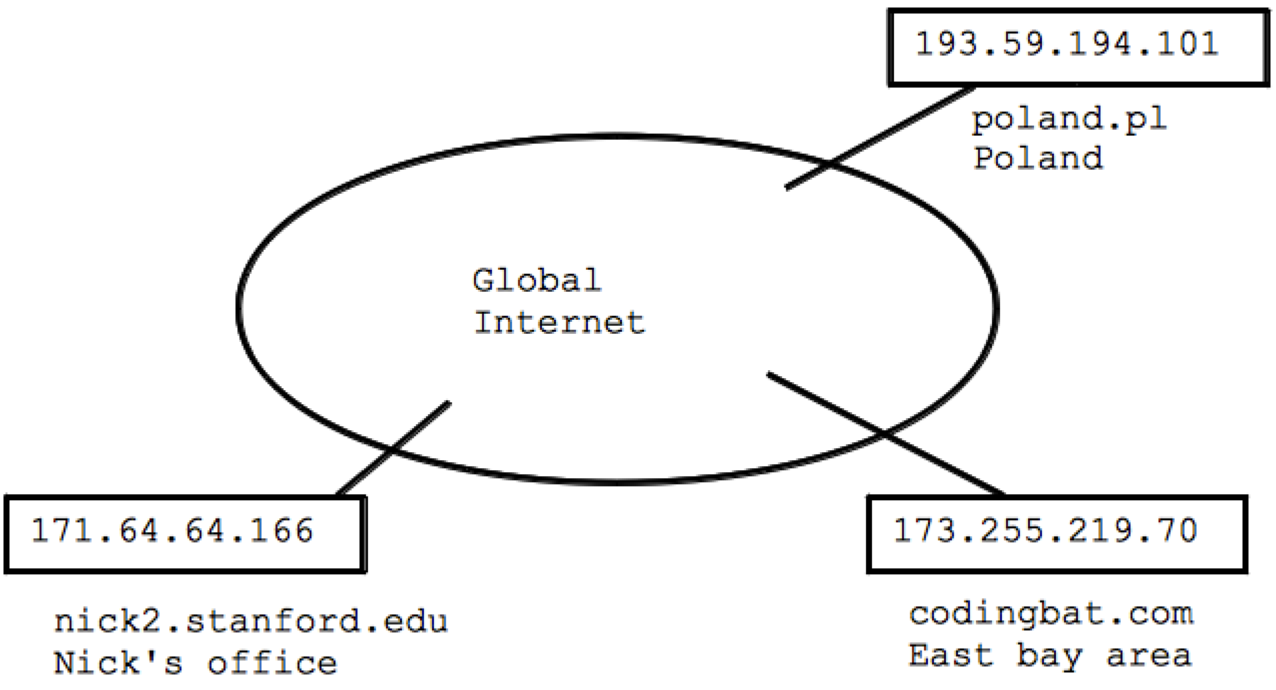
The previous LAN examples connect computers all on the same LAN. Now we will scale the problem up to send packets between any two computers on earth.

The worldwide Internet is built on the TCP/IP family of standards (Transmission Control Protocol / Internet Protocol) which solves the larger problem of sending packets between computers across the whole internet. These are free and open, vendor-neutral standards which is probably the reason they have been so incredibly successful.

IP ADDRESS

- Every computer on the internet has an IP address
- Here looking at IP v4 addresses, v6 on the horizon
- e.g. 171.64.64.166
- **4 bytes**
- Left part encodes "neighborhood" on internet
- Just like phone, 650-725-0000
- e.g. 171.64.xxx.xxx generally Stanford campus
- e.g. 171.64.64.xxx my floor of Gates building

- Sandra Bullock blooper in The Net: "23.75.345.200" Goodbye Oscar award!



Every computer on the internet has an "IP address" that identifies it (like a phone number). The IP address is 4 bytes, written between dots, like "171.64.2.3". The left part of the address encodes in part where that IP address is in the whole internet -- for example any 171.64.(anything) is part of Stanford (like the area code of a phone number). More specifically, in my part of the Gates building, all the IP addresses begin 171.64.64.XX varying only in that last byte.

Sandra Bullock: there's a blooper in the movie The Net where the IP address "23.75.345.200" is shown. This is not a valid address, since 345 is larger than the largest possible byte value which is 255.

DOMAIN NAMES

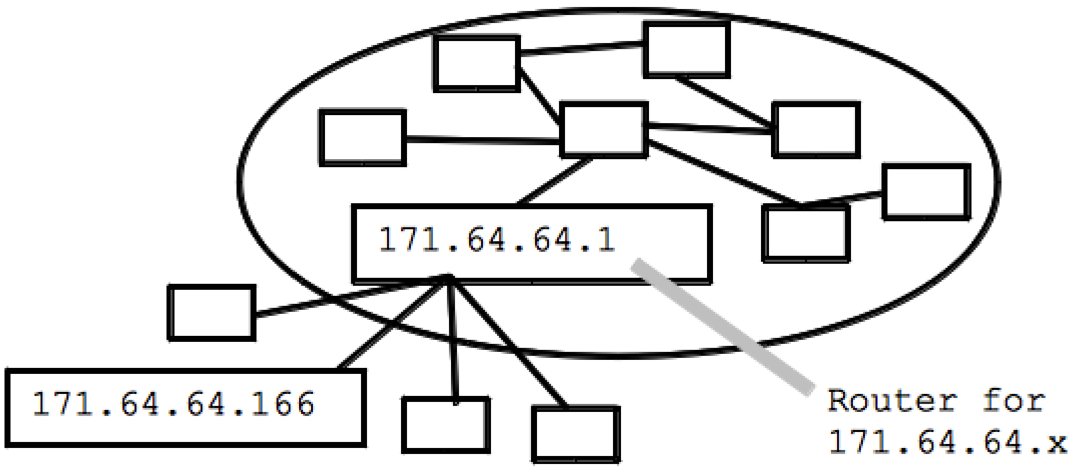
- Domain names, essentially alternate names for IP addrs www.google.com

nick2.stanford.edu

- Domain names are easy for people to remember and type
- Domain system can look up an IP addr from a domain name
- So when you use a domain name, it is looked up to get an IP addr for the actual packets

ROUTER

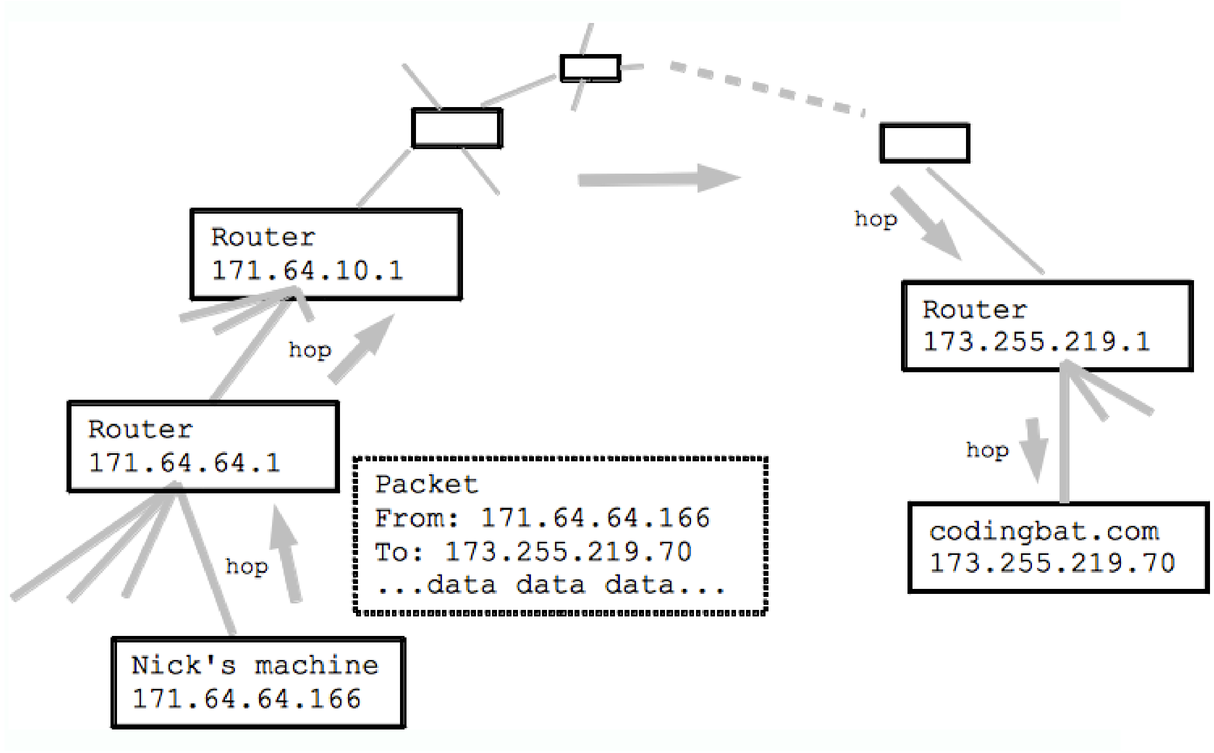
- Router has multiple connections, copies/routes packets between them
- My office computer is at 171.64.64.166
- That computer connects "upstream" to router 171.64.64.1
- That router handles traffic for a few local computers
- Left side of computer and router IP addresses typically the same -- same neighborhood



The most common way for a computer to be "on the internet" is to establish a connection with a "router" which is already on the internet. The computer establishes a connection via, say, ethernet to communicate packets with the router. The router is "upstream" of the computer, connecting the computer to the whole internet. For example, the computer in my Stanford office has IP address 171.64.64.166, and it has a one-hop ethernet connection to its router upstream at 171.64.64.1, and this router handles packets for my computer. Often the router's IP address will end in .1, such as my router's 171.64.64.1. Typically the IP address of the computer and its router will look at the same on the left side, since they are in the same "neighborhood" of the internet.

HOW DOES A PACKET GET ACROSS THE INTERNET?

Answer: Hop Hop Hop Hop Hop



- Suppose 171.64.64.166 sends a packet over to 173.255.219.70
- IP packet marked with ultimate From:/To: IP addrs
- Key idea: packet proceeds one hop at a time
- Hop 1: 171.64.64.166 sends packet up to its router
- Hop 2: 171.64.64.1 sends packet up to its bigger router
- Hop hop hop, over to destination, 20-30 hops typically
- Analogy: source capillary up to major artery, over, and down to destination capillary

Suppose my computer at 171.64.64.166 wants to send a packet to a computer at 173.255.219.70 somewhere out on the internet (actually that's the codingbat.com server I administer). The Internet is essentially made of a big web of routers talking to each other.

1. My computer prepares an IP packet which includes in particular From:/To: information as IP addresses, like this: (IP Packet From:171.64.64.166 To:173.255.219.70 data data data data).

2. My computer sends that IP packet to my upstream router, one hop, over ethernet. This is the "first hop" of the packet on its journey.

3. The 171.64.64.1 router looks at the To:/From: of the packet and forwards it to the next router, one hop closer to its ultimate destination. Essentially, the router has its own upstream router which is bigger and knows more about the layout of the internet. The packet is forwarded, one hop at a time, until it reaches its ultimate destination. Each router does not need to know the whole route to the destination; each router just needs to know which way to send the packet to get it one-hop closer to its destination. The routers look at the left part of the IP address to get the packet to the right neighborhood -- 173.255.x.x -- with the right part of the address -- x.x.219.70 -- coming into play only when the packet is near its ultimate destination.

ROUTER ANALYSIS

- Each router knows enough to figure the next hop, not the whole route
- The original computer does not typically know much, delegating to routers
- "Core" routers, towards the middle

  --Many connections

--Higher capacity

--More expensive

--More complicated routing decisions

- Routers measure connection functionality/breakage all the time, route around
- Routers are another distributed, collaborative system

The routing of a packet from your computer is like a capillary/artery system .. your computer is down at the capillary level, your packet gets forwarded up to larger and larger arteries, makes its way over to the right area, and then down to smaller and smaller capillaries again, finally arriving at its destination. The ultimate destination puts all the packets back together in the right order to recover the original image file or whatever. The routers at the ends have a trivial upstream/downstream configuration, so the next hop for a packet is pretty simple. More central "core" routers tend to have several possible outgoing connections, so they have a more complicated choice about which link to use for the next hop.

The routers, collectively, measure what networks are reachable over what links, and dynamically adjust what links to use for each packet. One simple metric would be to route packets the way that takes the fewest number of hops. In reality, the metrics used are more complex than this. The routing system resilient to router hardware failures, overloading of certain links due to normal traffic, and links going down. The path taken by an IP packet can change from minute to minute. The routers are another example of a distributed, collaborative system. The old joke is that the backhoe is the IP packet's natural predator in the wild, as construction will sometimes slice through an important data cable, suddenly breaking a link in use. The routers "route around" such damage automatically.

Note that my computer does not need to know the layout of the internet. My computer just needs to have a connection to its upstream router, and the router, and its upstream router etc., will handle the routing from there.

Very broadly speaking, most data you get or send on the Internet goes in packets which take more than 10 but less than 30 hops from origin to destination.

SPECIAL "LOCAL" IP ADDRESSES

- Note that 10.x.x.x and 192.168.x.x addresses are special "local" IP addresses
- These addresses are not valid out on the internet at large
- These are translated to a real IP addr as a packet makes its way
- Frequently given out by Wi-Fi routers .. why I mention them

WHAT DOES IT MEAN TO BE ON THE INTERNET?

- On the internet -- e.g. connect to a Wi-Fi router
- 1. Computer connects to an upstream router to handle traffic. Most Wi-Fi access points combine Wi-Fi radios and a router.
- 2. The router typically gives the computer an IP address to use

- The computer cannot pick an arbitrary IP address, since the left part of the address depends on the location on the internet ... details known by the router
- 3. DHCP "Dynamic Host Configuration Protocol" - automatically configure network settings to work locally. Computers very often use this feature to get needed network configuration from the router automatically.

So what does it mean for a computer to be on the internet? Typically it means the computer has established a connection with a router. The commonly used **DHCP** standard (Dynamic Host Configuration Protocol), facilitates connecting to a router; establishing a temporary connection, and the router gives your computer an IP address to use temporarily. Typically DHCP is used when you connect to a Wi-Fi access point.

Experiment: bring up the networking control panel of your computer. It should show what IP address you are currently using and the IP address of your router. You will probably see some text mentioning that DHCP is being used.

PING

"Ping" is an old and very simple internet utility. Your computer sends a "ping" packet to any computer on the internet, and the computer responds with a "ping" reply (not all computers respond to ping). In this way, you can check if the other computer is functioning and if the network path between you and it works. As a verb, "ping" has now entered regular English usage, meaning a quick check-in with someone.

Experiment: Most computers have a ping utility, or you can try "ping" on the command line. Try pinging www.google.com or nick2.stanford.edu (171.64.64.166, nick's desktop computer). Try pinging poland.pl ... much farther away from Stanford.

Here I run the "ping" program for a few addresses, see what it reports

```
$ ping www.google.com  # I type in a command here

PING www.l.google.com (74.125.224.144): 56 data bytes

64 bytes from 74.125.224.144: icmp_seq=0 ttl=53
time=8.219 ms

64 bytes from 74.125.224.144: icmp_seq=1 ttl=53
time=5.657 ms

64 bytes from 74.125.224.144: icmp_seq=2 ttl=53
time=5.825 ms

^C                       # Type ctrl-C to exit

--- www.l.google.com ping statistics ---

3 packets transmitted, 3 packets received, 0.0% packet
loss

round-trip min/avg/max/stddev = 5.657/6.567/8.219/1.170
ms

$ ping nick2.stanford.edu

PING nick2.stanford.edu (171.64.64.166): 56 data bytes

64 bytes from 171.64.64.166: icmp_seq=0 ttl=63 time=1.019
ms

64 bytes from 171.64.64.166: icmp_seq=1 ttl=63 time=3.638
ms

64 bytes from 171.64.64.166: icmp_seq=2 ttl=63 time=3.566
ms

^C

--- nick2.stanford.edu ping statistics ---
```

```
3 packets transmitted, 3 packets received, 0.0% packet
loss

round-trip min/avg/max/stddev = 1.019/2.741/3.638/1.218
ms
```

TRACEROUTE

- See series of hops
- First few hops in your IP neighborhood
- Farther away hops .. more milliseconds
- Note New York, London .. big jump in milliseconds
- Hops does not go up linearly with distance
- East bay - 30 miles from Stanford - 11 hops
- Poland - 8000 miles from Stanford - 27 hops

Traceroute is a program that will attempt to identify all the routers in between you and some other computer out on the internet - demonstrating the hop-hop-hop quality of the internet. Most computers have some sort of "traceroute" utility available if you want to try it yourself (not required). Some routers are visible to traceroute and some not, so it does not provide completely reliable output. However, it is a neat reflection of the hop-hop-hop quality of the internet. Here's an example traceroutes from my office:

```
$ traceroute -q 1 codingbat.com   # I type in a command
here

traceroute to codingbat.com (173.255.219.70), 64 hops
max, 52 byte packets

 1  yoza-vlan70 (171.64.70.2)  2.039 ms

 2  bbra-rtr-a (171.64.255.129)  0.932 ms
 3  boundarya-rtr (172.20.4.2)  3.174 ms

 4  dca-rtr (68.65.168.51)  27.085 ms

 5  dc-svl-agg1--stanford-10ge.cenic.net (137.164.50.157)
2.485 ms

 6  dc-oak-core1--svl-agg1-10ge.cenic.net (137.164.47.123)
3.262 ms

 7  dc-paix-px1--oak-core1-ge.cenic.net (137.164.47.174)
4.046 ms

 8  hurricane--paix-px1-ge.cenic.net (198.32.251.70)
14.252 ms

 9  10gigabitethernet1-2.core1.fmt1.he.net
(184.105.213.65)  9.117 ms

10  linode-llc.10gigabitethernet2-3.core1.fmt1.he.net
(64.62.250.6)  4.975 ms

11  li229-70.members.linode.com (173.255.219.70)  4.761 ms

$ traceroute -q 1 poland.pl

traceroute to poland.pl (193.59.194.101), 64 hops max, 52
byte packets

 1  yoza-vlan70 (171.64.70.2)  1.573 ms

 2  bbra-rtr-a (171.64.255.129)  1.106 ms

 3  boundarya-rtr (172.20.4.2)  32.970 ms

 4  dca-rtr (68.65.168.51)  1.530 ms

 5  dc-svl-agg1--stanford-10ge.cenic.net (137.164.50.157)
1.359 ms

 6  dc-svl-core1--svl-agg1-10ge.cenic.net (137.164.47.121)
1.967 ms

 7  dc-svl-isp1--svl-core1-10ge.cenic.net (137.164.47.132)
1.728 ms

 8  xe-4-1-2.edge1.sanjose1.level3.net (4.53.16.185)
1.687 ms
```

```
 9  vlan60.csw1.sanjose1.level3.net (4.69.152.62)  2.210
ms

10  ae-61-61.ebr1.sanjose1.level3.net (4.69.153.1)  1.637
ms

11  ae-2-2.ebr2.newyork1.level3.net (4.69.135.186)  70.263
ms

12  ae-72-72.csw2.newyork1.level3.net (4.69.148.38)
72.573 ms

13  ae-91-91.ebr1.newyork1.level3.net (4.69.134.77)
72.235 ms

14  ae-42-42.ebr2.london1.level3.net (4.69.137.69)
139.832 ms

15  ae-58-223.csw2.london1.level3.net (4.69.153.138)
146.678 ms

16  ae-21-52.car1.london1.level3.net (4.69.139.98)
172.864 ms

17  212.113.16.110 (212.113.16.110)  139.842 ms

18  ae1-0.ams-koo-score-1-re0.interoute.net
(84.233.190.57)  169.460 ms

19  ae0-0.ams-koo-score-2-re0.interoute.net (84.233.190.2)
182.792 ms

20  ae1-0.ber-alb-score-1-re0.interoute.net
(84.233.190.22)  168.766 ms

21  gi1-0.waw-002-access-1002.interoute.net
(84.233.213.33)  168.403 ms

22  po6-0.waw-002-access-1001.interoute.net
(84.233.213.45)  170.038 ms

23  195.81.72.214 (195.81.72.214)  168.413 ms

24  pkp-gw-ae0-100.core.nask.pl (195.187.255.156)  179.974
ms

25  pw-gw0-ae1-100.core.nask.pl (195.187.255.152)  168.538
ms
```

```
26  helm-at2-0-1.nask.waw.pl (194.92.0.158)  169.701 ms

27  193.59.194.101 (193.59.194.101)  169.830 ms
```

The numbers down the left side are the number of "hops" to that machine. The "ms" figures are the number of milliseconds (1 ms = 1 thousandth of a second) it took for the send/reply. Notice that as the hops get further away, it does roughly take more milliseconds. The first few hops are Stanford addresses, then the route goes over some provider, until it arrives at Linode, which is the company that provides the hardware where codingbat.com currently lives. Small mystery: it seems like the first hop should be 171.64.64.1 which is the first router from my office; apparently that router is invisible to traceroute.