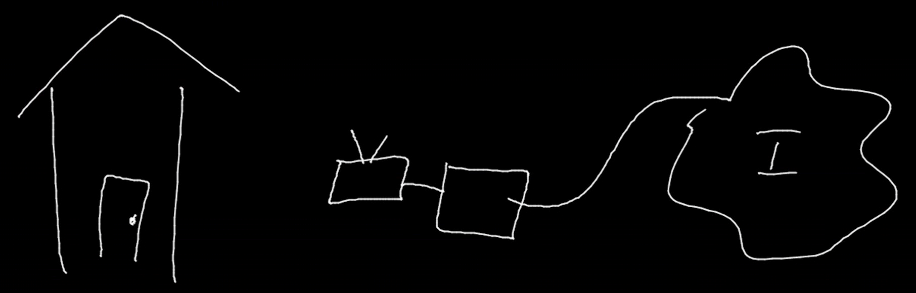
# [Internet](https://cs50.harvard.edu/technology/2017/notes/internet/#internet)

by Spencer Tiberi

## [Introduction](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=0m10s)

* We use the internet on a daily basis and have constant access and connectivity
* Home network



* + Cable modem, DSL modem, or FIOS device
    - Connects to the internet
    - Pay monthly for an ISP (Internet Service Provider)
      * Verizon, Comcast, etc.
    - Could have built in wireless connectivity for your devices
      * May need an additional home router
        + Devices connect to a router via cables or wifi

## [IP](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=2m13s)

* Every computer on the internet has an IP (Internet Protocol) address
  + Of the form #.#.#.#
    - Four numbers separated by dots of the values 0-255
    - Other IP address formats exist today as well
  + Like postal addresses, they uniquely identify computers on the internet
    - Any device connected to the internet has an IP address
      * Allows other computers to talk to it
* ISPs assign a IP address to your computer (router)
  + Used to be physically configured
  + DHCP (Dynamic Host Configuration Protocol)
    - Software that ISPs provides to allow your computer to request an IP address
    - DHCP servers respond with a specific IP address for your Home
  + Multiple devices can connect to your home network
    - The home router supports DHCP and assigns IP addresses to your devices

## [DNS](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=6m43s)

* We access websites using domain names (Facebook.com, Google.com, etc.), but it turns out that these sites too have IP addresses
* DNS (Domain Name System) servers convert domain names into IP addresses

## [Packets](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=8m21s)

* Computers communicate by sending packets, which are like virtual envelopes sent between computers
  + Ultimately still 0s and 1s
* As an analogy, suppose we want to find a cat image on the internet
* So, we send a request to a server, say Google, like “get cat.jpg”
  + We place this request in an envelope
* On the envelope, we list out IP as the return address
* However, for the recipient of the request, we don’t know the IP address for Google
  + Have to rely on DNS
  + Send a request to our ISPs DNS server for Google’s IP address
    - If the ISP’s DNS server doesn’t know a website’s IP address, it has been configured to ask another DNS server
    - There exist root servers that know where to look to for an IP address if it exists
* After sending the request off, we’ll get a response ms later



* The cat will be sent back in one or more packets
  + If the cat image is too large for a single envelope, sending it in one packet could take up internet traffic
  + To solve this, Google will divide the cat image into smaller fragments
    - Put the fragments into different envelopes
    - Write information on the envelopes
      * Return address: Google’s IP address
      * Delivery address: Our IP address
      * List the number of packets on each envelope (1 of 4, 2 of 4, etc.)

## [TCP/IP](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=14m15s)

* IP goes beyond addresses
  + Set of conventions computers and servers follow to allow intercommunication
* Fragmentation like in the envelope example are supported by IP
  + If missing a packet, you can logically infer which packet you’re missing based on the ones received
    - However, IP doesn’t tell computers what to do in this case
* TCP (Transmission Control Protocol) ensures packets can get to their destination
  + Commonly used with IP (TCP/IP)
  + Supports sequence numbers that help data get to its destination
    - When missing a packet, a computer can make a request for the missing packet
    - The computer will put packets together to get a whole file
  + Also includes conventions for requesting services (port identifiers)
    - To make sure Google knows we’re requesting a webpage and not an email or other service

## [Ports](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=18m14s)

* Per TCP, the world has standardized numbers that represent different services
* If 5.6.7.8 is Google’s IP address, 5.6.7.8;80 (port 80) lets use know that we want a webpage
  + 80 means http (hypertext transfer protocol)
    - The language that web servers speak
  + Google will send the request to their web server via http
* Many websites use secure connections with SSL or HTTPS, which uses the port 443
* Email uses port 25
* Other ports exist as well

## [Protocols](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=19m53s)

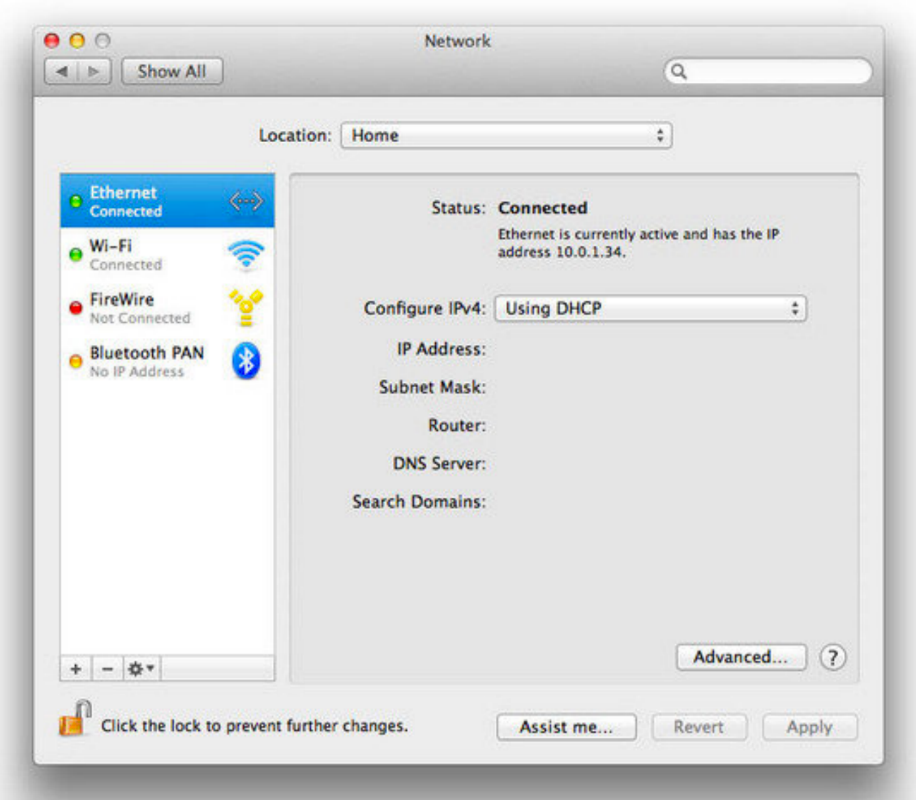
* Protocols are just sets of rules
  + Humans use these all the time, such as the protocol for meeting people: handshakes
* When a request is made to Google for an image, HTTP tells Google how to respond appropriately

## [UDP](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=21m12s)

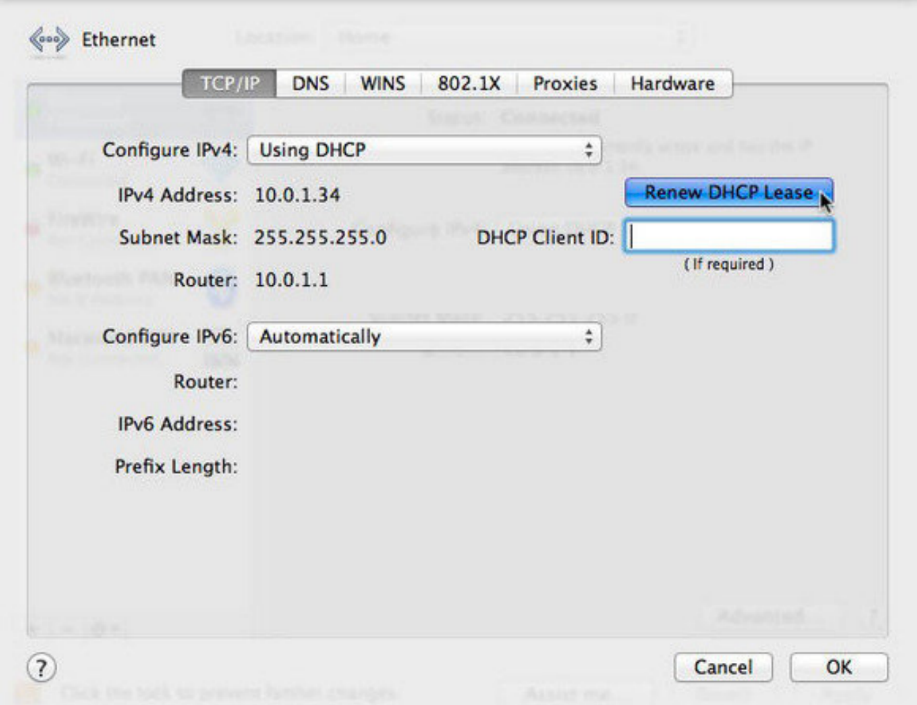
* User Datagram Protocol
  + Doesn’t guarantee delivery
  + Used for video conferencing such as FaceTime
    - Packets can be dropped for the sake of keeping the conversation flowing
  + Used anytime you want to keep data coming without waiting for a buffer to fill

## [IPs in More Detail](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=23m28s)

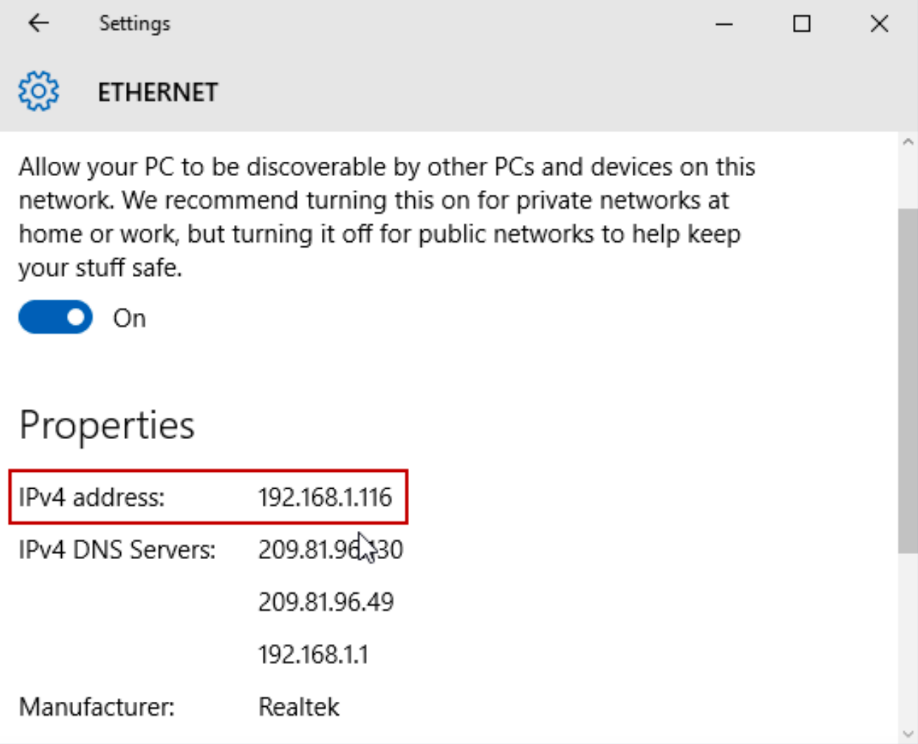
* IP addresses are limited
  + In the format #.#.#.#, each number is 8 bits, so 32 bits total
    - This yields 232 or about 4 billion possible addresses
      * We’re running out of addresses for all computers
  + Current version of addresses is IPv4
  + Moving towards IPv6
    - Uses 128 bits, yielding 2128 possible addresses
* How do you find your IP address?
* On a Mac, go to system preferences an poke around a bit



* Private addresses exist
  + 10.#.#.#, 192.168.#.#, or 172.16.#.#
  + Only with special configuration can someone talk to your computer
  + Your personal device is not a server, so people should not need to access them directly
    - Your device needs to request data from servers
  + Even email is stored on a server such as Gmail and your device makes a request to that server to access that email
* Looking at advanced settings…



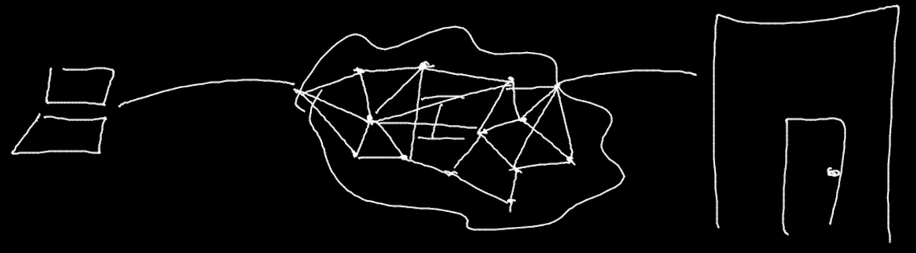
* + Subnet mask is used to decide if another computer is on the same network
  + Router (aka Gateway) has its own address
    - Routs data in different directions
* On windows:



* + Shows DNS servers as well

## [Routers](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=27m6s)

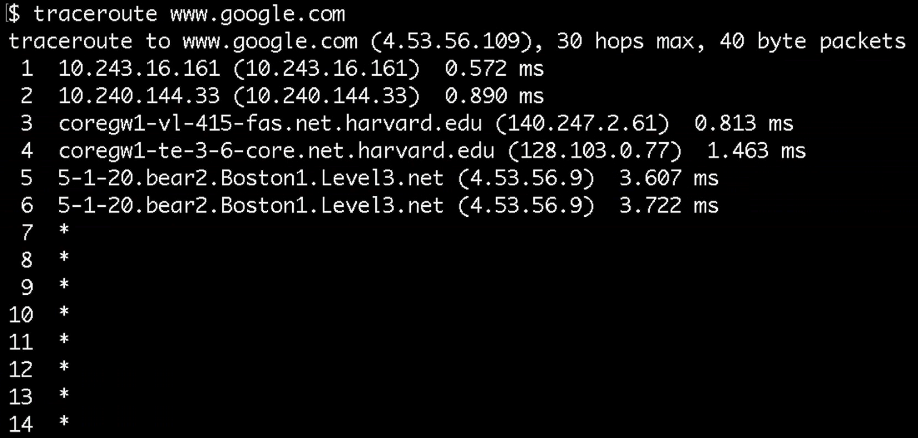
* Routers have bunches if wires coming and going out of them
  + They have a big table with IP addresses and where data should be routed to get to that destination
    - Often, the data is routed to some next router
* Routers purpose is to send data in the direction of a destination
  + The next router will send it to another until it reaches a destination



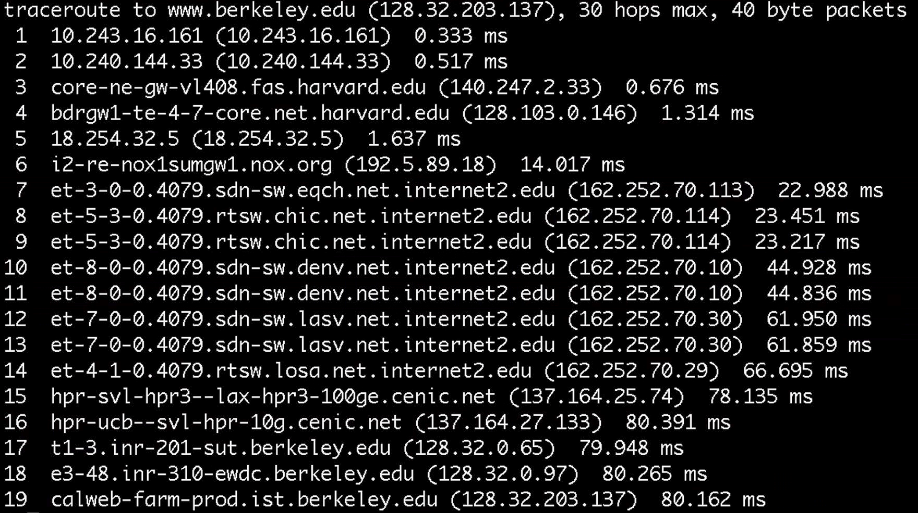
* The internet is a network of networks (with their own routers)
  + Often multiple ways to go from A to B
    - Based in US Military logic to prevent downtime if a particular router goes down
    - When multiple packets are sent, like cat.jpg from Google, they can each take a different path, still getting to their destination eventually
      * Sometimes the internet is busy and the quickest path changes

## [Traceroute](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=32m31s)

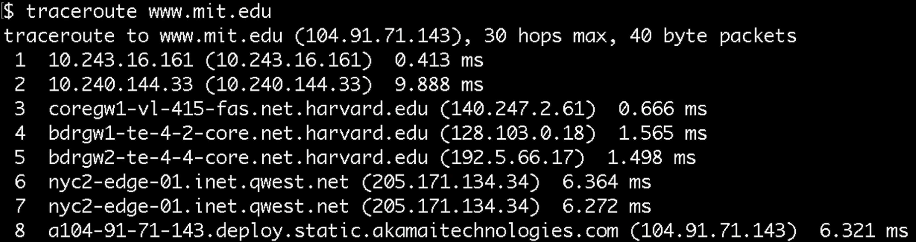
* How long does it take for this process of data transfer to take on the internet?
* Traceroute is a program that sends packets to each router on a path to a destination, reporting the time it takes to reach that router
* From Sanders Theatre to Google.com:



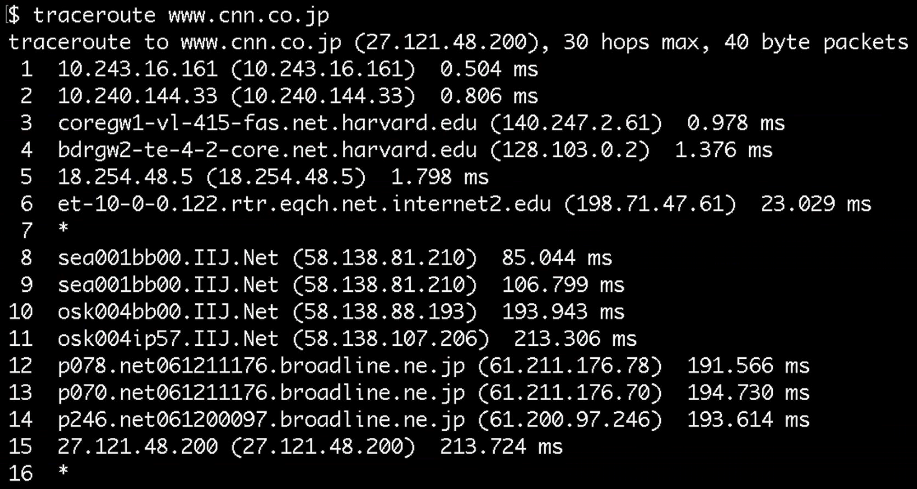
* + 1-2: A few unnamed routers at Harvard
  + 3-4: More Harvard routers
  + 5-6: Level3 is a ISP
  + 7+: The routers are denying the request
* From Sanders Theatre to Berkeley.edu



* + 6: Northern Crossroads
  + 7-14: A fast connection
    - 8-9: Chicago
    - 10-11: Denver
    - 12-13: Las Vegas
    - 14: Los Angeles
  + 19 is where it arrives at Berkeley in 80 ms!
* From Sanders Theatre to MIT.edu



* + 6-7: Goes to New York connectivity
  + 8: MIT’s website is outsourced to Akamai’s NYC servers
* From Sanders Theatre to CNN.jp



* + 9-10 jumps from Seattle to Osaka past an ocean!
    - Using undersea cabling

## [Undersea Cabling](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=40m56s)

* David shows a video about undersea cables

## [Cable Modem Demo](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=43m5s)

* David examines a home cable modem, focusing on its ports
  + Coaxial cable to plug into the wall
  + Phone jacks (RJ11) as many services are bundled together these days
  + Four jacks for ethernet cables (RJ45)
    - Devices can plug into these for internet connectivity
  + This modem has wifi support built in

## [Network Switch Demo](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=46m27s)

* David examines a network switch
  + A device that you can plug into your router to allow more connections for all your other devices

## [Home Router Demo](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=47m48s)

* David examines a home router
* Home routers can have wifi, firewall, and switching capabilities

## [Network Cable Demo](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=48m54s)

* David cuts open a network ethernet cable to examine its inner workings
* Inside a network cable are 8 wires of different colors
  + Some are for transmitting data, others for receiving data
  + Others still are for insulation and cancellation of interference

## [Closing Thoughts and Homework](https://video.cs50.net/cscie1a/2017/fall/lectures/internet?t=50m24s)

* For homework, find a device that looks like a modem or router and take a look at the connectors on the back of it
  + If brave, play around with unplugging cables
    - Note: Your internet may go down in the process, but can be easily restarted with the cables properly reconnected!
  + If you have a spare ethernet cable, take a look inside yourself
    - These are a bit harder to put back together!