Computer Networks

Socket API, HW 1 fundamentals

Autumn 2023

Administrivia - Course Structure

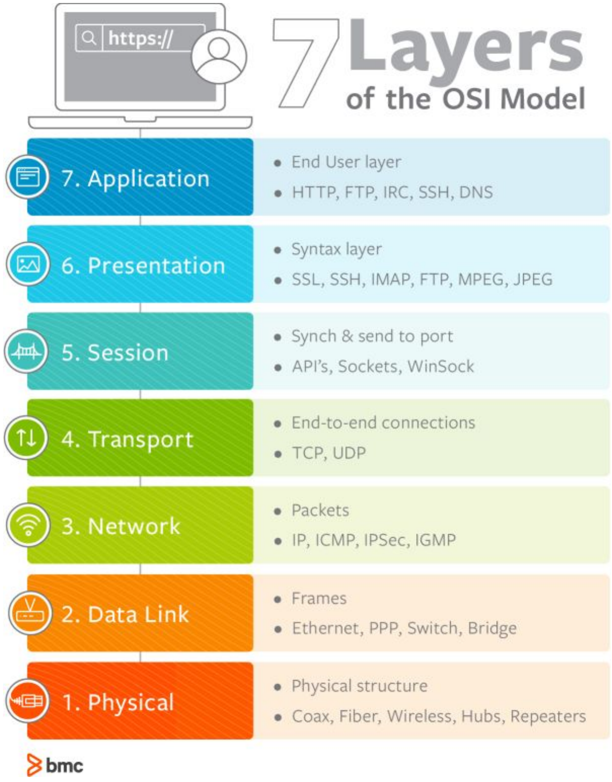
- Assignments
  - 3 group projects
    - Build a client and server application
    - Practice with Software-Defined Networking
    - Experimenting to learn about latency in real-world networks (Bufferbloat)
  - About 5 homework assignments (Gradescope)
    - Detailed practice with the concepts discussed in textbook & lecture
    - Conceptual overview
  - In-person Midterm & Final Exam
  - Occasional "surprise" quizzes
- Quiz Sections
  - Intro to labs (helpful hints!) + networking software
  - Re-explaining and clarifying conceptual topics (e.g. various protocols)
  - Practice with mechanics (e.g. calculations, algorithms, etc.)

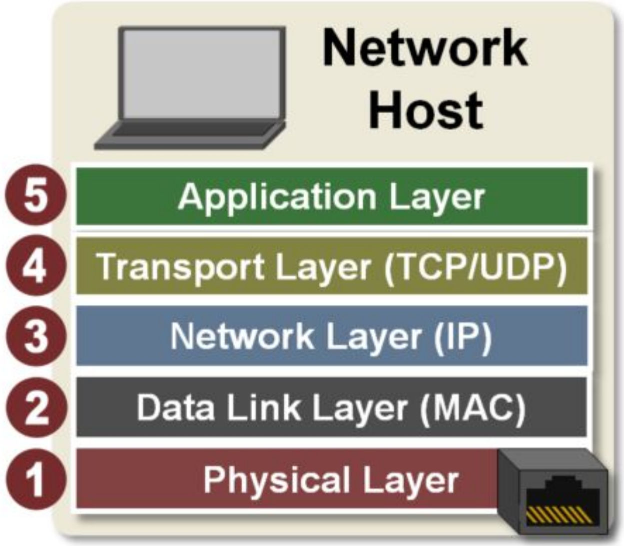
Administrivia

- Project 1 is will be out tomorrow!
    - Can be done in groups of 2-3
    - Can be done in any language (recommend Java / Python)
        - Future labs will be in Python
        - Intent is to allow you to become familiar with some languages Socket API!

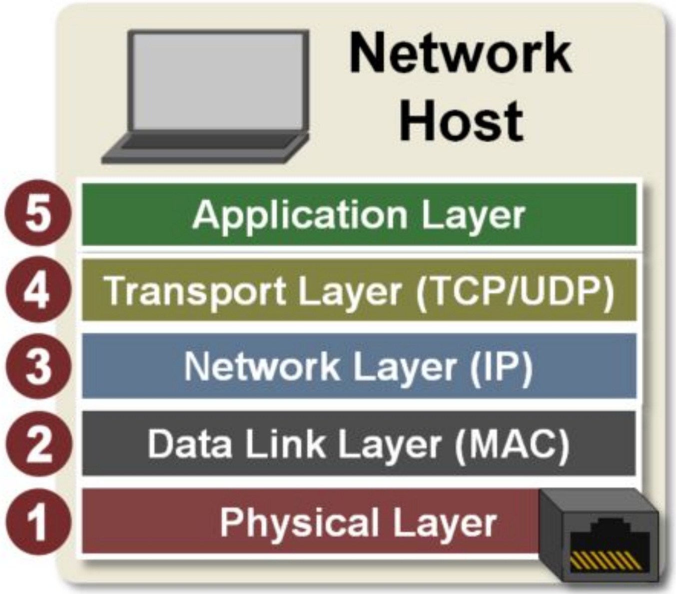
Socket API & Project 1

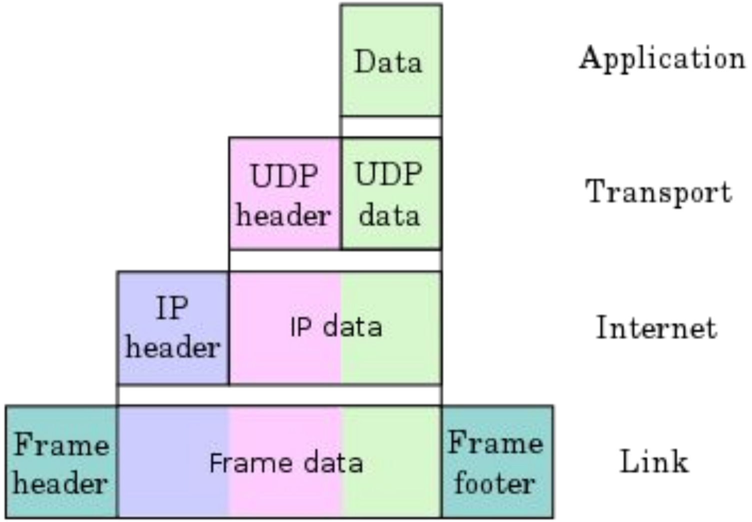
Network Stack - OSI Model vs TCP/IP Model





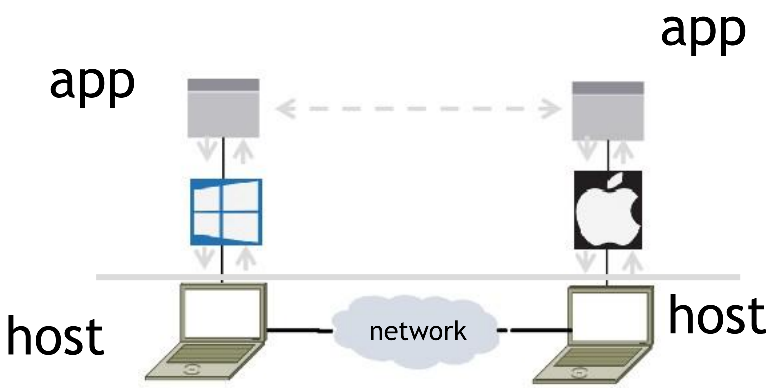
Network Stack - Packet Encapsulation

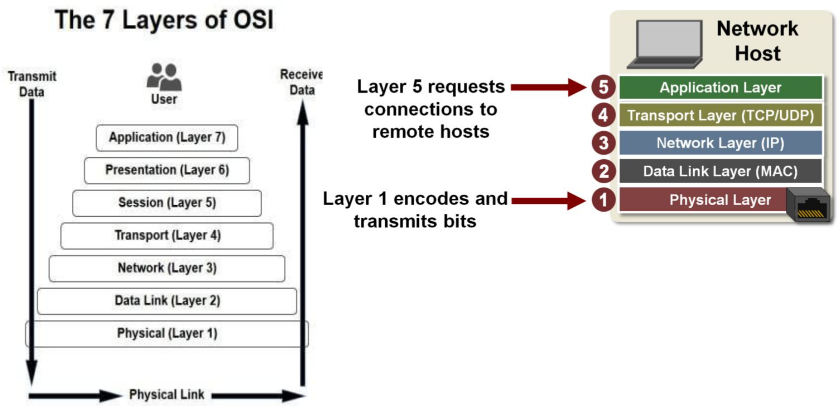




Network-Application Interface

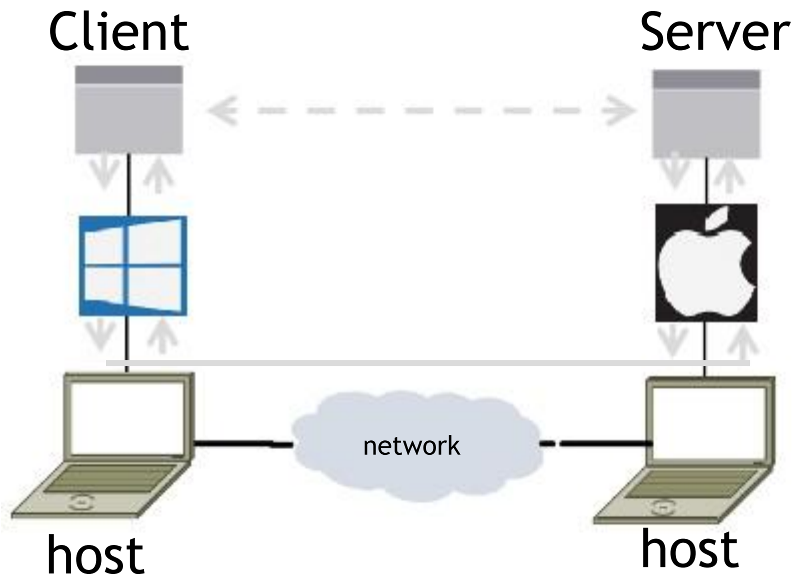
- Defines the operations that programs (apps) call to use the network
  - Application Layer API
  - Defined by the Operating System
    - These operations are then exposed through a particular programming language
    - All major Operating Systems support the Socket API
  - Allows two computer programs potentially running on different machines to talk
  - Hides the other layers of the network





Project 1

- Part 1: Simple Client
  - Send requests to attu server
  - Wait for a reply
  - Extract the information from the reply
  - Continue...
- Part 2: Simple Server
  - Server handles the Client requests
  - Multi-threaded
- This is the basis for many apps!
  - File transfer: send name, get file
  - Web browsing: send URL, get page
  - Echo: send message, get it back

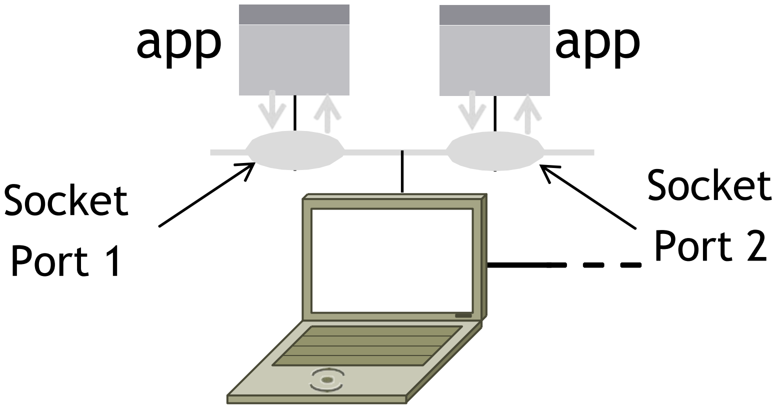


Socket API

- Simple application-layer abstractions (APIs) to use the network
  - The network service API used to write all Internet applications
  - Part of all major OSes and languages; originally Berkeley (Unix) ~1983
- Two kinds of sockets
  - Streams (TCP): reliably send a stream of bytes
    - Detects packet loss with timeouts (uses adaptive timeout protocol)
    - Uses flow control: similar to selective repeat
  - Datagrams (UDP): unreliably send separate messages

Ports

- Sockets let apps attach to the local network at different **ports**
  - Ports are used by OS to distinguish services / apps all using the same physical connection to the internet
  - Think of ports like apartment numbers, allowing mail sent to a shared building address (IP) to be sorted into the correct destination unit (application)

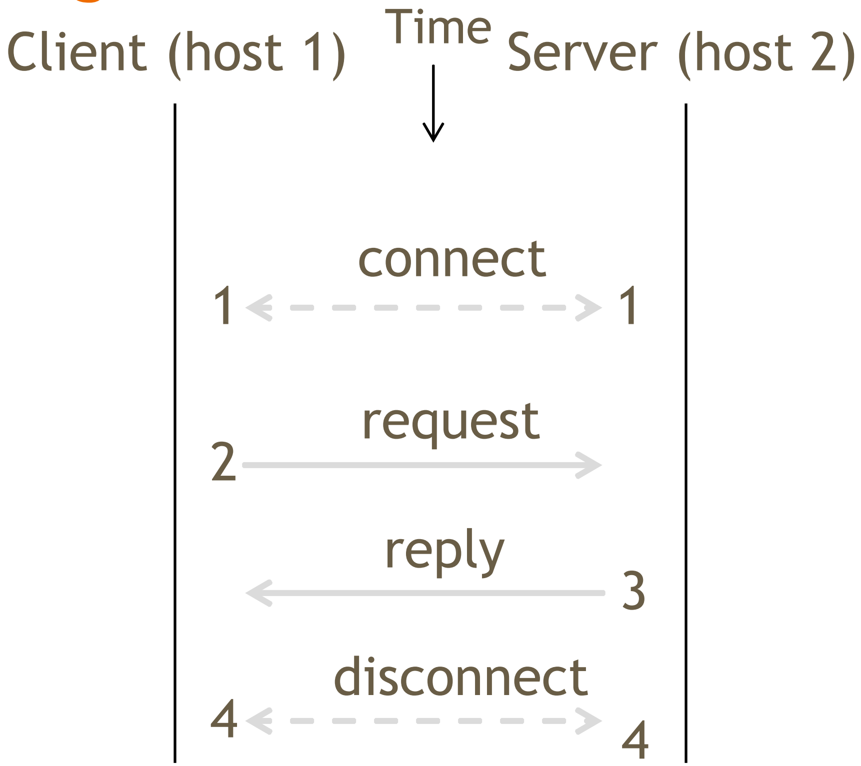


Socket API Operations

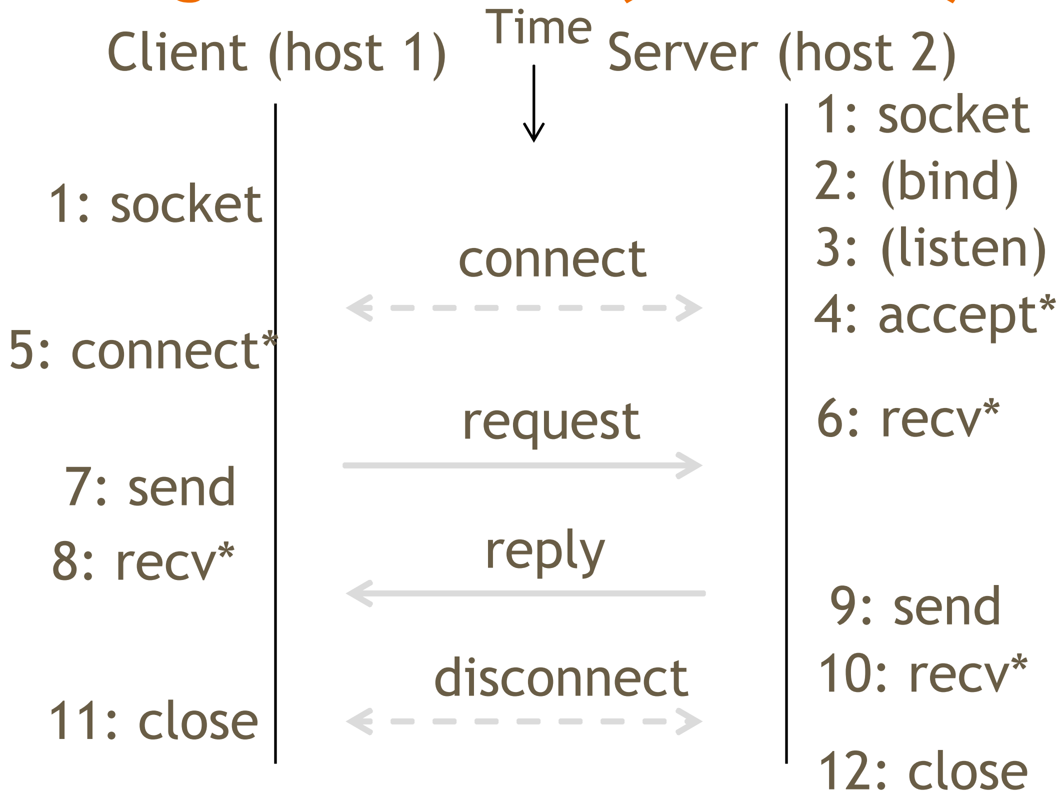
Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address (port) with a socket
LISTEN	Announce willingness to accept connections; (give queue size)
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

https://docs.oracle.com/javase/8/docs/api/java/net/Socket.html
https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html

Using TCP Sockets

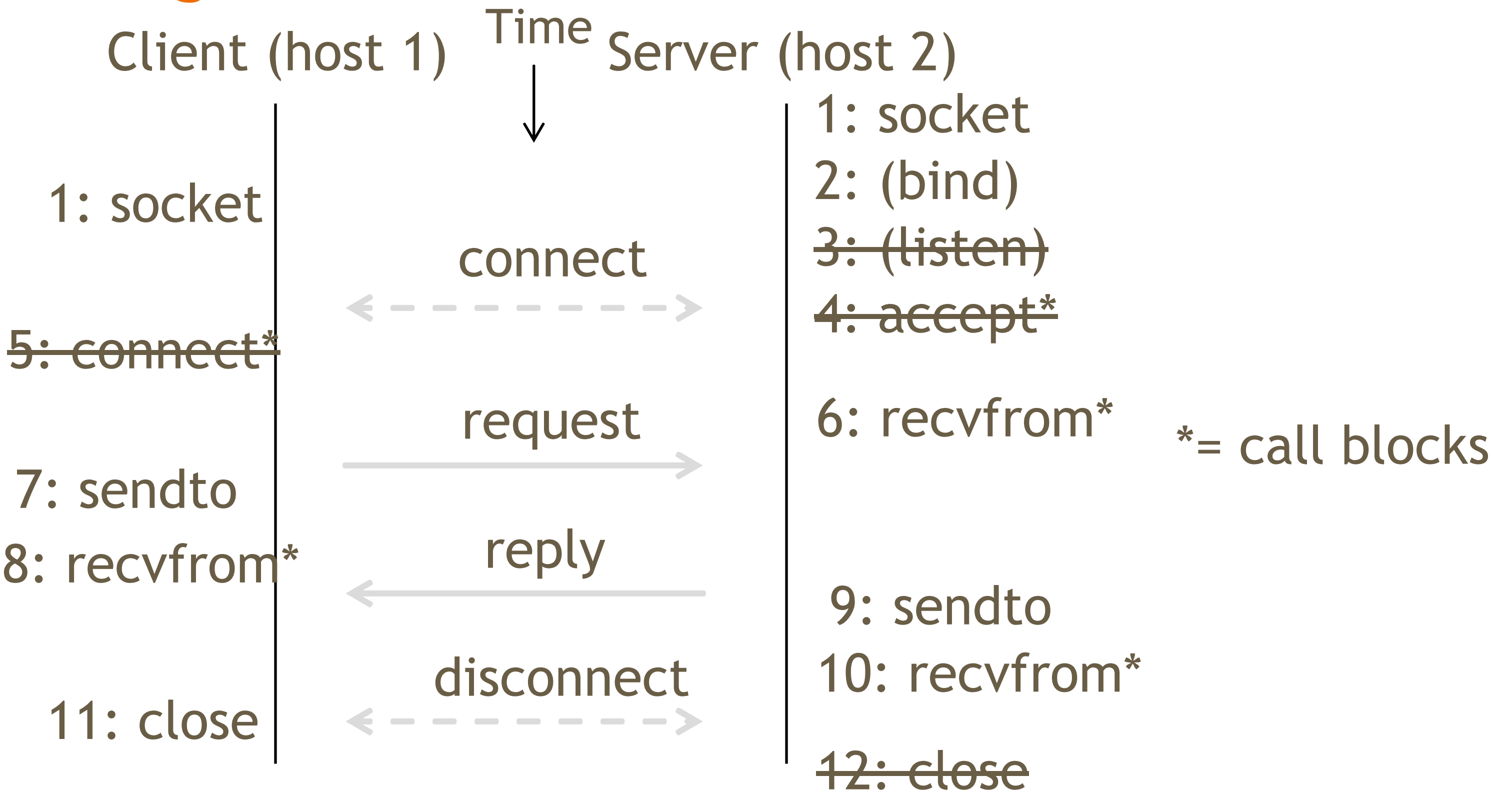


Using TCP Sockets (Continued)



*= call blocks

Using UDP Sockets



Client Program Outline

```
socket()        // make socket
getaddrinfo()   // server and port name
                // www.example.com:80
connect()       // connect to server

send()          // send request
recv()          // await reply [block]
...             // do something with
                //      data!
close()         // done, disconnect
```

Server Program Outline

```
socket()          // make socket
getaddrinfo()     // for port on this host
bind()            // associate port with socket
listen()          // prepare to accept connections
accept()          // wait for a connection [block]
…
recv()            // wait for request [block]
…
send()            // send the reply
close()           // eventually disconnect
```

Python Examples with socket

- Server

```
listener = socket.socket(socket.AF_INET,
                         socket.SOCK_STREAM)
listener.bind(server_address)

while True:
    try:
        connection, client_addr = listener.accept()
        try:
            connection.recv(n_bytes)
        finally:
            connection.close()
    except:
        listener.close()
```

- Client

```
socket = socket.socket(socket.AF_INET,
                       socket.SOCK_STREAM)
socket.connect(server_address)
socket.sendto(message, server_address)
socket.close();
```

- [Python socket documentation]
- [UDP socket example]
- [socketserver (a little overkill)]

Java Examples with Socket & ServerSocket

- Server

```
ServerSocket listener = new
        ServerSocket(9090);  try {
            while (true) {
                Socket socket = listener.accept();
                try {
                    socket.getInputStream();
                } finally {
                    socket.close();
                }
            }
        }
        finally {
            listener.close();
        }
```

- Client

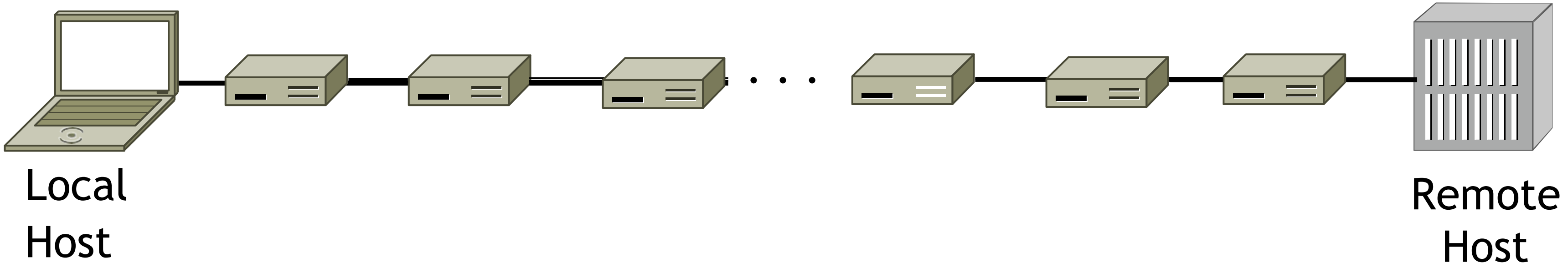
```
Socket socket = new Socket(server, 9090);
out =
        new PrintWriter(socket.getOutputStream(), true);
socket.close();
```

- http://cs.lmu.edu/~ray/notes/javanetexamples/
- https://docs.oracle.com/javase/tutorial/networking/datagrams/clientServer.html
- https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

HW1 Fundamentals

Traceroute

- We want to find network path from our system to a given remote host
- Core mechanism: Time-To-Live(TTL)
  - TTL defines the number of hops a packet will travel through until it is dropped
    - TTL is decremented every hop
    - Once TTL is 0 then the packet is dropped and a report is sent to the source

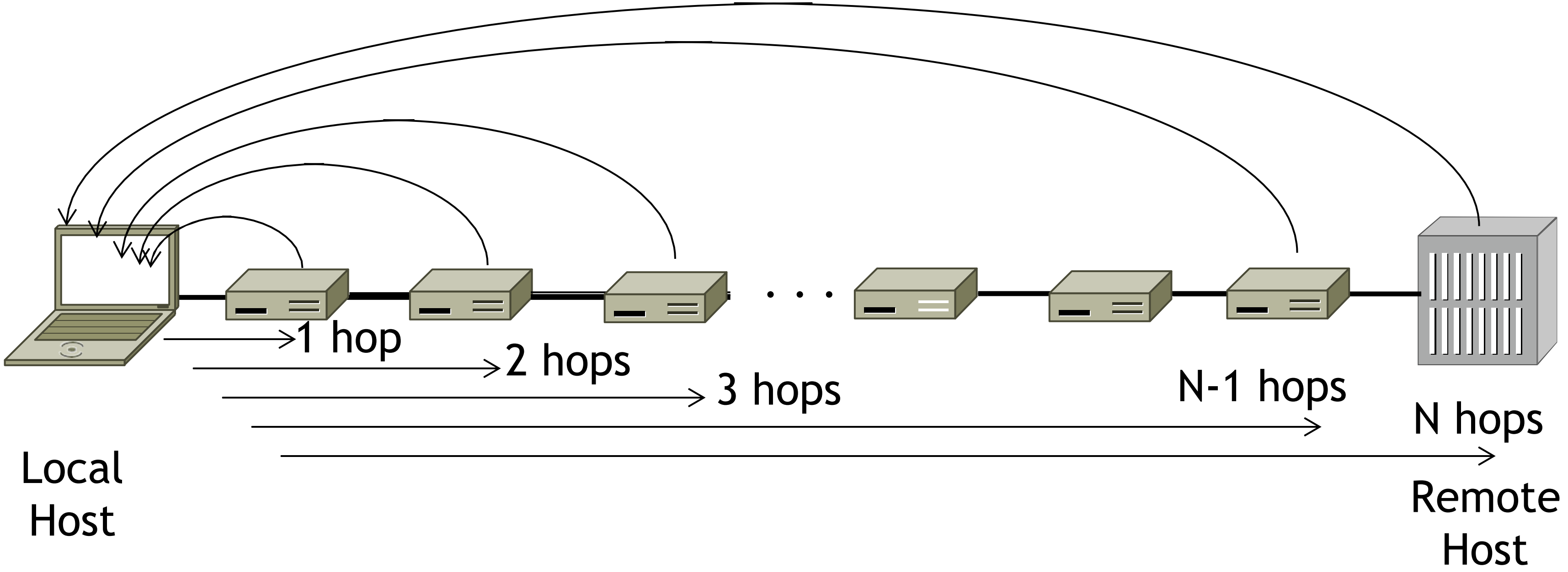


Resources:

- http://www.exit109.com/~jeremy/news/providers/traceroute.html
- https://serverfault.com/questions/6403/what-do-the-three-columns-in-traceroute-output-mean

Traceroute

- Traceroute sends out three packets per TTL increment
  - To have 3 trials of data for each hop distance
- Each data point corresponds to the total RTT time



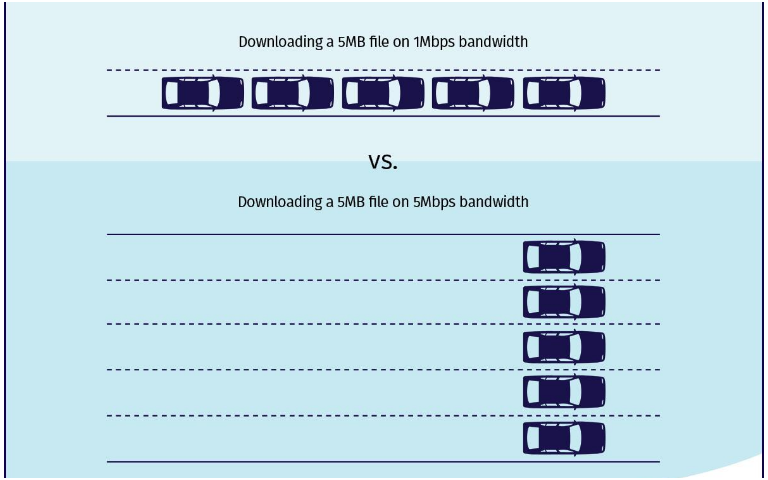
Using Traceroute

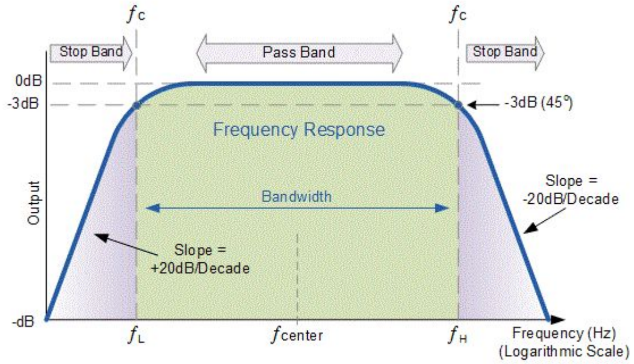
```
dominickta@Prota ~
☹  traceroute edstem.org
traceroute: Warning: edstem.org has multiple addresses; using 172.66.40.189
traceroute to edstem.org (172.66.40.189), 64 hops max, 52 byte packets
 1  10.18.0.2 (10.18.0.2)  6.327 ms  6.836 ms  8.570 ms
 2  lo0--5.uwcr-ads-1.infra.washington.edu (198.48.65.5)  6.998 ms  4.229 ms  11.492 ms
 3  10.132.5.66 (10.132.5.66)  11.876 ms  9.706 ms  5.816 ms
 4  10.132.255.17 (10.132.255.17)  11.205 ms  4.184 ms  3.535 ms
 5  10.132.255.18 (10.132.255.18)  15.567 ms  5.217 ms  2.527 ms
 6  ae20--4000.icar-sttl1-2.infra.pnw-gigapop.net (209.124.188.132)  7.411 ms  5.022 ms  20.059 ms
 7  six.as13335.com (206.81.81.10)  29.820 ms  105.949 ms  11.541 ms
 8  172.71.140.3 (172.71.140.3)  25.781 ms
    172.71.144.3 (172.71.144.3)  10.867 ms
    172.71.140.3 (172.71.140.3)  9.835 ms
 9  172.66.40.189 (172.66.40.189)  17.161 ms  9.327 ms  7.720 ms
```

Latency Stuff

Bandwidth

- Bandwidth (data rate): The number of bits that can be transmitted over a period of time
  - Units of bits per second (bps)
  - Confusingly also used to refer to the frequency range of a signal
    - In this case the units are given as hertz (Hz)
- Throughput: The measured performance of a system
  - Units of bits per second (bps)
- Bandwidth is a pipe and throughput is the water





Bandwidth & Transmission Time

Transmission time = Size of data / Bandwidth

- Transmission time of 1 bit of data at a bandwidth of 1 Mbps?
  - 1 bit / 1,000,000 bps = 1/1,000,000 seconds = 1 microsecond
- Transmission time of 1 bit of data at a bandwidth of 2 Mbps?
  - 1 bit / 2,000,000 bps = 1/2,000,000 seconds = 0.5 microseconds

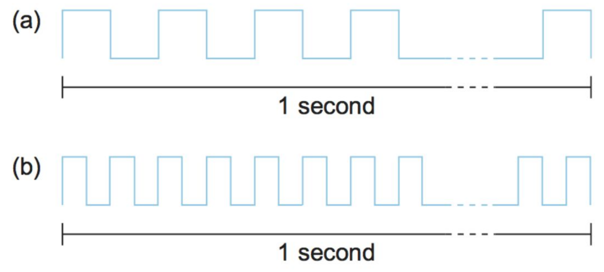
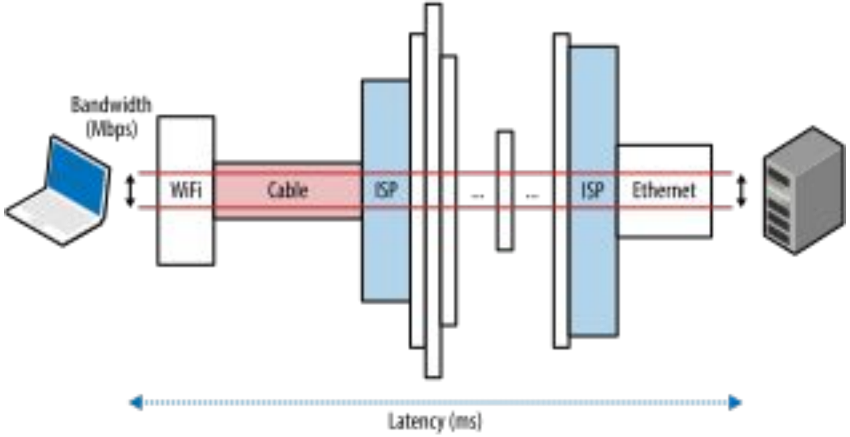


Figure 1.16.: Bits transmitted at a particular bandwidth can be regarded as having some width: (a) bits transmitted at 1 Mbps (each bit is 1 microsecond wide); (b) bits transmitted at 2 Mbps (each bit is 0.5 microseconds wide).

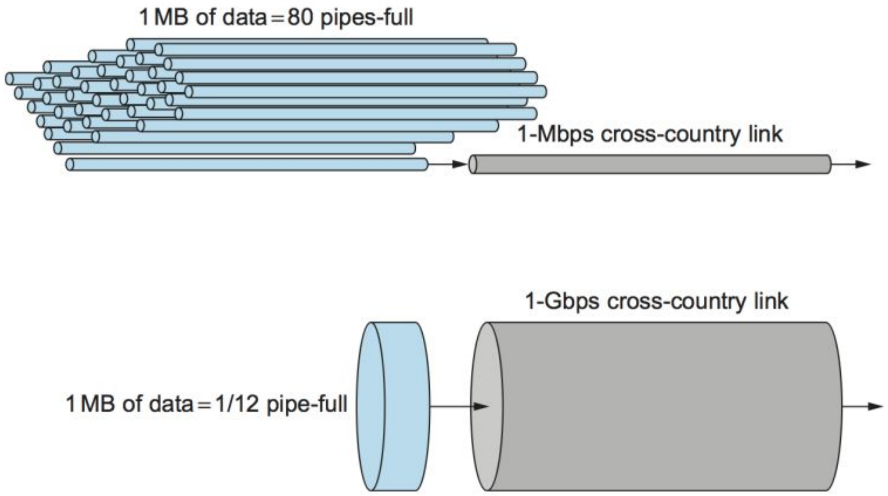
Latency

- Latency: Total time for a message to arrive on a network
  - Round trip time (RTT) is the latency for travel from source to destination to source
- Latency = Propagation + Transmit + Queue
  - Propagation = Distance / “Speed Of Light”
    - How long it takes for information to travel a distance from source to destination
    - Speed varies by medium
  - Transmit = Size / Bandwidth
    - How long it takes for information to be put onto the wire before travelling
  - Queue time
    - How long data has to wait until it’s their turn to be transmitted



Bandwidth-Delay Product

- Product between bandwidth and propagation delay
  - Units in bits (bps * s = b)
- Propagation delay is either **one way latency** or **RTT**
  - Usually RTT
- Conceptually defines the maximum amount of data that can be "in-flight" at a given time
  - think the amount of water in a pipe



Practice Exercises

Exercise 1

Suppose we have a network link with a **bandwidth of 10 Mbps**. We want to send a **100 KB file** to a friend somewhere else in the network. The RTT from us to our friend is **20 ms**. How long does it take for the entire file to be delivered?

- Transmit time = 100 KB / 10 Mbps = 100,000 B / 10 Mbps

  = 800,000 b / 10,000,000 bps = 0.08 seconds = 80 ms
- At t=80ms, the final bit of data is transmitted onto the wire.
  - This bit still needs to actually travel to the destination (propagation delay)
- At t=90ms, the final bit of data arrives at the destination

Exercise 2

Consider a point to point link **50 km in length**. Suppose the propagation speed is $2 * 10^8$ m/s. At what bandwidth in Mbps would the propagation delay equal the transmit delay for 100 B packets?

- Propagation delay = Distance / Speed Of Light (varies by medium)
  - = $50 * 10^3$ m / ($2 * 10^8$ m/s) = .00025 seconds = 250 microseconds
- Transmit = Size / Bandwidth
  - 250 microseconds = 100 B / x Mbps (solve for X)
  - 100 * 8 = 800 bits -> 800 bits / 250 μs = 3.2 Mbps

What about for 512 byte packets?

- 512 * 8 bits / 250 μs = 16.4 Mbps

Exercise 3

Suppose a 128-kbps point-to-point link is set up between Earth and a SpaceX colony on Mars. The distance from Earth to Mars (when they are closest together) is approximately 55 Gm, and data travels over the link at the speed of light ($3 * 10^8$ m/s)

- Calculate the minimum RTT for the link.
- Calculate the delay x bandwidth product for the link.
- Say your aunt Betty takes a selfie on Olympus Mons, and sends a 5 MB picture to you on Earth. How quickly after the picture is taken can you receive the image from Betty?

Exercise 3

Suppose a 128-kbps point-to-point link is set up between Earth and a SpaceX colony on Mars. The distance from Earth to Mars (when they are closest together) is approximately 55 Gm, and data travels over the link at the speed of light ($3 * 10^8$ m/s)

- Calculate the minimum RTT for the link.
  - RTT = 2 * Propagation delay = $2 * 55 * 10^9$ m / ($3 * 10^8$ m/s) = 2 * 184 = 368 seconds
- Calculate the delay x bandwidth product for the link.
  - delay x bandwidth = 368 seconds * ($128 * 10^3$ bps) = 5.888 MB
- Say your aunt Betty takes a selfie on Olympus Mons, and sends a 5 MB picture to you on Earth. How quickly after the picture is taken can you receive the image from Betty?
  - Transmit delay for 5 MB = 40,000,000 bits / ($128 * 10^3$ bps) = 312.5 seconds
  - Total time = transmit delay + propagation delay = 312.5 + 184 = 496.5 seconds = about 9 minutes

Thanks for coming!