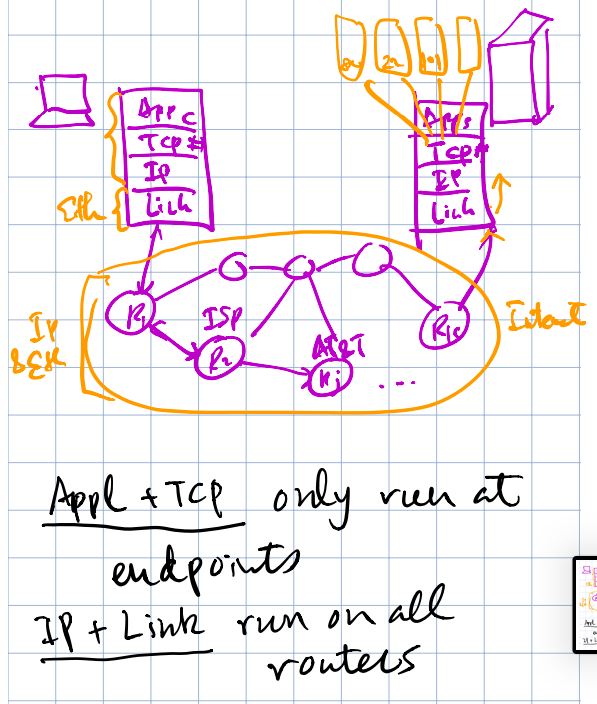
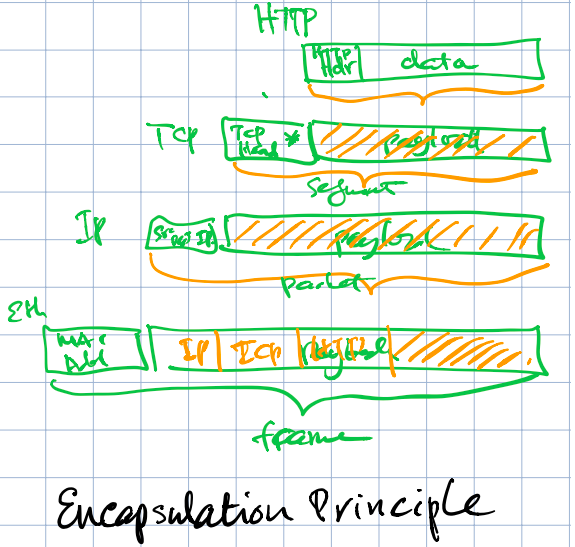
CSC 361

**1/8/19**

* VM
  + Username: csc361 PW: csc361
  + By default, should be attached to NAT.
  + Mininet
    - Run command: mn
    - Run as root: sudo mn -x

**1/9/18**

* **Unit 1: Internet Architecture**
  + No guarantee that it will receive every packet sent.
    - No guarantee because it makes it more complicated if we need that guarantee.
  + The internet only runs on IP.
    - Doesn’t care about the payload.
    - The endpoints will create some sort of TCP.
      * The endpoints guarantee connection.
      * The TCP is the part that makes it reliable.
      * TCP and the endpoints are responsible for re-transmission.
    - No concept of port number, or connection.
  + Internet Architecture has lots of performance issues.
  + Re-transmission
    - Done through timing.
    - If we don’t receive a response from the server after a certain amount of time we re-transmit our packet.
    - The tough part is figuring out how long to wait for time-out. (Bandwidth Delay Product)
  + TCP creates a header, describing the connection, and transmits it to the IP layer with the data, as a packet.
  + All the intelligence usually exists at the endpoints. The client and server application usually handle all the logic of how to do things. (End-to-end principle)
  + 4 Layer Model of the Intenet
    - Application Layer and Transport layer only exist at the endpoints.
    - Top three layers (application, transport, and network) exist in software.
    - The link layer is hardware.
    - Applications
      * HTTP, DNS, FTP, Email
      * Socket API sits between Application and Transport
    - Transport
      * TCP, UDP
      * UDP – User Datagram Protocol
        + IP with something added
        + No guarantee
      * TCP – More popular model.
        + Guarantee delivery in order
      * Port Number sits in the transport layer.
        + Port number identifies a service.
        + If you make a new application, you manually allocate a port number above 5000.
        + If you don’t care what port number it is, the application will dynamically create a port number for you.
    - Network
      * IP, ARP
    - Link
      * Ethernet
  + Encapsulation
    - HTTP
      * Header + Data = Payload
    - The whole payload is transmitted to the TCP layer. Then a port number is added into the header. It is then called a **segment**.
    - The whole **segment** is then transmitted to the IP layer and an IP address is added onto the header. It is then called a **packet**.
    - The whole **packet** is then transmitted to the link layer and then it adds on a MAC address to the header. It is then called a **frame**.
    - 

**1/11/19**

* **Exercises** – In tests and quizzes.
  + Each exercise allows you submit three times.
  + Doesn’t tell you which you got wrong.
  + No time limits.
* VPN
  + Will add a new TCP\* and IP\* to the packet.
* If you have 3 ports, then you will have 3 MAC addresses and 3 IP addresses.
  + EX: Ethernet Port, Router
* Curl doesn’t cache information since it’s not a browser.
* A browse will save data to its cache so when you load the page again it won’t download the information all over again.
* **Packet Switching**
  + Your computer sends the packet to the nearest connection.
    - EX: At home it’s your home router.
  + Each router then has a table, when your packet arrives it searches the table and finds where to send next.
  + If you send ten packets back to back, it may not take the same path, because of congestion.
  + IP doesn’t guarantee delivery. The thing that ensures packet delivery is TCP.
  + Router has no congestion control.
  + 1/3 – ½ of the course is TCP.
* **End-to-End Delay**
  + Two primary parameters you can control
    - Propagation – How long does it take for a packet to get from one place to another assuming a direct line.
      * Speed of light
    - Transmission Rate (R) – The speed your ISP gives you.
    - Transmission Delay – Transmission rate/ how big your packet is.
  + Two parameters you can’t control
    - Queueing Delay – Congestion (Buffer)
    - Processing – (Switching)
  + These four parameters define End-to-End Delay

**1/15/19**

**Queueing Delay – (Congestion)**

* If there is a lot of traffic going through that router, there is probably going to be some queueing delay.

**Processing Delay**

* Some routers are very primitive so there may be a delay in processing time of the packets received.

**Effective Bandwidth**

* The actual bits per second that you achieve with all the delays.
* When you download things, the effective bandwidth changes.
* Diagram:
  + Data and Acknowledgement
  + RTT (s) – Round Trip Time.
    - Changes from time to time.
  + Avg(RTT)
    - The average of a bunch of RTT data.
  + Bandwidth x Avg(RTT) = how many bits we can send through the pipe at one time.
    - TCP wants to ensure that we fill up the pipe every time.
    - A lot of the TCP machinery just goes to figure out this number.

**Bandwidth Delay Product**

* The overall end-to-end delay varies constantly due to hops, congestion, transmission rates, arrival rates, packet loss, etc.
* **Bandwidth \* Avg(RTT)** shows how much you can send through the pipe at one time.
* **Keep Alive** – Keep the connection open for future requests.

**Port Number**

* Multiplexing – When data goes out
* Demultiplexing – When data comes back in

**Summary**

* Most network applications rely on a reliable bidirectional data stream connection.
* HTTP is a ASCII-based protocol for transmitting HTML documents.
* HTTP relies on a reliable bi-directional data stream, which is supported by the transport layer.
* **NAT** created problems for network applications.
* Network applications can be **client/server** or **peer-to-peer** based.
* Applications layer transmits data streams, **end to end**
* Transport layer transmits segments, **end to end**
* Network layer transmits datagrams, **router by router**
* Link layer transmits frames **hop by hop**
* A connection is a 3 way handshake.
  + From initiator
    - SYN
    - SYN+ACK
  + From server
    - ACK

**Unit 2:**

**Python Coding**

* Most of what you do is about sockets.
* Application

**Important Concepts**

* IP Service Model
* UDP/TCP Service Model
* Multiplexing and Demultiplexing
* Client/Server Architecture
* Sockets API
* Python Sockets

**IP Service Model**

* **Datagram** – Individually routed packets.
  + The header has a:
    - IP SA – IP Source Address
    - IP DA – IP Destination Address
  + Hop-by-hop routing – Jumps from hop to hop determined by a table.
* **Unreliable** – Makes no guarantees and is unreliable.
* **Best Effort** – It will only drop datagrams if necessary.
* **Why is the IP service so simple?**
  + Simple, dumb, minimal: Faster, more streamlined and lower cost to build and maintain.
  + **The end-to-end principle**: Where possible, implement features in the end hosts.
  + Allows a variety of reliable (or unreliable) services to be built on top.
  + Works over any link layer: IP makes very few assumptions about the link layer below.
    - So simple, you could technically run IP over carrier pigeons.
* **IP Service Model (Details)**
  + **1.** Tries to prevent packets looping forever.
    - **TTL** – Time to live. There’s a certain number of loops it allows, each time it goes through a loop it decrements by one.
  + **2.** Will fragment packets if they are too long.
    - Most links have a size limit on the packet they can carry so IP will make small packets to fit through that link layer.
  + **3.** IP uses a header checksum to reduce chances of delivering datagram to wrong destination.
  + **4.** Allows for new versions of IP.
    - Currently IPv4 with 32-bit addresses
    - And IPv6 with 128-bit addresses

**1/18/19**

**Port** is associated with application

**UDP (User Datagram Protocol)**

* DNS Lookup
* Also unreliable.
* Connectionless
* Uses port numbers to multiplex/demultiplex.
* **Socket** – (IP Address) + (Port Number Pair)

**Ping** – Check if this site is alive and returns the IP address.

**TCP**

* Reliable end to end.
* TCP only runs on the endpoints at the client and the server.
* **3 Way Handshake**
  + Syn – Host A sends a request to B
    - Indicate it wants to initiate a connection.
    - Sends also the start point of the byte stream.
  + Syn + Ack – B sends back to host A.
    - Indicates it wants to initiate a connection as well and acknowledges the connection from A.
  + Ack
    - A acknowledges the connection with B.
* **Connection Teardown**
  + 4 way
* **Initial Sequence Number**
  + Prevent “stale” connections.
  + A transmission through a connection that prevents sending unauthorized packets through an already open connection.
* **Service Model**

|  |  |
| --- | --- |
| Property | Behavior |
| Stream of Bytes | Reliable byte delivery service. |
| Reliable Delivery | 1. Acknowledgements indicate correct delivery. 2. Checksums detect corrupted data. 3. Sequence numbers detect missing data. 4. Flow-control prevents overrunning receiver. |
| In-sequence | Data delivered to application in sequence transmitted. |
| Congestion Control | Control network congestion |

* + Flow Control – Means you don’t want to overload the buffer.
    - The receiver tells the sender if it can keep sending depending on the size of the buffer.
  + In-sequence – If the packets are sent out of order the TCP will re-sequence them according to the sequence number.
  + Congestion Control – The TCP layer
* TCP header is much more complicated than the IP header because TCP is reliable.
* When you make a connection and you don’t have a port number already allocated the OS will dynamically allocate a port number for you.

**Difference between TCP and UDP**

* TCP is reliable bidirectional byte-stream service.
  + You receive everything as one package no matter how fragmented the whole package is when it is sent.
  + No frame.
  + EX: Send 100 bytes 10 times.
    - TCP reads as 1000 bytes as one package.
* UDP is unreliable datagram.
  + Putting data in an envelope and sending it.
  + Each thing you send is totally self-contained.
  + Frame
  + EX: Send 100 bytes 4 times.
    - Reads as 100 bytes 4 times.

**Sockets**

* **TCP**