CSSE2310 — 9.3

Networks — actual layers

TCP

We will talk about the 5 layer "TCP/IP" stack[1]

[1] Mentioning the 7 layer ISO/OSI stack in your exam means you didn't study.

Physical

Layer 1 is the "Physical" layer.
This is the medium through which signals travel through. eg:

- current in a wire
- infra-red?
- microwave?
- audio?
- pigeons?

(Data)Link

Layer 2 is the (Data-)Link layer.

Peers can communicate directly via messages:

- Unicast or Broadcast?
- Need addressing information.
- Timing or other information?

Example:

- ethernet frames
- wifi

Addresses:

- MAC addresses = “Medium Access Control”[2]
- ethernet MAC addresses are 48bits.

Sends messages via the physical layer.

[2] Controlling access to the medium

Network

Layer 3 is the Network layer.

Exchange messages with any other host on the “internet”.

Uses the internet protocol (IP):

- messages are “IP datagrams”
- IPv4 addresses are 32bits (written as “dotted quads” eg 130.102.72.9)

Sends messages via the link layer.

- Messages may travel through multiple devices before they reach their destination.
- The network layer needs to know “which direction” to send a message to get it to its destination.
- How the network layer works this out is beyond the scope of this course.

Transport layer

Exchange messages with a process on a host on the internet.

Two protocols to choose from:

- UDP = User Datagram Protocol (Datagrams)
- TCP = Transmission Control Protocol (Segments)

Sends messages using the network layer.

Transport addresses

Addresses? "ports" = a 16 bit integer.

- Web → 80
- SSH → 22
- SSL → 443
- Overwatch → 1119, 3724, 6113, 80
- Ports below 1024 are restricted on unix type systems.
- Port 0 doesn't do what you think it does.
- High numbered ports are "ephemeral"[3]

[3] How high depends on the system.

Application layer

"Everything else"

- Web clients (browsers)
- ssh clients (eg putty)
- games?
- SMB (files and printers)

Sends messages using via the transport layer.

Addresses?

- URL/URI?
- . . .

Why so many addresses?

- Application specific addresses : probably boils down to an IP eventually.
- Port : To differentiate between processes on the same computer. (Think phone extension)
- IP : which computer is the process on?
- MAC : which device is this direct message to?

MAC vs IP?

So why do we need IP and MAC?

- The Internet[4] was designed to connect lots of networks together[5]
- …at a time when people had all sorts of different network tech.
- Different hardware was not going to understand addresses from other systems.
- Need a separate "global" address that new software/hardware could process even if the local network didn't understand it.
- ethernet MAC is "supposed to be" globally unique but it isn't heirachical (note how all UQ addresses start with 130.102).

[4] Yes "Internet" and "internet" are different

[5] an internetwork if you will.

Notes

- The layer isolation is not completely enforced.
    - With high priviledges it is possible to directly access some lower layers.
- IP addresses and MAC addresses technically don't identity devices.
    - MAC addresses identify interfaces (the way a device connects to a network)
    - A device could have multiple interfaces:
        - Laptop has an ethernet socket and a wireless interface.
        - A router (node connecting connecting networks) could have many.
    - To be usable on the internet, an interface must have at least one IP address.
        - It may have more

What is the transport layer for?

First we need to talk about two generals.

What is the transport layer for?

The network layer deals with packets. We'd like:

- Streams of bytes
  - or at least not needing to worry about how big a message can be.
- "Reliable delivery"
  - This is actually impossible[6] but the internet will try.
  - Basically, send some packets, if you don't get an acknowledgement send them again.
  - Note: reliable links is not equivalent to reliable end-to-end.

The transport layer's TCP protocol does these things.

[6] Machine powered down?

TCP — connections

TCP is connection oriented.

- In order to communicate with something, you establish a connection first.
- (Not actually a physical circuit)
  - The internet is still packet switched
- Making a connection requires messages to travel there and back.
- Closing a connection is polite
  - There will be a timeout as a fallback.
- Connections are bi-directional

UDP

UDP deals with discrete messages

- With a maximum size
- No guarantee of delivery[7] or acknowledgement[8]

So why would you want this?

[7] The U in UDP does not stand for “unreliable” but it’s not a bad guess

[8] Although one could implement this on top

UDP?

Consider:

- Downloading a file
- Committing to SVN

You don't want part of that going missing.
So not good where you need reliability.

UDP?

What about:

- streaming video
  - One corrupt frame or audio segment is better than stalling the whole thing until a perfect copy gets through.
- Games or interactive environments
  - Synch up every so often
- Congested networks?
  - Some commands getting through in a timely fashion better than being late?
- Many small operations when you don't want to set up a connection for each one.

Note

Remember there is a distinction between *bandwidth* and *latency*.

- Bandwidth = how much information you can send per second. (bits/Time)
- Latency = How long it takes for something to start (Time)