**TCP/IP and UDP**

**TCP/IP is broken into 2 sub-protocols**

* Transmission Control Protocol
* Internet Protocol

Main protocol used to interconnect networks between nodes

TCP/IP uses client/server type model for communication

**TCP**

* Divides data into packets and sends each packet over to the IP protocol for its addition
  + Segmentation
  + Windowing – only so much data can be sent at a time as the other side is only expecting a certain amount
    - Wrong amount sent = rejection, this is why segmentation occurs
* Reassembles data into its original form at the end destination
* Type of data received depends on data being sent
* Each TCP packet is numbered in a sequence unique to that transmission set
  + Knows what’s in what order for that connection
* Time to live before its considered dead
  + If the time to live is expired then it will ask to resend the packet (retransmission)
* Every packet must be acknowledged, this is to
  + If a request contains 500 packets, then the dst needs to receive 500 packets for the request to be complete

**IP (Header)**

* Handles the addressing of the packet with an IP addr and MAC addr
* When a switch looks at a packet, it looks at the IP header
* Keeps track of the following per packet:
  + Source IP
  + Destination IP
  + Time to live

**Port**

Software related docking point for external entities can speak to a specific process

* A web server is a server on a machine that wants to be spoken to through port 80
* When speaking through port 80, the machine will know which service you want to communicate with
* Software can run on any port
  + Although there are a common set of services that run on specific ports

**MAC addr** – Media Access Control

* An address commonly identified by the physical hardware
* This does not change
* Hardcoded into hardware
  + However, VMs (software) can generate a MAC addr too
  + Therefore, software can spoof a MAC addr
* E.g., 1a:00:01:37:f3:40

**3-Way Handshake**

Every TCP to IP connection must complete a 3-way handshake

* Client sends SYN (Synchronised packet) to the end-node to ask if the port is open and if there is availability for a new connection
* Target node will send back a response with a SYN/ACK (synchronise acknowledgement packet)
  + Must have the port open and available to connect
* Client then sends an ACK back to the server and now has an active connection to send data
* Any data sent using the sequence numbers contained in these packets will be referred to in this particular TCP/IP session

**UDP**

**User Datagram Protocol**

* Considered the opposite of TCP/IP
* Doesn’t care if the packet is lost or mangled during transit
  + Can arrive in any order
  + Can arrive in any amount (only 600 packets out of the 800 in the request)
* Down to the application itself to determine the order of packets etc.

**Doesn’t require a 3-Way handshake**

* Just send data and receives data

**Important because port scans with UDP will receive messages back from every port and may look like every port is open when it’s not**

Used a lot for streaming services etc. as losing a couple of packet s don’t matter

Important for scanning as certain services only work over UDP like DNS

* Always scan both TCP and UDP

**Protocols**

**A set of predefined rules that dictates the:**

* Format of data
* Transmission of data
* Reception of data

This is done so than any machine can communicate with each other, regardless of the infrastructure