Today we'll be exploring Computer Networks. There's a lot of details when it comes to Networking, but we'll just cover relevant material here.

A computer goes through several processes and several media to communicate with other computers through the internet. Like the wireshark challenge mentioned, this can briefly be summarised by the OSI model. Each layer is modular and connected to the layer above and below it, so each layer needs to be secure.

Application
Transport
Network
Data Link
Physical

Physical and Data Link explore how information is transferred on and between physical networks. We'll be looking briefly at the Networking Layer and focus more on the Transport and Application Layer. To ensure standardisation across every computer, each layer of the OSI model uses protocols(these define a fixed method on communicating and we'll be hearing a lot about protocols in this document).

The internet consists of *very many* computers and for these computers need to know how to locate on another. Each computer on the Internet is assigned a number called an IP address. This is a 32 bit number in the form:
X.X.X.X
Where **x** can be any number between 0 and 255.

When computers communicate, they do so by sending packets across the internet. Packets can be thought of as self contained units that contain information being sent by computers(amongst other things). The network layer uses the Internet Protocol(IP) to ensure that packets reach the correct destination. These packets use the source and destination IPs to ensure that they reach the correct destination.

Now we know how packets get to their destinations, then what? *We enter the realm of the transport layer*. Getting packets from one destination to another is important, but we have a lot of different things to think about:

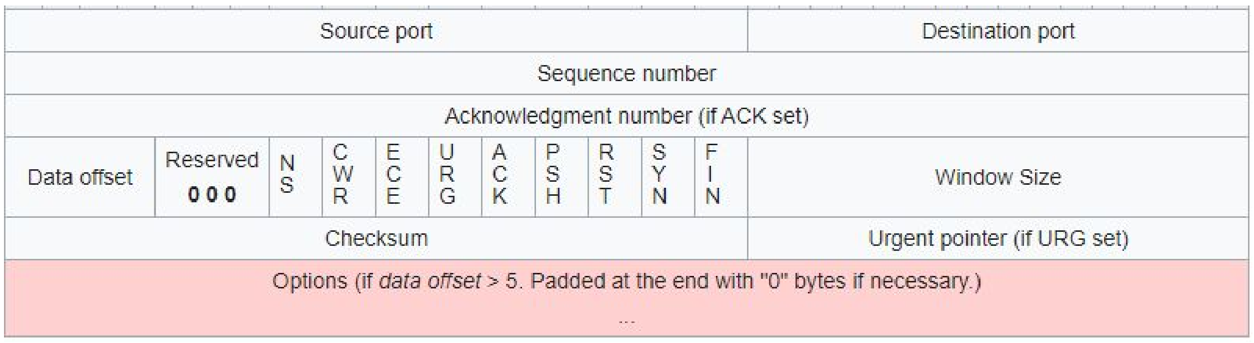
- Reliability - how do we ensure that a packet reliably gets to its destination
- Congestion Control/Flow Control - In the event of too much traffic, how do we ensure that no data is lost or jumbled up
- Connection - how do we keep track of data coming and going from computers
- Multiplexing - applications on computers run on ports(ports are assigned numbers from 0-65535). This is necessary so many applications can run and communicate over the internet at the same time.

The transport layer mostly comprises of 2 protocols and we'll have a brief look at both of them.

TCP - Transmission Control Protocol

TCP is a connection oriented, reliable transmission protocol. It has the following features:

- Reliable - when transferring data across the internet, packets may be dropped due to lost connection. TCP uses acknowledgement to ensure that data is retransmitted even if it is dropped
- Connection Oriented - depending on what data is being sent, the ordering is quite important. TCP uses sequence numbers to keep track of the order in which data is being sent.
- Flow/Congestion Control - TCP uses mechanisms to ensure that there's no congestion when data is being transmitted. Sending too much or too little data can cause reliability issues:
    - Too much data can lead to packet loss which triggers constant retransmission of data(this is quite inefficient)
    - Too little data would mean that less data is sent(which is also quite inefficient)

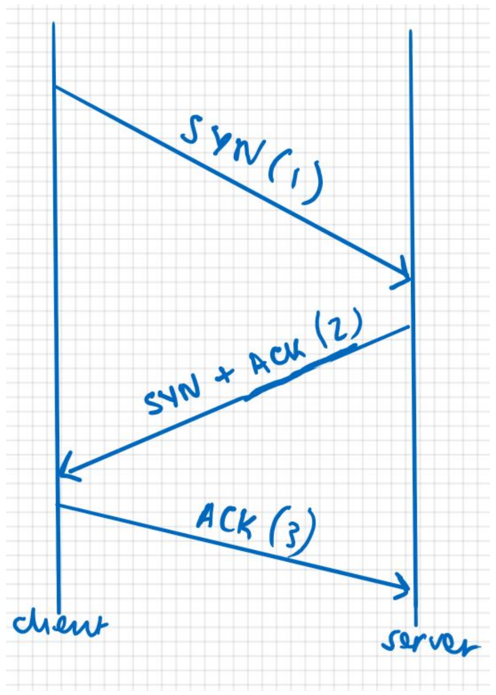


Source port											Destination port
Sequence number											
Acknowledgment number (if ACK set)											
Data offset	Reserved 0 0 0	NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size
Checksum											Urgent pointer (if URG set)
Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...											

This is what a TCP packet looks like. It contains the following data:

- 1st row:
    - source/destination ports(16 bit)- port number to send/receive data
- 2nd row:
    - Sequence number(32 bit) - to keep track of the order of data
- 3rd row:
    - Acknowledgement Number(32 bit) - to keep track of what data has been received
- 4th row:
    - Data offset: Specifies the size of the header so the computer knows what position to start reading off to obtain data
    - Flags: these can be thought of as options for how the protocol works:
        - ACK - indicates that the packet contains an acknowledgement
        - RST - reset the connection
        - SYN - start a connection
        - FIN - end a connection
- 5th row:
    - Checksum - a value that is checked by the receiver to ensure that the header is not corrupted
- 6th row:
    - Data sent by the application

When a computer wants to send data using TCP, it needs to start a connection. It does this using what is called a 3 way handshake:

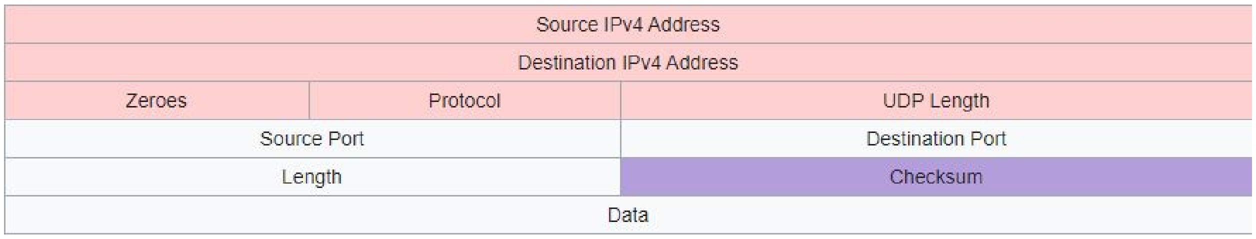


[1] The initiating connection(client) first sends a SYN packet with an initial packet number
[2] The receiving end(server) sends a packet with the SYN and ACK flags set where the acknowledgement number of this packet is the sequence number of the packet sent by the client. The server sets its own sequence number
[3] The client receives this packet and sends a new packet with the ACK flag set and the acknowledgement number set as the initial sequence number sent by the server

After the 3rd packet, the client and server begin transferring data.TCP also has a handshake to tear down a connection but this isn't relevant for now.

UDP - User Datagram Protocol

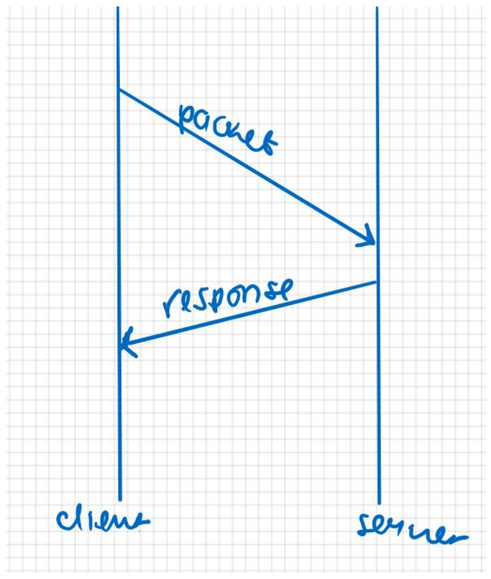
This protocol is a connectionless, stateless protocol. Unlike TCP, it doesn't focus on reliability or creating a connection. This is useful in scenarios where the loss of data is tolerated e.g. streaming video and audio.



Source IPv4 Address		
Destination IPv4 Address		
Zeroes	Protocol	UDP Length
Source Port		Destination Port
Length		Checksum
Data		

This is the format of a UDP packet:

- The first row contains a source address to indicate the source
- The second row contains a destination address to indicate the destination
- The third row:
    - The length contains the length of the UDP header and data
- The fourth row contains source and destination ports
- The 5th row contains:
    - Length of the data
    - Checksum used to check for errors
- The 6th row contains the data transmitted



Unlike TCP, UDP doesn't require a handshake(as it is connectionless).

Application Layer

We've explored how data gets from one computer to another, and why these connections can be reliable(depending on the transport protocol used). Once data actually reaches the computer, it needs to be processed by the computer. We know that computers run different services(act as web servers, act as file storage servers and more). Computers process this data differently depending on what the computer is doing. To standardise this, computers also use protocols on the application layer. Different protocols run on different port numbers, and different services tend to have default port numbers.

TCP Scanning

The most common attack scenario is when an attacker is given the IP address(es) of a machine(s). An attacker would have to enumerate the machine to understand what services are running, and exploit these running services. They do this by scanning ports on the machine they are enumerating. The most common tool used to carry out scans is nmap. Here are the stages you would usually follow when starting out scanning:

1. Start out with a ping scan to see if the host is alive
2. Carry out a TCP Scan
3. Carry out a UDP Scan

Ping Scan
When given an IP address, you want to check if the box is actually up. To do this, you can use what is known as a ping scan. A ping scan uses ICMP(which is a protocol that's part of the

internet protocol), more specifically, an ICMP echo request to check if a machine is up. If the machine is up, it will respond with an ICMP echo reply. You can do this using:

- ping IP-ADDRESS
- nmap -sn ip-address

Something to remember is that some machines will either have ICMP blocked or a firewall that disabled ICMP packets. Even though the machine is alive, it will not respond to this request.

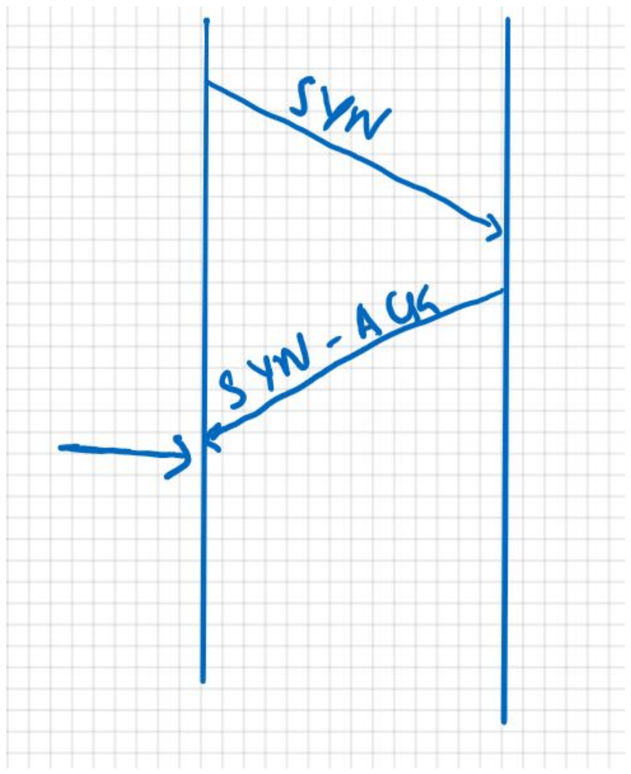
TCP Scan

Now that you've determined a machine is actually up, you can start scanning for services. We'll start doing this with TCP. You can use nmap to scan for TCP services:

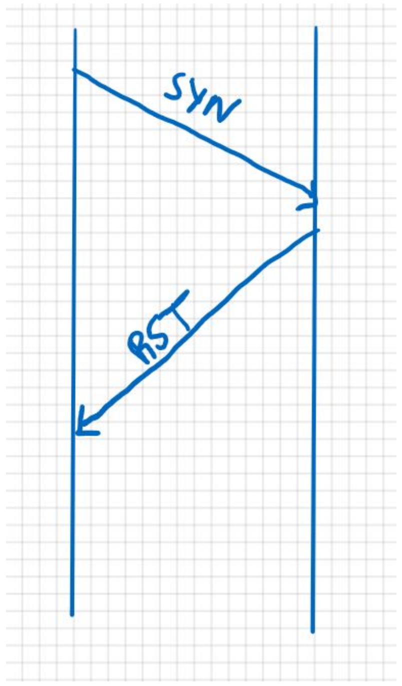
```
nmap -sT -p port-number -O -sC -sV -T[1-5] -oA output-file-name ip-address
```

- -sT is a TCP connect scan(which is the same as the 3 way handshake). It uses this to check if the port is open
  - -sS is a TCP stealth scan which only sends one packet to see if the TCP port is open
- -p port-number is used to specify what port to scan:
  - -p- can also be used to scan all the ports
  - -F is a common flag used to specify the top 250 common ports
- -O is used to determine the host operating system
- -sC is used to run default scripts. NMAP has particular scripts it can run against services running on hosts to gain more information
- -sV is used to determine the versions of services running on open ports
- -T is used to specify the timing with 1 being the slowest and 5 being the fastest. The faster the scan, the more unreliable the results. A good compromise is 3
- -oA is used to store the output in all the formats provided by nmap:
  - .nmap
  - .gnmap
  - .xml

The -sC, -O and -sV flag can all be combined by using the -A flag.



This shows what a stealth scan would be like. Once NMAP receives the SYN-ACK packet from the server, it will assume the server is open.



When a TCP port is closed, it sends an RST packet. Once NMAP receives this packet, it will mark the port as closed.

Another thing to note is that some machines may block ICMP which is used for ping scans. You can still ask nmap to assume the host is alive by adding -Pn flag.

UDP Scanning

UDP scanning uses the same combination as above. Instead of using the -sT or -sS scan type, it uses the -sU flag to specify a UDP scan.

Here's how a UDP scan responds:

- Open port: the UDP port accepts the packet and doesn't respond
- Closed port: the UDP port sends an ICMP port unreachable message

Since UDP doesn't respond if the port is open, UDP ports take a *very long* time to scan.

NMAP displays these particular messages to determine the status of ports:

- Open: the port is open
- Closed: the port is closed
- Filtered: a firewall, filter or something else is blocking the port so nmap cannot tell whether the port is open or closed
    - Nmap uses open and closed in combination with filtered when it cannot tell the status of a port

The NMAP man page gives more information.