**Introduction**

The aim of this room is to provide a beginner's introduction to the basic principles of networking. Networking is a *massive* topic, so this really will just be a brief overview; however, it will hopefully give you some foundational knowledge of the topic, which you can build upon for yourself*.*

The topics that we're going to cover in this room are:

* The OSI Model
* The TCP/IP Model
* How these models look in practice
* An introduction to basic networking tools

***Answer the questions below***

Let's get started!



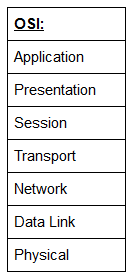
Question Done

Task 2  The OSI Model: An Overview

**The OSI Model: An Overview**

The OSI (**O**pen **S**ystems **I**nterconnection) Model is a standardised model which we use to demonstrate the theory behind computer networking. In practice, it's actually the more compact TCP/IP model that real-world networking is based off; however the OSI model, in many ways, is easier to get an initial understanding from.

The OSI model consists of seven layers:



There are many mnemonics floating around to help you learn the layers of the OSI model -- search around until you find one that you like.

I personally favour: **A**nxious **P**ale **S**hakespeare **T**reated **N**ervous **D**runks **P**atiently

Let's briefly take a look at each of these in turn:

Layer 7 -- Application:

The application layer of the OSI model essentially provides networking options to programs running on a computer. It works almost exclusively with applications, providing an interface for them to use in order to transmit data. When data is given to the application layer, it is passed down into the presentation layer.

Layer 6 -- Presentation:

The presentation layer receives data from the application layer. This data tends to be in a format that the application understands, but it's not necessarily in a standardised format that could be understood by the application layer in the *receiving* computer. The presentation layer translates the data into a standardised format, as well as handling any encryption, compression or other transformations to the data. With this complete, the data is passed down to the session layer.

Layer 5 -- Session:

When the session layer receives the correctly formatted data from the presentation layer, it looks to see if it can set up a connection with the other computer across the network. If it can't then it sends back an error and the process goes no further. If a session *can* be established then it's the job of the session layer to maintain it, as well as co-operate with the session layer of the remote computer in order to synchronise communications. The session layer is particularly important as the session that it creates is unique to the communication in question. This is what allows you to make multiple requests to different endpoints simultaneously without all the data getting mixed up (think about opening two tabs in a web browser at the same time)! When the session layer has successfully logged a connection between the host and remote computer the data is passed down to Layer 4: the transport Layer.

Layer 4 -- Transport:

The transport layer is a very interesting layer that serves numerous important functions. Its first purpose is to choose the protocol over which the data is to be transmitted. The two most common protocols in the transport layer are TCP (**T**ransmission **C**ontrol **P**rotocol) and UDP (**U**ser **D**atagram**P**rotocol); with TCP the transmission is *connection-based* which means that a connection between the computers is established and maintained for the duration of the request. This allows for a reliable transmission, as the connection can be used to ensure that the packets *all* get to the right place. A TCP connection allows the two computers to remain in constant communication to ensure that the data is sent at an acceptable speed, and that any lost data is re-sent. With UDP, the opposite is true; packets of data are essentially thrown at the receiving computer -- if it can't keep up then that's *its*problem (this is why a video transmission over something like Skype can be pixelated if the connection is bad). What this means is that TCP would usually be chosen for situations where accuracy is favoured over speed (e.g. file transfer, or loading a webpage), and UDP would be used in situations where speed is more important (e.g. video streaming).

With a protocol selected, the transport layer then divides the transmission up into bite-sized pieces (over TCP these are called *segments*, over UDP they're called *datagrams*), which makes it easier to transmit the message successfully.

Layer 3 -- Network:

The network layer is responsible for locating the destination of your request. For example, the Internet is a huge network; when you want to request information from a webpage, it's the network layer that takes the IP address for the page and figures out the best route to take. At this stage we're working with what is referred to as *Logical*addressing (i.e. IP addresses) which are still software controlled. Logical addresses are used to provide order to networks, categorising them and allowing us to properly sort them. Currently the most common form of logical addressing is the IPV4 format, which you'll likely already be familiar with (i.e 192.168.1.1 is a common address for a home router).

Layer 2 -- Data Link:

The data link layer focuses on the *physical*addressing of the transmission. It receives a packet from the network layer (that includes the IP address for the remote computer) and adds in the physical (MAC) address of the receiving endpoint. Inside every network enabled computer is a **N**etwork **I**nterface **C**ard (NIC) which comes with a unique MAC (**M**edia **A**ccess **C**ontrol) address to identify it.

MAC addresses are set by the manufacturer and literally burnt into the card; they can't be changed -- although they *can* be spoofed. When information is sent across a network, it's actually the physical address that is used to identify where exactly to send the information.

Additionally, it's also the job of the data link layer to present the data in a format suitable for transmission.

The data link layer also serves an important function when it receives data, as it checks the received information to make sure that it hasn't been corrupted during transmission, which could well happen when the data is transmitted by layer 1: the physical layer.

Layer 1 -- Physical:

The physical layer is right down to the hardware of the computer. This is where the electrical pulses that make up data transfer over a network are sent and received. It's the job of the physical layer to convert the binary data of the transmission into signals and transmit them across the network, as well as receiving incoming signals and converting them back into binary data.

***For the "Which Layer" Questions below, answer using the layer number (1-7)***

***Answer the questions below***

Which layer would choose to send data over TCP or UDP?



Correct Answer

Which layer checks received information to make sure that it hasn't been corrupted?



Correct Answer

In which layer would data be formatted in preparation for transmission?



Correct Answer

Which layer transmits and receives data?



Correct Answer

Which layer encrypts, compresses, or otherwise transforms the initial data to give it a standardised format?



Correct Answer

Which layer tracks communications between the host and receiving computers?



Correct Answer

Which layer accepts communication requests from applications?



Correct Answer

Which layer handles logical addressing?



Correct Answer

When sending data over TCP, what would you call the "bite-sized" pieces of data?



Correct Answer

**[Research]** Which layer would the FTP protocol communicate with?



Correct Answer

 Hint

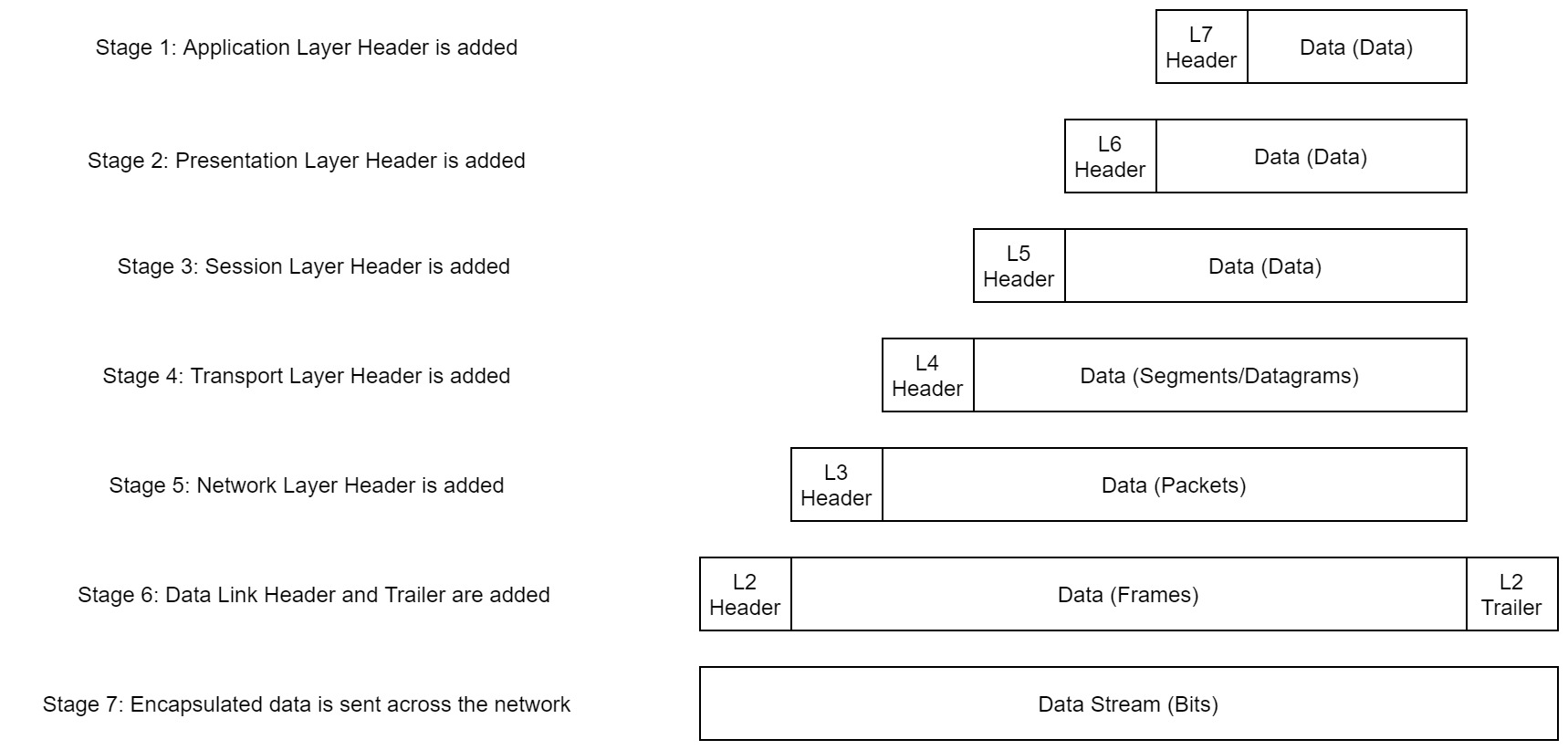
Which transport layer protocol would be best suited to transmit a live video?



Correct Answer

**Encapsulation**

As the data is passed down each layer of the model, more information containing details specific to the layer in question is added on to the start of the transmission. As an example, the header added by the Network Layer would include things like the source and destination IP addresses, and the header added by the Transport Layer would include (amongst other things) information specific to the protocol being used. The data link layer also adds a piece on at the *end* of the transmission, which is used to verify that the data has not been corrupted on transmission; this also has the added bonus of increased security, as the data can't be intercepted and tampered with without breaking the trailer. This whole process is referred to as *encapsulation;*the process by which data can be sent from one computer to another.



Notice that the encapsulated data is given a different name at different steps of the process. In layers 7,6 and 5, the data is simply referred to as data. In the transport layer the encapsulated data is referred to as a segment or a datagram (depending on whether TCP or UDP has been selected as a transmission protocol). At the Network Layer, the data is referred to as a packet. When the packet gets passed down to the Data Link layer it becomes a frame, and by the time it's transmitted across a network the frame has been broken down into bits.

When the message is received by the second computer, it reverses the process -- starting at the physical layer and working up until it reaches the application layer, stripping off the added information as it goes. This is referred to as *de-encapsulation.*As such you can think of the layers of the OSI model as existing inside every computer with network capabilities. Whilst it's not actually as clear cut in practice, computers all follow the same process of encapsulation to send data and de-encapsulation upon receiving it.

The processes of encapsulation and de-encapsulation are very important -- not least because of their practical use, but also because they give us a standardised method for sending data. This means that all transmissions will consistently follow the same methodology, allowing any network enabled device to send a request to any other reachable device and be sure that it will be understood -- regardless of whether they are from the same manufacturer; use the same operating system; or any other factors.

***Answer the questions below***

How would you refer to data at layer 2 of the encapsulation process (with the OSI model)?



Correct Answer

How would you refer to data at layer 4 of the encapsulation process (with the OSI model), if the UDP protocol has been selected?



Correct Answer

What process would a computer perform on a received message?



Correct Answer

Which is the only layer of the OSI model to add a *trailer* during encapsulation?



Correct Answer

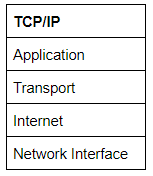
Does encapsulation provide an extra layer of security **(Aye/Nay)**?



Correct Answer

**The TCP/IP Model**

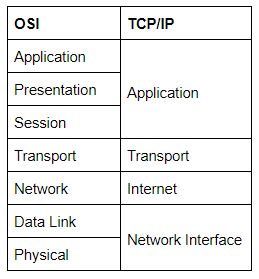
The TCP/IP model is, in many ways, very similar to the OSI model. It's a few years older, and serves as the basis for real-world networking. The TCP/IP model consists of four layers: Application, Transport, Internet and Network Interface. Between them, these cover the same range of functions as the seven layers of the OSI Model.



***Note:****Some recent sources split the TCP/IP model into five layers -- breaking the Network Interface layer into Data Link and Physical layers (as with the OSI model). This is accepted and well-known; however, it is not officially defined (unlike the original four layers which are defined in RFC1122). It's up to you which version you use -- both are generally considered valid.*

You would be justified in asking why we bother with the OSI model if it's not actually used for anything in the real-world. The answer to that question is quite simply that the OSI model (due to being less condensed and more rigid than the TCP/IP model) tends to be easier for learning the initial theory of networking.

The two models match up something like this:



The processes of encapsulation and de-encapsulation work in exactly the same way with the TCP/IP model as they do with the OSI model. At each layer of the TCP/IP model a header is added during encapsulation, and removed during de-encapsulation.

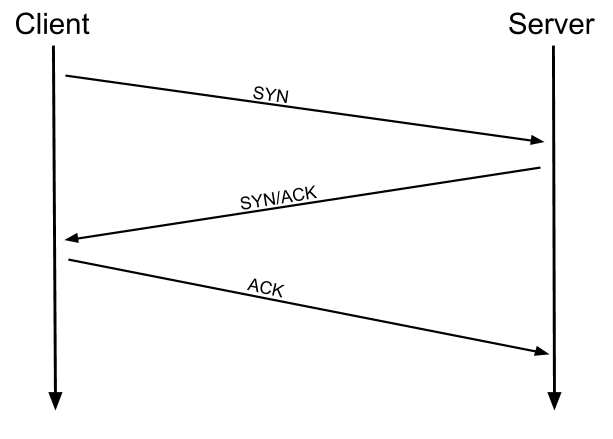
Now let's get down to the practical side of things.

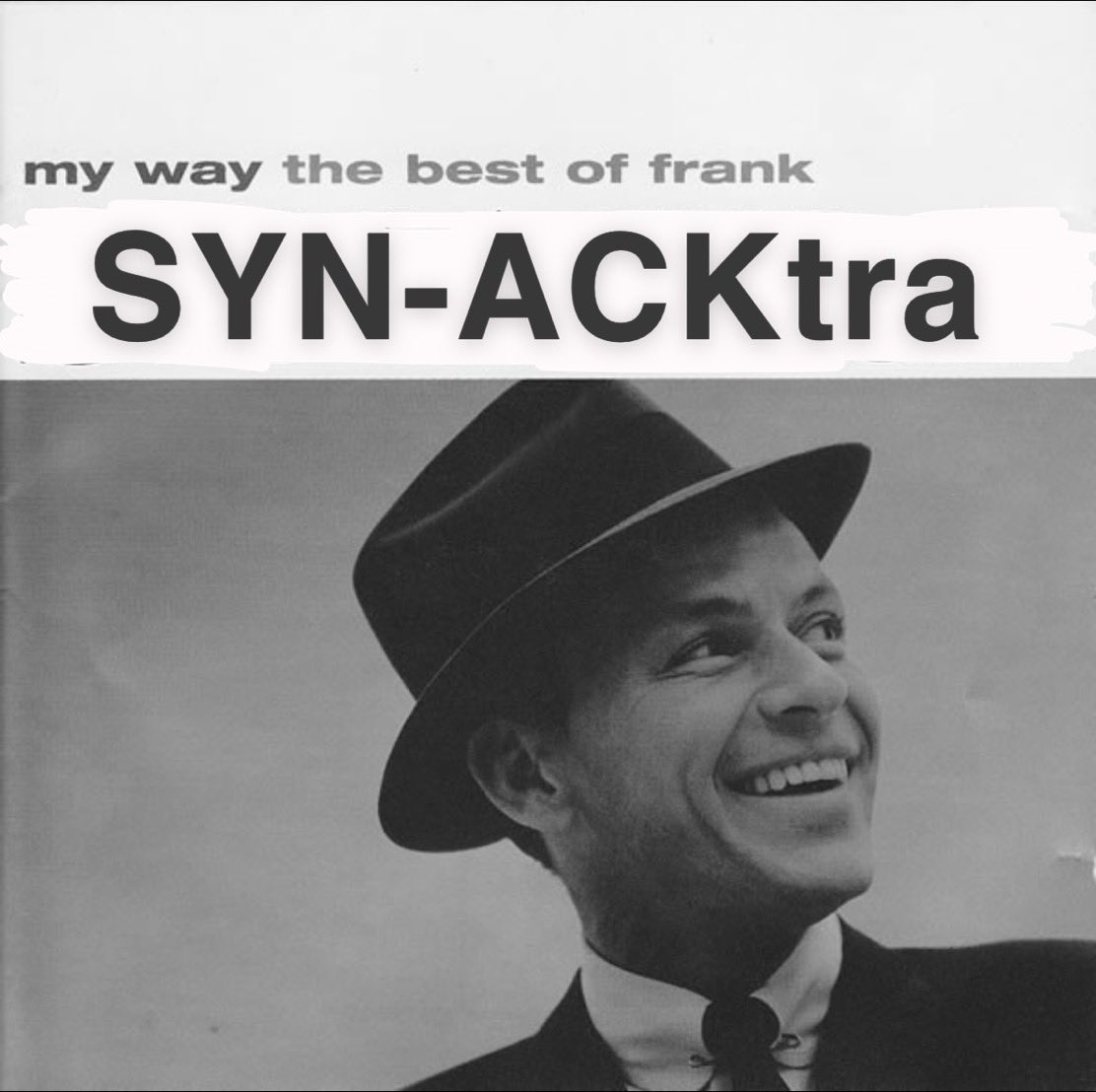
A layered model is great as a visual aid -- it shows us the general process of how data can be encapsulated and sent across a network, but how does it actually happen?

When we talk about TCP/IP, it's all well and good to think about a table with four layers in it, but we're actually talking about a suite of protocols -- sets of rules that define how an action is to be carried out. TCP/IP takes its name from the two most important of these: the **T**ransmission **C**ontrol **P**rotocol (which we touched upon earlier in the OSI model) that controls the flow of data between two endpoints, and the **I**nternet **P**rotocol, which controls how packets are addressed and sent. There are many more protocols that make up the TCP/IP suite; we will cover some of these in later tasks. For now though, let's talk about TCP.

As mentioned earlier, TCP is a *connection-based* protocol. In other words, before you send any data via TCP, you must first form a stable connection between the two computers. The process of forming this connection is called the *three-way handshake*.

When you attempt to make a connection, your computer first sends a special request to the remote server indicating that it wants to initialise a connection. This request contains something called a *SYN* (short for *synchronise*) bit, which essentially makes first contact in starting the connection process. The server will then respond with a packet containing the SYN bit, as well as another "acknowledgement" bit, called *ACK*. Finally, your computer will send a packet that contains the ACK bit by itself, confirming that the connection has been setup successfully. With the three-way handshake successfully completed, data can be reliably transmitted between the two computers. Any data that is lost or corrupted on transmission is re-sent, thus leading to a connection which appears to be lossless.





*(Credit Kieran Smith, Abertay University)*

We're not going to go into exactly *how* this works on a step-to-step level -- not in this room at any rate. It is sufficient to know that the three-way handshake must be carried out before a connection can be established using TCP.

**History:**

It's important to understand exactly *why* the TCP/IP and OSI models were originally created. To begin with there was no standardisation -- different manufacturers followed their own methodologies, and consequently systems made by different manufacturers were completely incompatible when it came to networking. The TCP/IP model was introduced by the American DoD in 1982 to provide a standard -- something for all of the different manufacturers to follow. This sorted out the inconsistency problems. Later the OSI model was also introduced by the International Organisation for Standardisation ([ISO](https://www.iso.org/home.html)); however, it's mainly used as a more comprehensive guide for learning, as the TCP/IP model is still the standard upon which modern networking is based.

***Answer the questions below***

Which model was introduced first, OSI or TCP/IP?



Correct Answer

Which layer of the TCP/IP model covers the functionality of the Transport layer of the OSI model **(Full Name)**?



Correct Answer

Which layer of the TCP/IP model covers the functionality of the Session layer of the OSI model **(Full Name)**?



Correct Answer

The Network Interface layer of the TCP/IP model covers the functionality of two layers in the OSI model. These layers are Data Link, and?.. **(Full Name)**?



Correct Answer

Which layer of the TCP/IP model handles the functionality of the OSI network layer?



Correct Answer

What kind of protocol is TCP?



Correct Answer

 Hint

What is SYN short for?



Correct Answer

 Hint

What is the second step of the three way handshake?



Correct Answer

What is the short name for the "Acknowledgement" segment in the three-way handshake?



Correct Answer

**Networking Tools Ping**

At this stage, hopefully all of the theory has made sense and you now understand the basic models behind computer networking. For the rest of the room we're going to be taking a look at some of the command line networking tools that we can use in practical applications. Many of these tools do work on other operating systems, but for the sake of simplicity, I'm going to assume that you're running Linux for the rest of this room. The first tool that we're going to look at will be the ping command.

The ping command is used when we want to test whether a connection to a remote resource is possible. Usually this will be a website on the internet, but it could also be for a computer on your home network if you want to check if it's configured correctly. Ping works using the ICMP protocol, which is one of the slightly less well-known TCP/IP protocols that were mentioned earlier. The ICMP protocol works on the Network layer of the OSI Model, and thus the Internet layer of the TCP/IP model. The basic syntax for ping is ping <target>. In this example we are using ping to test whether a network connection to Google is possible:

Pinging Google -- it is possible

Notice that the ping command actually returned the IP address for the Google server that it connected to, rather than the URL that was requested. This is a handy secondary application for ping, as it can be used to determine the IP address of the server hosting a website. One of the big advantages of ping is that it's pretty much ubiquitous to any network enabled device. All operating systems support it out of the box, and even most embedded devices can use ping!

Have a go at the following questions. Any questions about syntax can be answered using the man page for ping (man ping on Linux).

***Answer the questions below***

What command would you use to ping the bbc.co.uk website?



Correct Answer

Ping *muirlandoracle.co.uk*  
What is the IPv4 address?



Correct Answer

 Hint

What switch lets you change the interval of sent ping requests?



Correct Answer

 Hint

What switch would allow you to restrict requests to IPv4?



Correct Answer

What switch would give you a more verbose output?



Correct Answer

**Networking Tools Traceroute**