What are Packets and Frames

Packets and frames are small pieces of data that, when forming together, make a larger piece of information or message. However, they are two different things in the OSI model. A frame is at layer 2 - the data link layer, meaning there is no such information as IP addresses. Think of this as putting an envelope within an envelope and sending it away. The first envelope will be the packet that you mail, but once it is opened, the envelope within still exists and contains data (this is a frame).

This process is called encapsulation which we discussed in [room 3: the OSI model](https://tryhackme.com/room/osimodelzi" \t "_blank). At this stage, it's safe to assume that when we are talking about anything IP addresses, we are talking about packets. When the encapsulating information is stripped away, we're talking about the frame itself.

Packets are an efficient way of communicating data across networked devices such as those explained in Task 1. Because this data is exchanged in small pieces, there is less chance of bottlenecking occurring across a network than large messages being sent at once.

For example, when loading an image from a website, this image is not sent to your computer as a whole, but rather small pieces where it is reconstructed on your computer. Take the image below as an illustration of this process. The cat's picture is divided into three packets, where it is reconstructed when it reaches the computer to form the final image.

Packets have different structures that are dependant upon the type of packet that is being sent. As we'll come on to discuss, networking is full of standards and protocols that act as a set of rules for how the packet is handled on a device. With billions of devices connected on the internet, things can quickly break down if there is no standardisation.

Let's continue with our example of the Internet Protocol. A packet using this protocol will have a set of headers that contain additional pieces of information to the data that is being sent across a network.

Some notable headers include:

|  |  |
| --- | --- |
| **Header** | **Description** |
| Time to Live | This field sets an expiry timer for the packet to not clog up your network if it never manages to reach a host or escape! |
| Checksum | This field provides integrity checking for protocols such as TCP/IP. If any data is changed, this value will be different from what was expected and therefore corrupt. |
| Source Address | The IP address of the device that the packet is being sent **from**so that data knows where to **return to**. |
| Destination Address | The device's IP address the packet is being sent to so that data knows where to travel next. |

Answer the questions below

What is the name for a piece of data when it **does have** IP addressing information?

Principio del formulario

SubmitHint

Final del formulario

What is the name for a piece of data when it **does not have**IP addressing information?

**TCP**(or **T**ransmission **C**ontrol **P**rotocol for short) is another one of these rules used in networking.

This protocol is very similar to the OSI model that we have previously discussed in room three of this module so far. The TCP/IP protocol consists of four layers and is arguably just a summarised version of the OSI model. These layers are:

* Application
* Transport
* Internet
* Network Interface

Very similar to how the OSI model works, information is added to each layer of the TCP model as the piece of data (or packet) traverses it. As you may recall, this process is known as encapsulation - where the reverse of this process is decapsulation.

One defining feature of TCP is that it is **connection-based**, which means that TCP must establish a connection between both a client and a device acting as a server **before** data is sent.

Because of this, TCP guarantees that any data sent will be received on the other end. This process is named the Three-way handshake, which is something we'll come on to discuss shortly. A table comparing the advantages and disadvantages of TCP is located below:

|  |  |
| --- | --- |
| **Advantages of TCP** | **Disadvantages of TCP** |
| Guarantees the integrity of data. | Requires a reliable connection between the two devices. If one small chunk of data is not received, then the entire chunk of data cannot be used and must be re-sent. |
| Capable of synchronising two devices to prevent each other from being flooded with data in the wrong order. | A slow connection can bottleneck another device as the connection will be reserved on the other device the whole time. |
| Performs a lot more processes for reliability | TCP is significantly slower than UDP because more work (computing) has to be done by the devices using this protocol. |

TCP packets contain various sections of information known as headers that are added from encapsulation. Let's explain some of the crucial headers in the table below:

|  |  |
| --- | --- |
| **Header** | **Description** |
| Source Port | This value is the port opened by the sender to send the TCP packet from. This value is chosen randomly (out of the ports from 0-65535 that aren't already in use at the time). |
| Destination Port | This value is the port number that an application or service is running on the remote host (the one receiving data); for example, a webserver running on port 80. Unlike the source port, this value is not chosen at random. |
| Source IP | This is the IP address of the device that is sending the packet. |
| Destination IP | This is the IP address of the device that the packet is destined for. |
| Sequence Number | When a connection occurs, the first piece of data transmitted is given a random number. We'll explain this more in-depth further on. |
| Acknowledgement Number | After a piece of data has been given a sequence number, the number for the next piece of data will have the sequence number + 1. We'll also explain this more in-depth further on. |
| Checksum | This value is what gives TCP integrity. A mathematical calculation is made where the output is remembered. When the receiving device performs the mathematical calculation, the data must be corrupt if the output is different from what was sent. |
| Data | This header is where the data, i.e. bytes of a file that is being transmitted, is stored. |
| Flag | This header determines how the packet should be handled by either device during the handshake process. Specific flags will determine specific behaviours, which is what we'll come on to explain below. |

Next, we'll come on to discuss the *Three-way handshake -*the term given for the process used to establish a connection between two devices. The Three-way handshake communicates using a few special messages - the table below highlights the main ones:

|  |  |  |
| --- | --- | --- |
| **Step** | **Message** | **Description** |
| 1 | SYN | A SYN message is the initial packet sent by a client during the handshake. This packet is used to initiate a connection and synchronise the two devices together (we'll explain this further later on). |
| 2 | SYN/ACK | This packet is sent by the receiving device (server) to acknowledge the synchronisation attempt from the client. |
| 3 | ACK | The acknowledgement packet can be used by either the client or server to acknowledge that a series of messages/packets have been successfully received. |
| 4 | DATA | Once a connection has been established, data (such as bytes of a file) is sent via the "DATA" message. |
| 5 | FIN | This packet is used to *cleanly (properly)* close the connection after it has been complete. |
| # | RST | This packet abruptly ends all communication. This is the last resort and indicates there was some problem during the process. For example, if the service or application is not working correctly, or the system has faults such as low resources. |

The diagram below shows a normal Three-way handshake process between Alice and Bob. In real life, this would be between two devices.

Any sent data is given a random number sequence and is reconstructed using this number sequence and incrementing by 1. Both computers must agree on the same number sequence for data to be sent in the correct order. This order is agreed upon during three steps:

1. SYN - Client: Here's my Initial Sequence Number(ISN) to **SYN**chronise with (0)
2. SYN/ACK - Server: Here's my Initial Sequence Number (ISN) to **SYN**chronise with (5,000), and I **ACK**nowledge your initial number sequence (0)
3. ACK - Client: I **ACK**nowledge your Initial Sequence Number (ISN) of (5,000), here is some data that is my ISN+1 (0 + 1)

|  |  |  |
| --- | --- | --- |
| **Device** | **Initial Number Sequence (ISN)** | **Final Number Sequence** |
| Client (Sender) | 0 | 0 + 1 = 1 |
| Client (Sender) | 1 | 1 + 1 = 2 |
| Client (Sender) | 2 | 2 + 1 = 3 |

**TCP Closing a Connection:**

Let's quickly explain the process behind TCP closing a connection. First, TCP will close a connection once a device has determined that the other device has successfully received all of the data.

Because TCP reserves system resources on a device, it is best practice to close TCP connections as soon as possible.

To initiate the closure of a TCP connection, the device will send a "FIN" packet to the other device. Of course, with TCP, the other device will also have to acknowledge this packet.

Let's show this process using Alice and Bob as we have previously.

In the illustration, we can see that Alice has sent Bob a "**FIN**" packet. Because Bob received this, he will let Alice know that he received it and that he also wants to close the connection (using FIN). Alice has heard Bob loud and clear and will let Bob know that she acknowledges this.

Answer the questions below

What is the header in a TCP packet that ensures the integrity of data?

Principio del formulario

Submit

Final del formulario

Provide the order of a normal Three-way handshake (with each step separated by a comma)

Principio del formulario

Final del formulario

Principio del formulario

Final del formulario

UDP/IP

The **U**ser **D**atagram **P**rotocol (**UDP**) is another protocol that is used to communicate data between devices.

Unlike its brother TCP, UDP is a **stateless**protocol that doesn't require a constant connection between the two devices for data to be sent. For example, the Three-way handshake does not occur, nor is there any synchronisation between the two devices.

Recall some of the comparisons made about these two protocols in Room 3: "OSI Model". Namely, UDP is used in situations where applications can tolerate data being lost (such as video streaming or voice chat) or in scenarios where an unstable connection is not the end-all. A table comparing the advantages and disadvantages of UDP is located below:

|  |  |
| --- | --- |
| **Advantages of UDP** | **Disadvantages of UDP** |
| UDP is much faster than TCP. | UDP doesn't care if the data is received or not. |
| UDP leaves the application (user software) to decide if there is any control over how quickly packets are sent. | It is quite flexible to software developers in this sense. |
| UDP does not reserve a continuous connection on a device as TCP does. | This means that unstable connections result in a terrible experience for the user. |

As mentioned, no process takes place in setting up a connection between two devices. Meaning that there is no regard for whether or not data is received, and there are no safeguards such as those offered by TCP, such as data integrity.

UDP packets are much simpler than TCP packets and have fewer headers. However, both protocols share some standard headers, which are what is annotated in the table below:

|  |  |
| --- | --- |
| **Header** | **Description** |
| Time to Live (TTL) | This field sets an expiry timer for the packet, so it doesn't clog up your network if it never manages to reach a host or escape! |
| Source Address | The IP address of the device that the packet is being sent from, so that data knows where to return to. |
| Destination Address | The device's IP address the packet is being sent to so that data knows where to travel next. |
| Source Port | This value is the port that is opened by the sender to send the UDP packet from. This value is randomly chosen (out of the ports from 0-65535 that aren't already in use at the time). |
| Destination Port | This value is the port number that an application or service is running on the remote host (the one receiving the data); for example, a webserver running on port 80. Unlike the source port, this value is not chosen at random. |
| Data | This header is where data, i.e. bytes of a file that is being transmitted, is stored. |

Next, we'll come on to discuss how the process of a connection via UDP differs from that of something such as TCP.  We should recall that UDP is **stateless**. No acknowledgement is sent during a connection.

The diagram below shows a normal UDP connection between Alice and Bob. In real life, this would be between two devices.

Answer the questions below

What does the term "UDP" stand for?

Principio del formulario

Submit

Final del formulario

What type of connection is "UDP"?

Principio del formulario

Submit

Final del formulario

What protocol would you use to transfer a file?

Principio del formulario

Submit

Final del formulario

What protocol would you use to have a video call?

Ports 101 (Practical)

View Site

Perhaps aptly titled by their name, ports are an essential point in which data can be exchanged. Think of a harbour and port. Ships wishing to dock at the harbour will have to go to a port compatible with the dimensions and the facilities located on the ship. When the ship lines up, it will connect to a **port** at the harbour. Take, for instance, that a cruise liner cannot dock at a port made for a fishing vessel and vice versa.

These ports enforce what can park and where — if it isn't compatible, it cannot park here. Networking devices also use ports to enforce strict rules when communicating with one another. When a connection has been established (recalling from the OSI model's room), any data sent or received by a device will be sent through these ports. In computing, ports are a numerical value between **0** and**65535** (65,535).

Because ports can range from anywhere between 0-65535, there quickly runs the risk of losing track of what application is using what port. A busy harbour is chaos! Thankfully, we associate applications, software and behaviours with a standard set of rules. For example, by enforcing that any web browser data is sent over port 80, software developers can design a web browser such as Google Chrome or Firefox to interpret the data the same way as one another.

This means that all web browsers now share one common rule: data is sent over port 80. How the browsers look, feel and easy to use is up to the designer or the user's decision.

While the standard rule for web data is *port 80*, a few other protocols have been allocated a standard rule. Any port that is within **0**and **1024** (1,024) is known as a common port. Let's explore some of these other protocols below:

|  |  |  |
| --- | --- | --- |
| **Protocol** | **Port Number** | **Description** |
| **F**ile **T**ransfer **P**rotocol (**FTP**) | 21 | This protocol is used by a file-sharing application built on a client-server model, meaning you can download files from a central location. |
| **S**ecure **Sh**ell (**SSH**) | 22 | This protocol is used to securely login to systems via a text-based interface for management. |
| **H**yper**T**ext Transfer Protocol (**HTTP**) | 80 | This protocol powers the World Wide Web (WWW)! Your browser uses this to download text, images and videos of web pages. |
| **H**yper**T**ext **T**ransfer **P**rotocol **S**ecure (**HTTPS**) | 443 | This protocol does the exact same as above; however, securely using encryption. |
| **S**erver**M**essage**B**lock (**SMB**) | 445 | This protocol is similar to the File Transfer Protocol (FTP); however, as well as files, SMBallows you to share devices like printers. |
| **R**emote **D**esktop **P**rotocol (**RDP**) | 3389 | This protocol is a secure means of logging in to a system using a visual desktop interface (as opposed to the text-based limitations of the SSH protocol). |

We have only briefly covered the more common protocols in cybersecurity. You can [find a table of the 1024 common ports listed](http://www.vmaxx.net/techinfo/ports.htm" \t "_blank) for more information.

What is worth noting here is that these protocols only follow the standards. I.e. you can administer applications that interact with these protocols on a different port other than what is the standard (running a web server on 8080 instead of the 80 standard port). Note, however, applications will presume that the standard is being followed, so you will have to provide a **colon (:)** along with the port number.

**Practical Challenge:**

Open the site attached to this task and connect to the IP address "**8.8.8.8**" on port "**1234**", and you'll receive a flag.

Answer the questions below

What is the flag received from the challenge?

Principio del formulario

Principio del formulario

Final del formulario