TEMA 2: Programación de comunicaciones en red

unir FORMACIÓN PROFESIONAL PROEDUCA

1. Comprender qué es la programación distribuida.
2. Distinguir los elementos que forman parte de cualquier proceso de comunicación entre dos programas conectados a la red.
3. Distinguir entre los protocolos IP, TCP, UDP.
4. Comprender el funcionamiento de la arquitectura de software Cliente-Servidor.

Programación Distribuida

La programación distribuida hace referencia a las aplicaciones que tienen sus programas distribuidos en distintos equipos conectados a través de la red. Los diferentes programas colaboran entre sí para alcanzar un objetivo común.

En un sistema distribuido, podría incluso prestarse el mismo servicio en distintos equipos garantizando así la disponibilidad si alguno de los equipos estuviera caído.

La programación distribuida, para que resulte eficiente, requiere del seguimiento de algún tipo de **Arquitectura de Software**.

Arquitectura de software

Una **arquitectura de software define un conjunto de normas y patrones** que proporcionan un estándar que sirve como guía para que los analistas y programadores que han de hacerse cargo de una aplicación sigan una línea común para cumplir los objetivos fijados de la forman más profesional posible.

Los modelos de arquitectura de software más extendidos hoy en día son los **modelos multicapa** que ofrecen una separación entre diferentes niveles de la aplicación. Cada capa tendrá unas tareas bien diferenciadas.

En esta unidad estudiaremos el modelo **cliente-servidor**.

Arquitectura cliente-servidor

En la arquitectura cliente-servidor, las tareas se reparten entre estos dos roles:

1. **El servidor:** como proveedor o suministrador de recursos.
2. **El cliente:** como consumidor de los recursos que provee el servidor.

Funciones del servidor

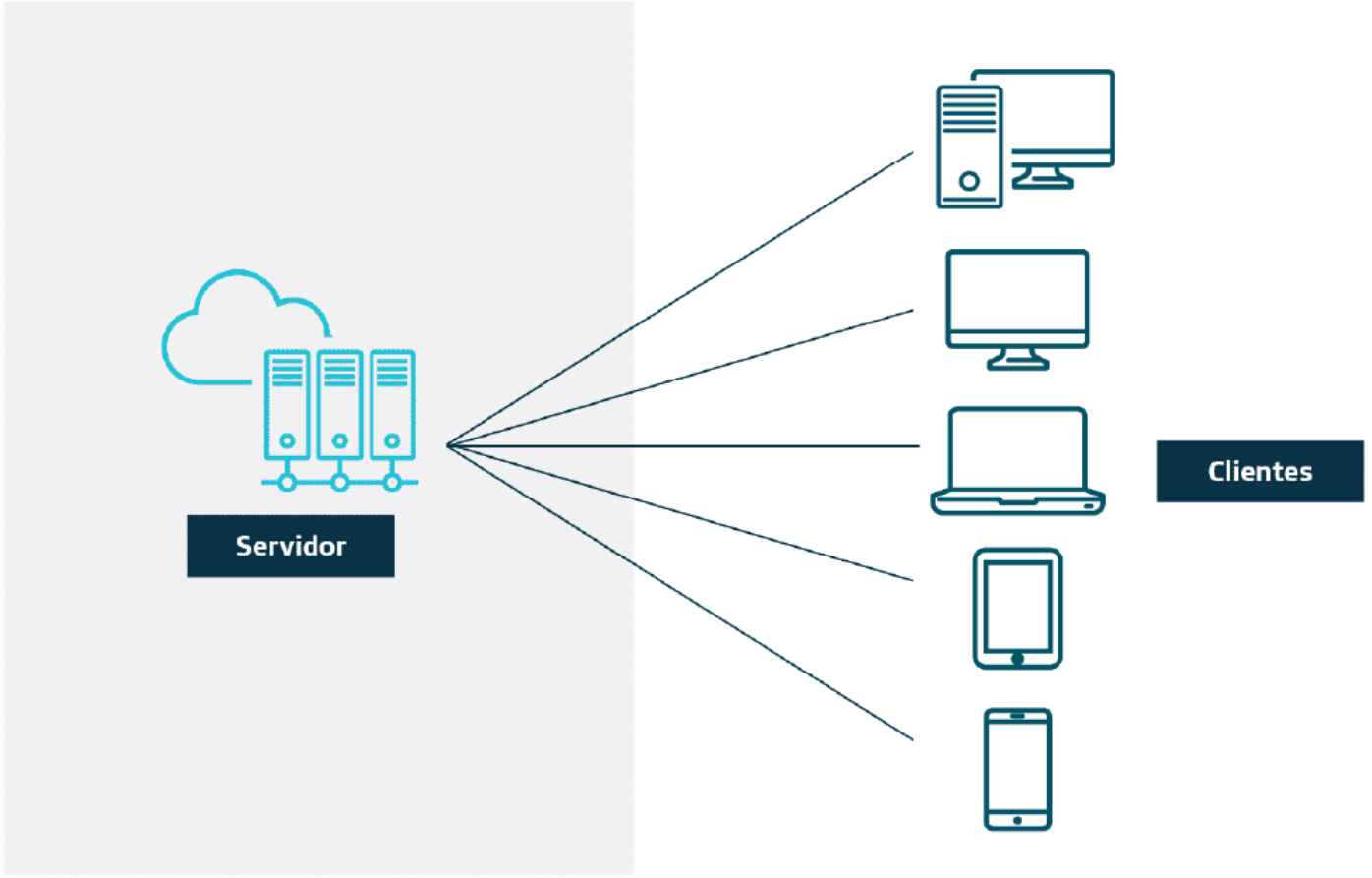
- Aceptar las peticiones de los clientes.
- Analizar la petición del cliente e interactuar con el resto de capas del sistema (capa de datos, capa lógica) para poder responder a la petición.
- Generar la respuesta para el cliente.
- Enviar la respuesta al cliente.

Funciones del cliente

- Suele Interactuar con el usuario.
- Realiza peticiones al servidor.
- Recibe los resultados del servidor.
- Presenta los resultados al usuario o a otros procesos que lo requieran.

Características de la arquitectura Cliente-Servidor:

- Los clientes inician una comunicación con el servidor a través de un protocolo y el servidor responde.
- El servidor presta los recursos y servicios al cliente que los consume.
- Existe una **transparencia física entre cliente y servidor**, ya que el cliente no tienen por qué saber dónde se encuentran los recursos que consume.
- Se utilizan tecnologías que permitan **independencia de la plataforma**, tanto del cliente como del servidor.
- Existe **encapsulamiento** de los servicios prestados por el servidor.
- El sistema debe garantizar la **integridad de los datos transmitidos**, evitando pérdidas de información e inconsistencia en los datos que finalmente sean transmitidos.
- Los sistemas deben ser **escalables** para garantizar la posibilidad de crecimiento y la adaptación a cambios imprevistos.

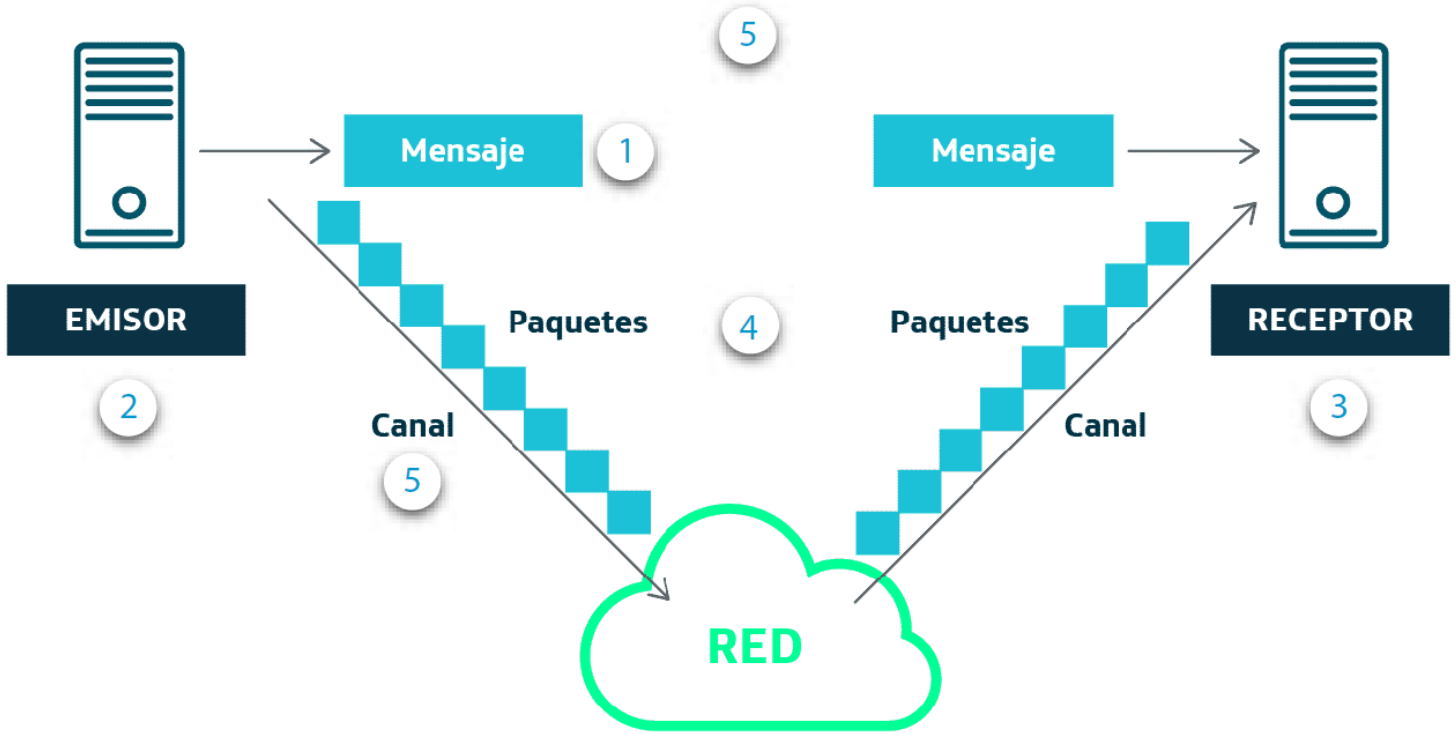


Esquema de un sistema Cliente-Servidor. El servidor provee de recursos a distintos tipos de clientes.

Comunicación entre aplicaciones

En un sistema informático distribuido, los programas que lo forman se comunican a través de la red. En este apartado aprenderás cómo se llevan a cabo estas comunicaciones.

Todo proceso de comunicación consta de los siguientes elementos:



Proceso de comunicación entre dos programas conectados a la red.

1. **Mensaje.** Información que se debe enviar desde un emisor a un receptor.

2. **Emisor.** Programa que envía el mensaje.

3. **Receptor.** Programa que recibe el mensaje.

4. **Paquete.** Unidad básica de información. Un mensaje debe dividirse en paquetes para poder enviarse.

5. **Canal de comunicación.** Por donde se transmite el conjunto de paquetes que componen un mensaje.

6. **Protocolo de comunicaciones.** Conjunto de instrucciones, normas o reglas que guían la acción de comunicar algo entre dos equipos distintos conectados a través de la red.

Protocolos de comunicaciones

Recuerda: un protocolo de comunicaciones es un conjunto de instrucciones, normas o reglas que guían la acción de comunicar algo entre dos equipos distintos conectados a través de la red.

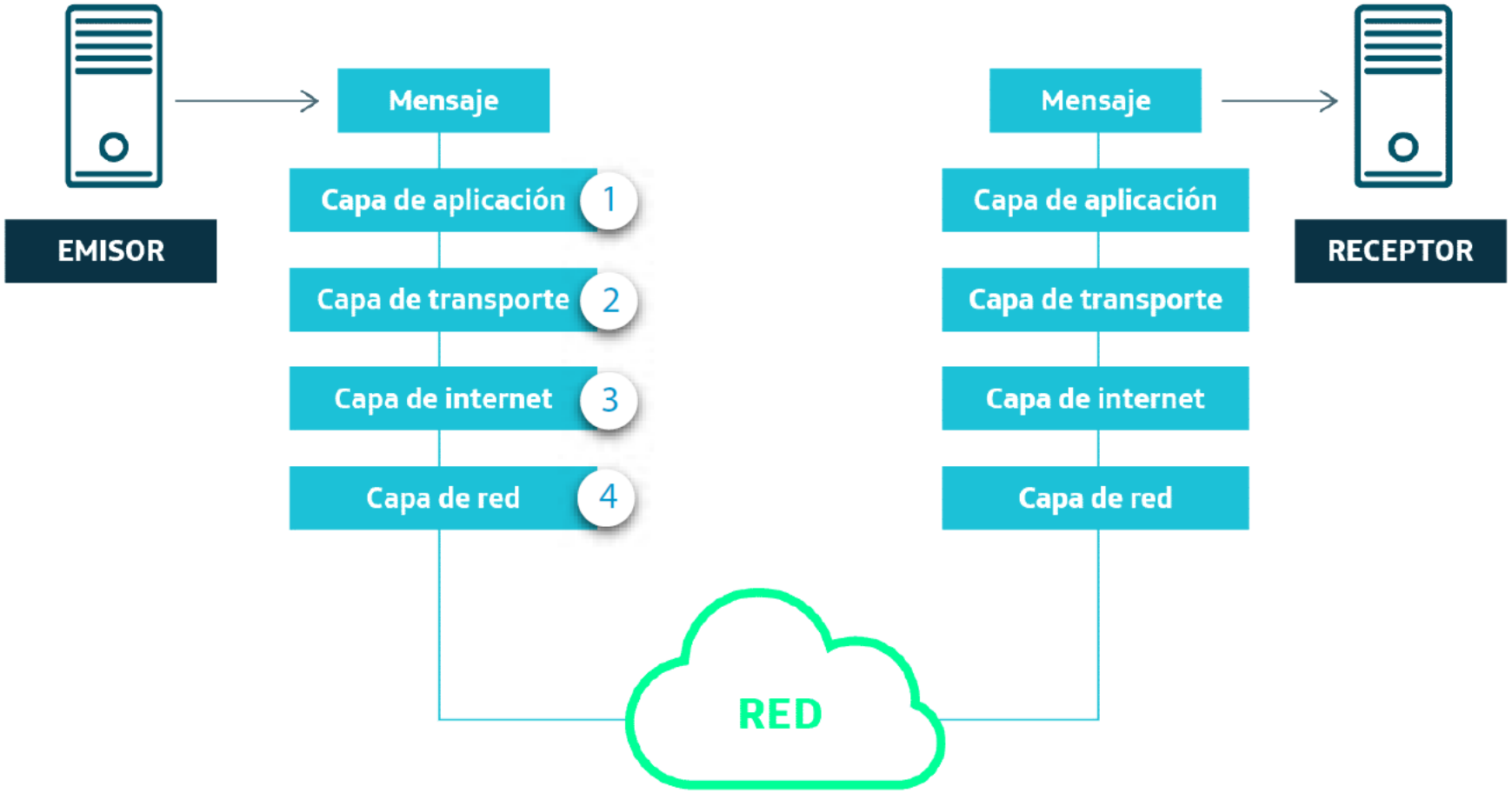
En este apartado verás cómo funcionan los protocolos de comunicación IP, TCP y UDP.

El protocolo IP

El protocolo IP (*Internet Protocol*) es el más extendido para el establecimiento de la comunicación entre los distintos programas de un Sistema distribuido. Se trata del estándar para el establecimiento de comunicaciones en Internet y define una pila de protocolos organizados en capas. Las capas inferiores prestan servicio a las capas superiores.

Pila de protocolos IP

La pila de protocolos IP realizan su tarea en la emisión y recepción del mensaje.


EMISOR
Mensaje
Capa de aplicación
1
Capa de transporte
2
Capa de internet
3
Capa de red
4
RED
Mensaje
Capa de aplicación
Capa de transporte
Capa de internet
Capa de red
RECEPTOR

1. Capa de aplicación.

Se trata de la capa de nivel superior formada por el conjunto de programas que componen el sistema distribuido. Estos programas se sirven de las capas inferiores para llevar a cabo sus comunicaciones.

2. Capa de transporte.

Compuesta por los elementos software encargados de crear un canal de comunicación, descomponer el mensaje en paquetes y llevar a cabo la transmisión entre emisor y receptor. Estas tareas mencionadas pueden ser llevadas a cabo por los protocolos TCP o UDP.
Es habitual hablar de TCP-IP como un protocolo único, aunque para ser más exactos, se trata de dos protocolos distintos. TCP se sitúa en la capa de transporte del protocolo IP.

3. Capa de internet.

Elementos software que prestan servicio a la capa de transporte dirigiendo los paquetes a través de la red y velando por que lleguen a su destino.

4. Capa de red.

Es la capa de más bajo nivel y está compuesta por los elementos hardware que permiten la comunicación entre los distintos equipos de la red: cables, enrutadores, etc.

El protocolo TCP

Se sitúa en la capa de transporte dentro de la pila de protocolos IP.

Características

- Garantiza que los datos lleguen íntegros a su destino siempre y cuando las capas inferiores en la pila de protocolos hayan realizado correctamente su trabajo.
- Garantiza que los mensajes llagarán en el mismo orden en que fueron enviados.
- Se trata de un protocolo orientado a conexión, lo que significa que el canal de comunicación que se establecen se mantendrá abierto permitiendo el envío de más de un mensaje mientras el emisor o el receptor decidan cerrarlo.

Protocolo UDP

Se trata de otro protocolo situado en la capa de transporte dentro de la pila de protocolos IP, aunque no está tan extendido como el protocolo TCP.

Características

- No es un protocolo orientado a conexión, lo que implica que es mucho más rápido pero mucho menos seguro.
- No se puede garantizar al 100% que el mensaje llegará a su destino.
- No se puede garantizar que los mensajes lleguen en el mismo orden en que se enviaron.
- Los mensajes enviados utilizando el protocolo UDP se denominan "datagramas".
- El tamaño máximo de cada mensaje es de 64 kb.

- La **programación distribuida** hace referencia a las aplicaciones que tienen sus programas distribuidos en distintos equipos conectados a través de la red. Los diferentes programas colaboran entre sí para alcanzar un objetivo común.

- En la **arquitectura Cliente-Servidor**, las responsabilidades se reparten entre los siguientes roles: un servidor, como proveedor de recursos o servicios, y cliente, como consumidor de los recursos y servicios que presta el proveedor.

- Todo **proceso de comunicación** consta de los siguientes elementos: protocolo de comunicaciones, emisor, receptor, mensaje, paquete y canal de comunicación.

- Un **protocolo de comunicaciones** es un conjunto de instrucciones, normas o reglas que guían la acción de comunicar algo entre dos equipos distintos conectados a través de la red. El mas extendido es el protocolo **IP** que define una pila de protocolos organizados en capas (capa de aplicación, capa de transporte, capa de internet, capa de red). En la capa de transporte pueden situarse los protocolos **TCP** o **UDP**.

https://kahoot.it/solo/?quizId=90db4c0d-3273-4f51-bf42-c160d4e3040d

TEMA 2: Programación de comunicaciones en red

unir FORMACIÓN PROFESIONAL PROEDUCA

En esta lección perseguimos los siguientes objetivos:

1. Conocer el funcionamiento de los *Sockets*.
2. Distinguir entre *sockets stream* y *socket datagram*.
3. Crear programas java que utilicen el modelo cliente-servidor implementado mediante *Sockets*.
4. Crear programas java que se comuniquen utilizando los *sockets* de tipo *datagram*.

¿Qué es un socket?

Un **socket** es como un **enchufe** (en sentido literal) que conecta dos dispositivos para que puedan **comunicarse** entre sí a través de una red, como si estuvieras haciendo una llamada de teléfono.

Ejemplo con dos móviles:

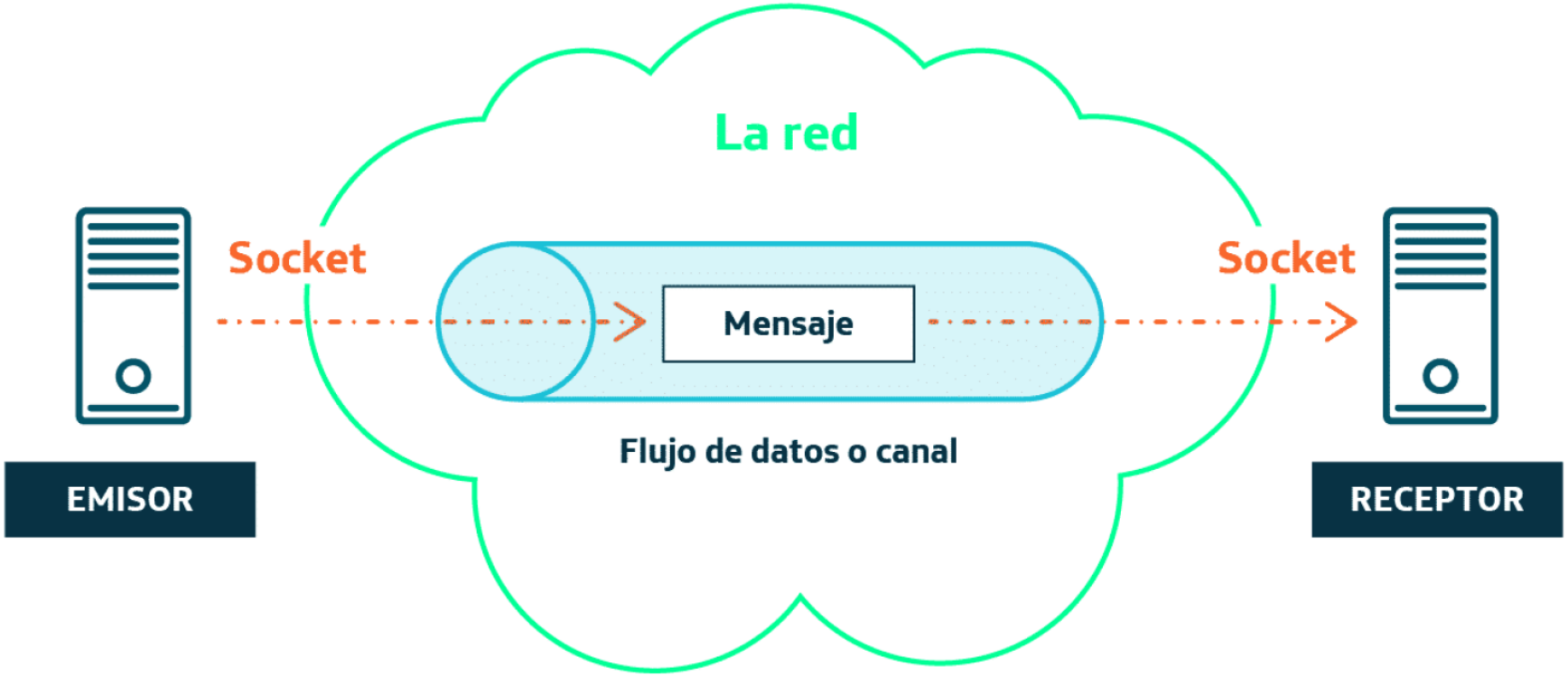
1.Tú y **un amigo** quereis chatear usando una aplicación de mensajería, como WhatsApp.
2.Ambos teneis la aplicación abierta (esto serían los **programas** que quieren comunicarse).
3.Cada teléfono tiene su **número de teléfono** y está conectado a Internet (esto sería la **dirección IP y el puerto**, que identifica los dispositivos en la red).
4.Para poder enviarse mensajes, los dos teléfonos necesitan estar **conectados a la red**. Aquí es donde entra el **socket**.
En este caso, el **socket** es como una especie de canal que se abre entre los dos teléfonos, permitiendo que los mensajes viajen de un móvil al otro a través de Internet. Así, cuando tú envías un mensaje, viaja por este "enchufe virtual" y llega al móvil de tu amigo, que luego lo muestra en su pantalla.

¿Qué es un socket?

Los *sockets* proveen de un mecanismo para la comunicación entre dos procesos que pueden encontrarse en distinta máquina.

Ofrecen una abstracción de la pila de protocolos IP que permite al programador olvidarse de los detalles técnicos de más bajo nivel en las comunicaciones a través de la red.

La traducción literal de ***socket*** es "enchufe" y representa un extremo de un canal o flujo de datos que comunicará dos procesos.



El uso de sockets es fundamental en los sistemas distribuidos donde una aplicación está compuesta por varios programas situados en diferentes equipos conectados a través de la red que colaboran y se comunican entre ellos para conseguir un objetivo común.

Existen dos tipos de *sockets*:

sockets stream

Están orientados a conexión, ya que en la capa de transporte de la pila de protocolos IP utilizan el protocolo TCP , lo que los hace más fiables al gozar de las características de este protocolo; **se garantiza que los mensajes llegan a su destino y en el orden correcto.**
Se utilizan en la implementación de aplicaciones cliente / servidor.

sockets datagram

No están orientados a conexión, ya que en la capa de transporte de la pila de protocolos IP utilizan el protocolo UDP, lo que **les resta fiabilidad:** no se garantiza al 100% que los mensajes lleguen a su destino y tampoco se garantiza que lleguen en el orden correcto.
En la comunicación mediante *sockets datagram* no existe un rol de cliente y otro rol de servidor.

Dirección IP y puerto

Para que dos programas puedan comunicarse a través de la red, necesitan de algún sistema para localizarse el uno al otro.

El sistema de localización es a partir de la dirección IP y el número de puerto.

Una dirección IP es un número que identifica de manera única a un equipo dentro de una red.

Por otro lado, una máquina podría tener varios procesos en ejecución utilizando *sockets* y es necesario identificarlos de alguna manera; por ese motivo existe lo que se denomina el número de puerto. Cada número de puerto identifica un *socket* dentro de la misma máquina.

Para que un proceso pueda comunicarse con otro proceso situado en otra máquina de la red necesita saber su dirección completa (número de IP y número de puerto).
Para que un ordenador pueda enviar un mensaje a otro, necesita conocer su dirección.

Una dirección IP es una secuencia de 32 bits formada por 4 grupos de 8 bits. Con 8 bits, es posible representar números con un rango entre 0 y 255.

Un ejemplo de dirección IP podría ser: 230.4.27.3.

El puerto es un número de 16 bits, lo que significa que puede alcanzar un rango entre 0 y 65535. Lo más habitual es utilizar 4 cifras decimales.

Ejemplo: 8085.

JAVA: La clase *InetSocketAddress*

Existe una clase Java que representa una dirección en la red, se trata de la clase *InetSocketAddress* que recibe en el constructor la dirección IP y el número de puerto.

Los objetos *InetSocketAddress* resultan de gran utilidad a la hora de establecer la comunicación mediante sockets.

La clase **InetSocketAddress** en Java es una clase de la API de **java.net** que encapsula una dirección de Internet y un número de puerto. Se utiliza comúnmente cuando se trabaja con sockets para especificar tanto la dirección IP (o nombre de host) y el puerto de la conexión.

```
InetSocketAddress direccion = new InetSocketAddress("127.191.3.8",2018);
```

MUCHAS GRACIAS POR VUESTRA ATENCIÓN!