

---

# IIC2523

# Sistemas Distribuidos

— Hernán F. Valdivieso López —  
(2025 - 2 / Clase 03)

---

# **Comunicación entre nodos I**

## **Características y Comunicación directa**

# Temas de la clase

1. Comunicación de red
  - a. Tipos y acoplamientos
  - b. Paradigmas
2. Protocolos de comunicación
  - a. De transporte: TCP vs UDP
  - b. De aplicación: HTTP, DNS, entre otros.
3. Comunicación directa
  - a. Características y *Socket*
  - b. *Socket* en Python con TCP/IP
  - c. MPI

# Comunicación de red

**Tipos y acoplamientos**  
**Paradigma**

---

# Comunicación de red - Tipos

- ◆ El Intercambio de información entre dispositivos conectados a una red se puede clasificar según dos criterios:

Persistencia de los mensajes y sincronía de los mensajes

# Comunicación de red - Tipos

- ◆ El Intercambio de información entre dispositivos conectados a una red se puede clasificar según dos criterios:

**Persistencia de los mensajes** y sincronía de los mensajes

- ◆ **Transitoria**: el mensaje vive sólo mientras las aplicaciones del emisor y receptor están en ejecución. Si uno de los 2 entes cae, el mensaje se pierde.
  - ◆ Llamada de *Google Meet* (no grabada), mensajes por `sockets` de Python

# Comunicación de red - Tipos

- ◆ El Intercambio de información entre dispositivos conectados a una red se puede clasificar según dos criterios:

**Persistencia de los mensajes** y sincronía de los mensajes

- ◆ **Transitoria**: el mensaje vive sólo mientras las aplicaciones del emisor y receptor están en ejecución. Si uno de los 2 entes cae, el mensaje se pierde.
  - ◆ Llamada de *Google Meet* (no grabada), mensajes por *sockets* de Python
- ◆ **Persistente**: se almacena, en algún lado, el mensaje enviado por el emisor el tiempo que tome entregarlo al receptor.
  - ◆ *Outlook, Telegram.*

# Comunicación de red - Tipos

- ◆ El Intercambio de información entre dispositivos conectados a una red se puede clasificar según dos criterios:

Persistencia de los mensajes y **sincronía de los mensajes**

- ◆ **Sincrónica:** El emisor queda bloqueado hasta que confirme que su mensaje es recibido, e incluso que se tenga respuesta del mensaje.
  - ◆ Acceder a una páginas web, la librería "requests" de Python.



# Comunicación de red - Tipos

- ◆ El Intercambio de información entre dispositivos conectados a una red se puede clasificar según dos criterios:

Persistencia de los mensajes y **sincronía de los mensajes**

- ◆ **Sincrónica:** El emisor queda bloqueado hasta que confirme que su mensaje es recibido, e incluso que se tenga respuesta del mensaje.
  - ◆ Acceder a una páginas web, la librería "requests" de Python.
- ◆ **Asincrónica:** El emisor puede continuar ejecutando inmediatamente después de enviar el mensaje, incluso si el mensaje todavía no llega al receptor.
  - ◆ Gmail, transmisión por Zoom.

# Comunicación de red - Acoplamiento

- ◆ Con los conceptos anteriores, una comunicación se puede caracterizar según el tipo de acoplamiento que tiene para que sea exitosa:

Acoplamiento Temporal o Acoplamiento Espacial (o Referencial)

# Comunicación de red - Acoplamiento

- ◆ Con los conceptos anteriores, una comunicación se puede caracterizar según el tipo de acoplamiento que tiene para que sea exitosa:

**Acoplamiento Temporal** o Acoplamiento Espacial (o Referencial)

- ◆ **Acoplado:** El emisor y el receptor deben existir al mismo tiempo para comunicarse.
  - ◆ Llamada por Zoom, transmisión por Youtube.

# Comunicación de red - Acoplamiento

- ◆ Con los conceptos anteriores, una comunicación se puede caracterizar según el tipo de acoplamiento que tiene para que sea exitosa:

**Acoplamiento Temporal** o Acoplamiento Espacial (o Referencial)

- ◆ **Acoplado**: El emisor y el receptor deben existir al mismo tiempo para comunicarse.
  - ◆ Llamada por Zoom, transmisión por Youtube.
- ◆ **Desacoplado**: El emisor y receptor pueden existir en tiempos distintos.
  - ◆ Memoria compartida, las aplicaciones de mensajería como gmail.

# Comunicación de red - Acoplamiento

- ◆ Con los conceptos anteriores, una comunicación se puede caracterizar según el tipo de acoplamiento que tiene para que sea exitosa:

Acoplamiento Temporal o **Acoplamiento Espacial (o Referencial)**

- ◆ **Acoplado:** El emisor conoce o necesita conocer la identidad de los receptores.
  - ◆ Mensaje privado de Whatsapp, Correo electrónico.

# Comunicación de red - Acoplamiento

- ◆ Con los conceptos anteriores, una comunicación se puede caracterizar según el tipo de acoplamiento que tiene para que sea exitosa:

Acoplamiento Temporal o **Acoplamiento Espacial (o Referencial)**

- ◆ **Acoplado:** El emisor conoce o necesita conocer la identidad de los receptores.
  - ◆ Mensaje privado de Whatsapp, Correo electrónico.
- ◆ **Desacoplado:** El emisor no necesita conocer la identidad del receptor o receptores. La comunicación se realiza a través de una entidad intermediaria.
  - ◆ Publicación en *instagram*, Aviso de Canvas.

# Comunicación de red - Paradigmas

◆ Se pueden distinguir 3 paradigmas de comunicación no excluyentes:

## Directa

La forma más simple de comunicación entre procesos a través del paso de mensajes.

Ligero, eficiente y minimalista.

Hay acoplamiento referencial.

## Indirecta

La comunicación se realiza **a través de un intermediario distinto al SSOO.**

Sin acoplamiento directo entre el emisor y el receptor.

Según cómo sea el intermediario, puede que los mensajes sean persistentes o transitorios.

## Invocación Remota

Se invocan operaciones o métodos que se ejecutan en otra máquina conectada en red.

Generalmente, se encapsula el paso de mensaje para simplificar la comunicación.

# Protocolos de comunicación

De transporte: TCP vs UDP

De aplicación: HTTP, DNS, entre otros.

---



# Comunicación de red - Protocolos

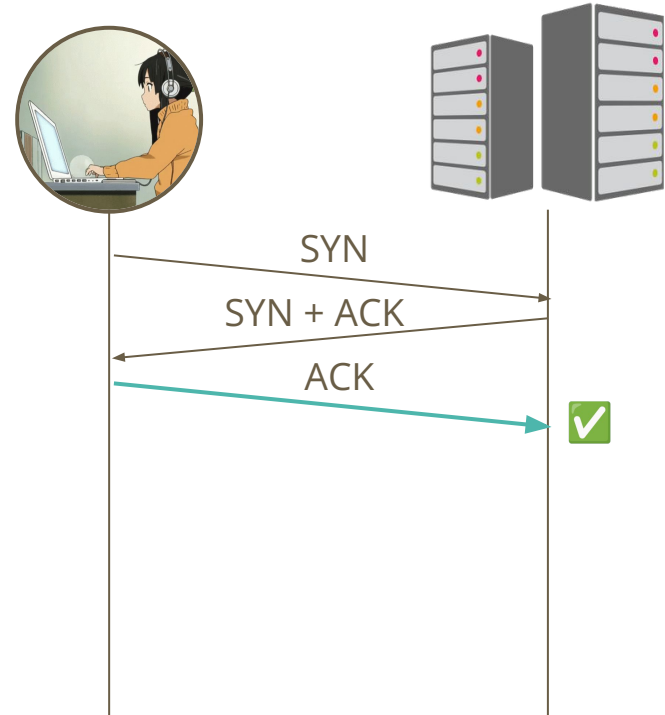
- ◆ ¿Cómo se transporta la información por la red? TCP o UDP
- ◆ ¿Qué pasos sigo para...
  - ◆ ...transferir un archivo? FTP
  - ◆ ...saber la IP de destino mediante la URL? DNS
  - ◆ ...acceder a una página web? HTTP
  - ◆ ...acceder a una página web de forma segura? HTTPS
  - ◆ ...enviar un correo? SMTP
  - ◆ ...acceder remotamente a un servidor por un un canal seguro? SSH

# Comunicación de red - Protocolos

- ◆ ¿Cómo se transporta la información por la red? **TCP o UDP**
- ◆ ¿Qué pasos sigo para...
  - ◆ ...transferir un archivo? FTP
  - ◆ ...saber la IP de destino mediante la URL? **DNS**
  - ◆ ...acceder a una página web? **HTTP**
  - ◆ ...acceder a una página web de forma segura? HTTPS
  - ◆ ...enviar un correo? SMTP
  - ◆ ...acceder remotamente a un servidor por un un canal seguro? SSH
- ◆ Vamos a ver de forma **general** algunos protocolos.

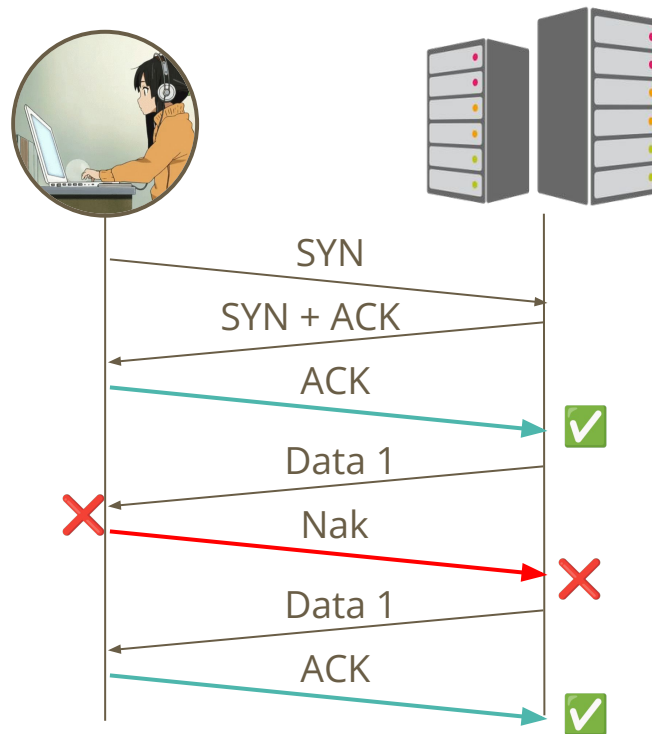
# Comunicación de red - Protocolos de transporte

- ◆ ¿Cómo se transporta la información por la red? **TCP** o UDP
- ◆ Antes de enviar datos, se establece una conexión bidireccional ("*three-way handshake*").
- ◆ Garantiza que todos los datos enviados lleguen a su destino, en el orden correcto y sin duplicados.



# Comunicación de red - Protocolos de transporte

- ◆ ¿Cómo se transporta la información por la red? **TCP** o UDP
- ◆ Antes de enviar datos, se establece una conexión bidireccional ("*three-way handshake*").
- ◆ Garantiza que todos los datos enviados lleguen a su destino, en el orden correcto y sin duplicados.
- ◆ Implementa un mecanismo de *acknowledgments* y retransmisiones para manejar la pérdida de paquetes.

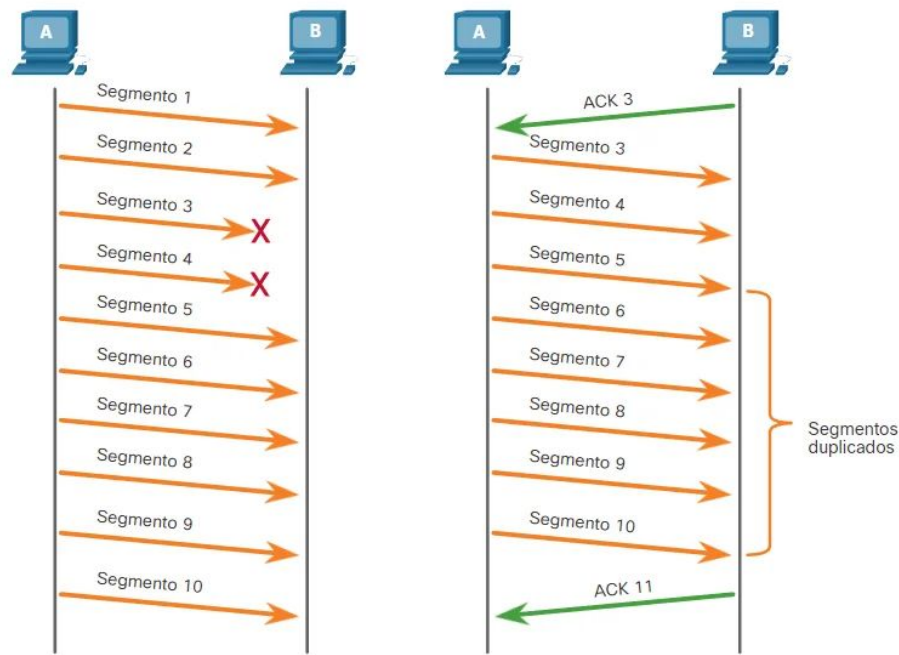


# Comunicación de red - Protocolos de transporte

◆ ¿Cómo se transporta la información por la red? **TCP** o UDP

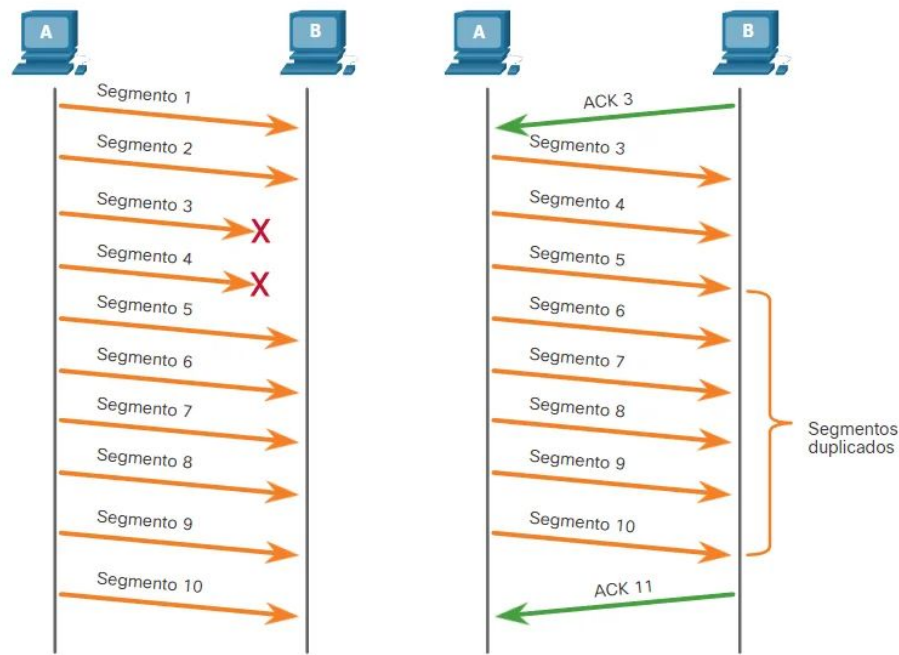
◆ Implementa un mecanismo de *acknowledgments* y retransmisiones para manejar la pérdida de paquetes.

◆ Pueden enviar paquetes de gran tamaño que lo fragmentan en segmentos.



# Comunicación de red - Protocolos de transporte

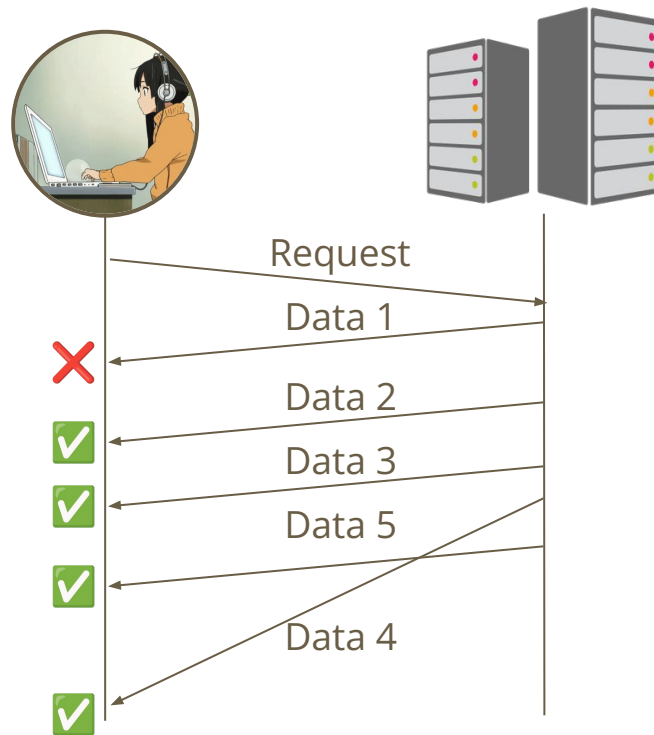
- ◆ ¿Cómo se transporta la información por la red? **TCP** o UDP
- ◆ Implementa un mecanismo de *acknowledgments* y retransmisiones para manejar la pérdida de paquetes.
- ◆ Pueden enviar paquetes de gran tamaño que lo fragmentan en segmentos.
- ◆ Protocolos de aplicación como HTTP, FTP, SMTP, SSH utilizan TCP.



# Comunicación de red - Protocolos de transporte

◆ ¿Cómo se transporta la información por la red? TCP o **UDP**

- ◆ No se establece ninguna conexión bidireccional previa.
- ◆ Manda paquetes de máximo 64Kb.
- ◆ Se pueden perder paquetes o llegar en distinto orden.
- ◆ No reenvía paquetes.
- ◆ Mucho más rápido debido a la ausencia de verificaciones y reenvío.



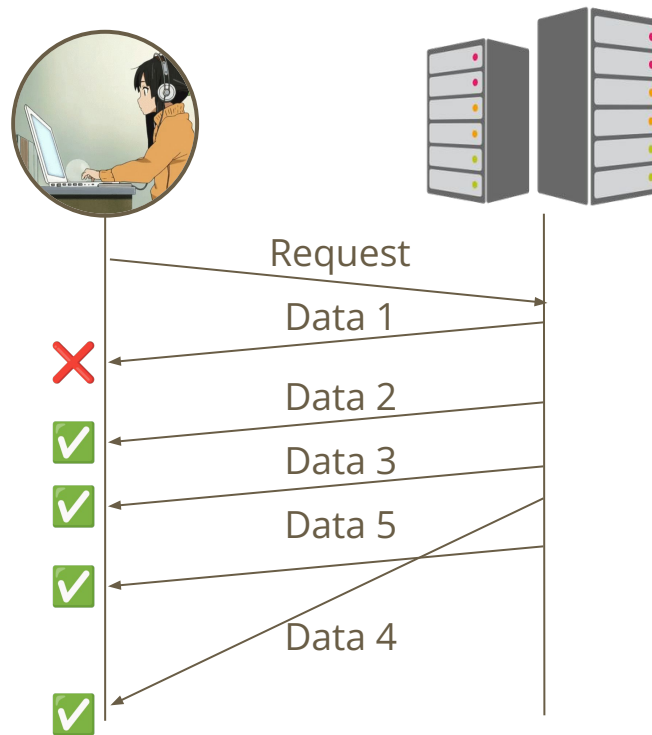
# Comunicación de red - Protocolos de transporte

◆ ¿Cómo se transporta la información por la red? TCP o **UDP**

◆ Usado generalmente cuando la velocidad prioriza sobre la garantía de integridad y orden de los mensajes.

◆ Usado en protocolos como DNS (veremos en 2 diapos más) o NTP (*Network Time Protocol*... veremos eventualmente).

◆ Usado en aplicaciones que requieren la interacción en tiempo real: juegos en línea, llamadas, sensores.





# Comunicación de red - Protocolos de transporte

◆ ¿Cómo se transporta la información por la red? TCP o UDP



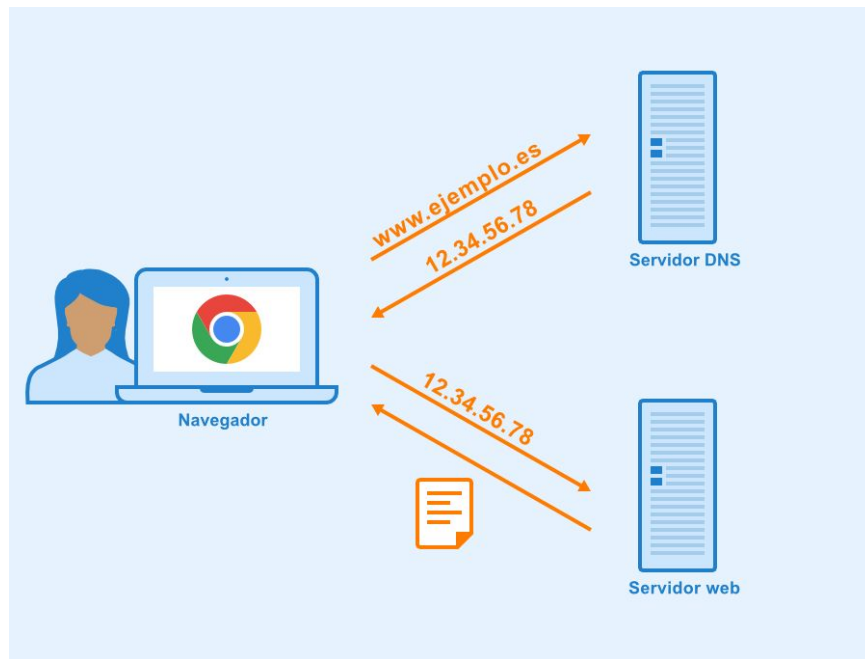
# Comunicación de red - Protocolos de aplicación

◆ ¿Qué pasos sigo para saber la IP de destino mediante la URL? **DNS**

◆ Existe un servidor **distribuido** que maneja este mapeo.

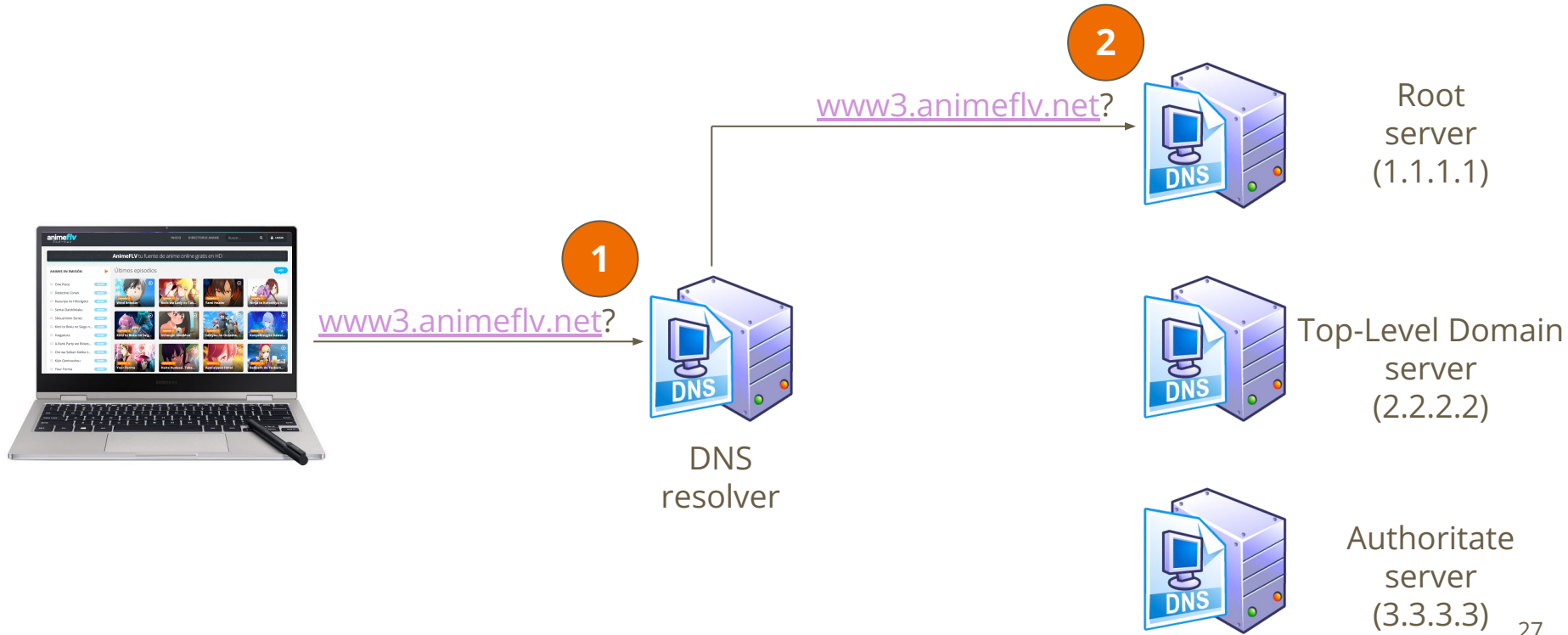
◆ Tanto el sistema operativo como los servidores guardan en caché las consultas para facilitar la búsqueda.

◆ Si la caché no tiene resultado, comienza la búsqueda...



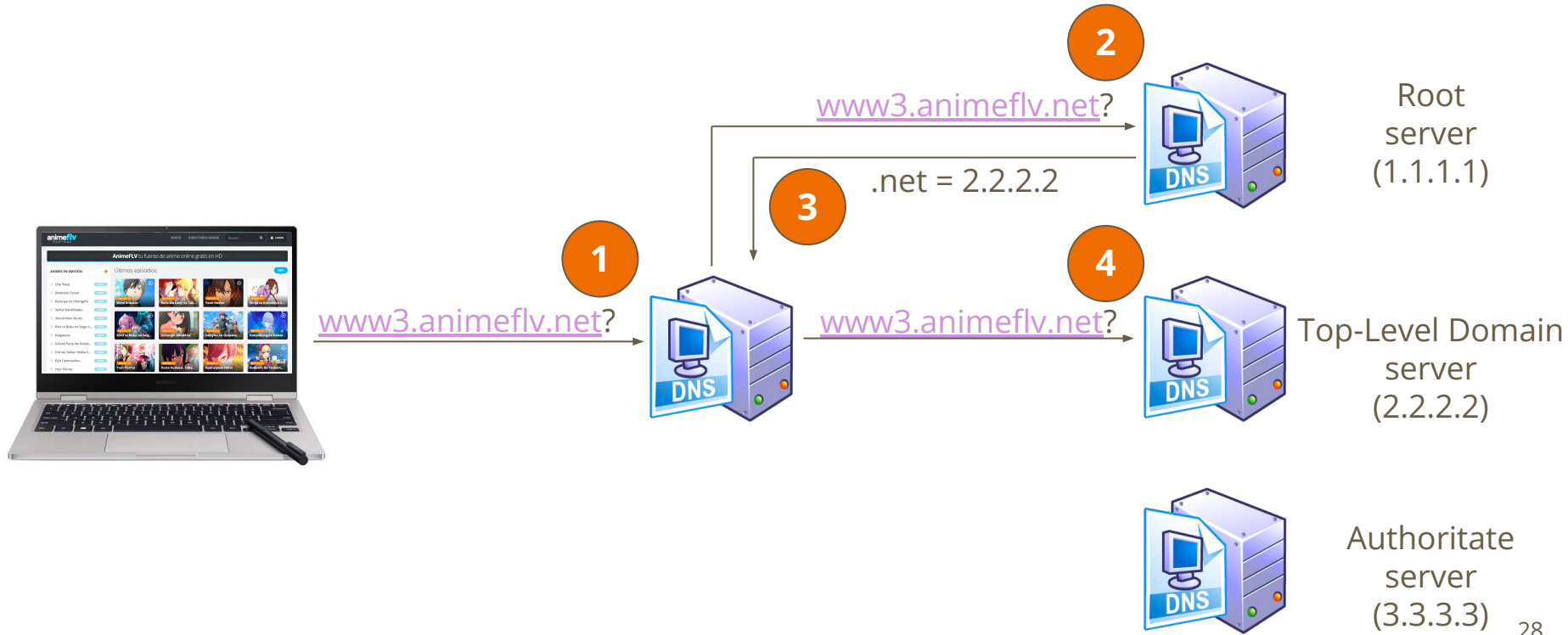
# Comunicación de red - Protocolos de aplicación

◆ ¿Qué pasos sigo para saber la IP de destino mediante la URL? **DNS**



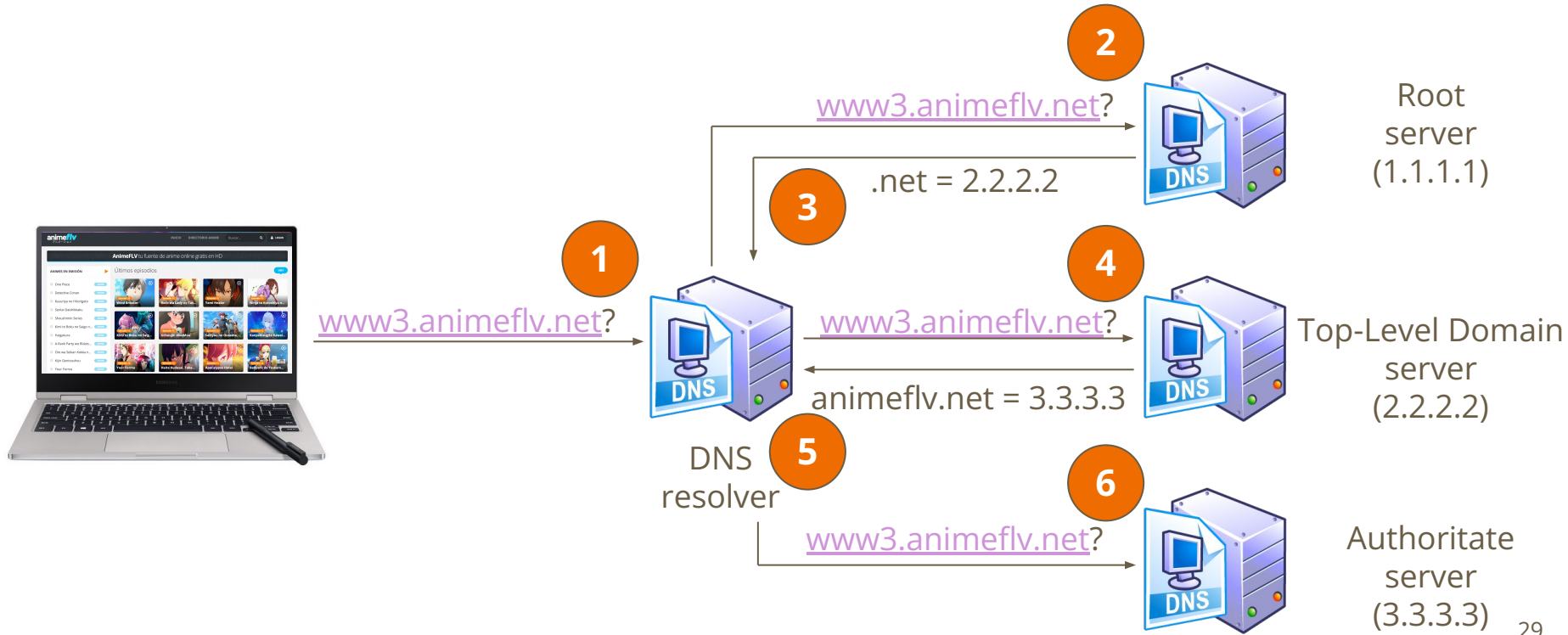
# Comunicación de red - Protocolos de aplicación

◆ ¿Qué pasos sigo para saber la IP de destino mediante la URL? **DNS**



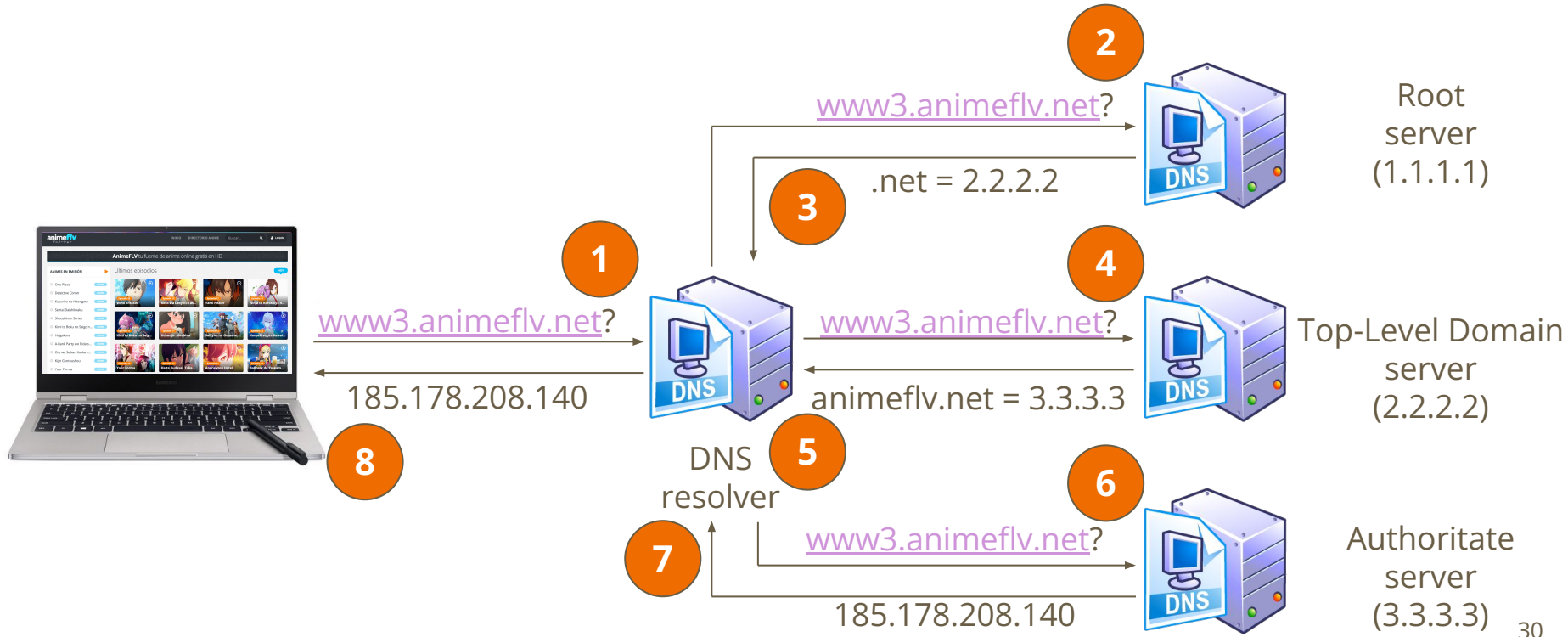
# Comunicación de red - Protocolos de aplicación

◆ ¿Qué pasos sigo para saber la IP de destino mediante la URL? **DNS**



# Comunicación de red - Protocolos de aplicación

◆ ¿Qué pasos sigo para saber la IP de destino mediante la URL? **DNS**



# Comunicación de red - Protocolos de aplicación

◆ ¿Qué pasos sigo para acceder a una página web? **HTTP**

◆ Primero conseguir la IP (DNS)

◆ Luego construir la petición (el mensaje) con el formato que solicita HTTP.

Petición

```
GET /anime/detective-conan HTTP/1.1  
Host: www3.animeflv.net  
Remote Address: 185.178.208.140:443  
Accept: text/html
```

# Comunicación de red - Protocolos de aplicación

◆ ¿Qué pasos sigo para acceder a una página web? **HTTP**

- ◆ Primero conseguir la IP (DNS)
- ◆ Luego construir la petición (el mensaje) con el formato que solicita HTTP.
- ◆ Se envía mensaje por protocolo de transporte TCP hacia el servidor.
- ◆ Esperar respuesta del servidor.

## Petición

```
GET /anime/detective-conan HTTP/1.1
Host: www3.animeflv.net
Remote Address: 185.178.208.140:443
Accept: text/html
```

## Respuesta

```
HTTP/1.1 200 OK
Server: www3.animeflv.net
Content-Type: text/html
Content-Length: 4444
<html>
...
</html>
```



# Comunicación directa

Características

*Socket*

*Socket* en Python con TCP/IP

MPI

---

# Comunicación directa

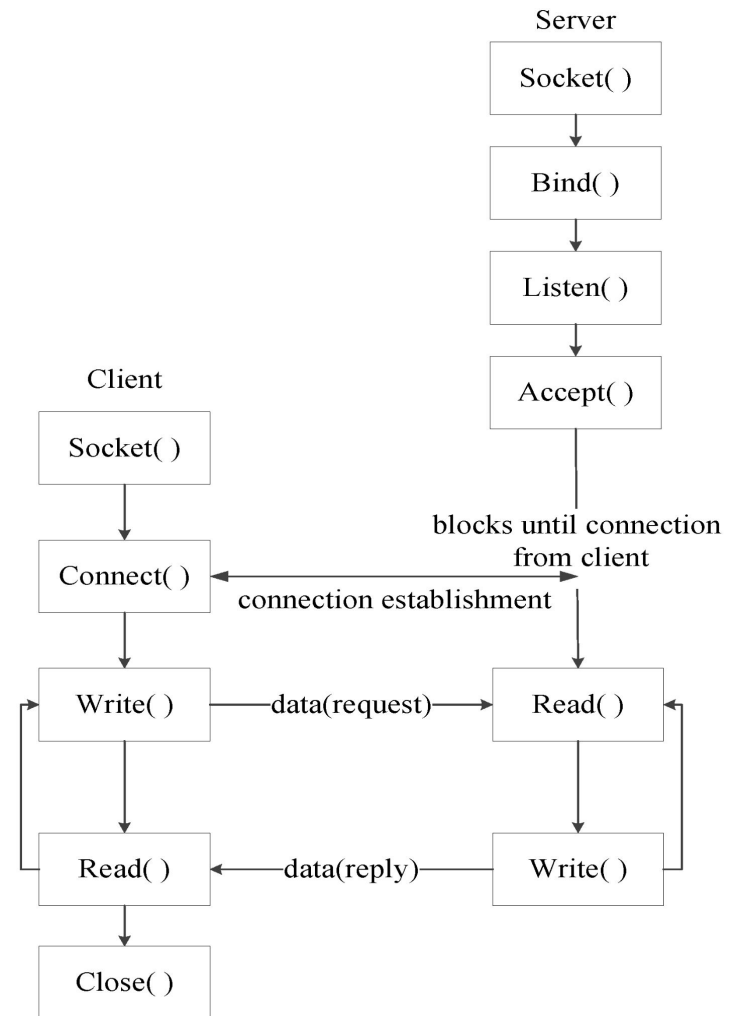
- ◆ La forma más simple de comunicación entre procesos es a través del pasaje de mensajes.
- ◆ El proceso emisor debe especificar el destino del proceso receptor (puede ser con una dirección y un puerto).

# Comunicación directa

- ◆ La forma más simple de comunicación entre procesos es a través del pasaje de mensajes.
- ◆ El proceso emisor debe especificar el destino del proceso receptor (puede ser con una dirección y un puerto).
- ◆ Herramienta clásica: *Socket*
  - ◆ Es una interfaz de entrada-salida de datos que permite la intercomunicación entre procesos.
  - ◆ Se pueden asociar al protocolo UDP o TCP.
  - ◆ Se debe diferenciar si el socket actúa como cliente o como servidor.
  - ◆ Con el uso de *thread* es posible tener interesantes comportamientos como múltiples servidores en un mismo proceso, atender múltiples clientes, etc.

# Comunicación directa - Socket

- ◆ Es una interfaz de entrada-salida de datos que permite la intercomunicación entre procesos.
- ◆ Se pueden asociar al protocolo UDP o TCP.
- ◆ Se debe diferenciar si el socket actúa como cliente o como servidor.
- ◆ Con el uso de *thread* es posible tener interesantes comportamientos como múltiples servidores en un mismo proceso, atender múltiples clientes, etc.



# Socket - Python (Servidor)

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind(('localhost', 4444))
sock.listen(5)
i = 0
while i < 2:
    connection, client_address = sock.accept()
    largo = connection.recv(4)
    mensaje = connection.recv(int.from_bytes(largo, byteorder="big"))
    print(mensaje.decode("UTF-8"))
    connection.sendall(largo + mensaje)
    connection.close()
    i += 1
sock.close()
```

# Socket - Python (Cliente)

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('localhost', 4444))

mensaje = "waku waku".encode("UTF-8")
largo = len(mensaje).to_bytes(4, byteorder="big")
sock.sendall(largo)
sock.sendall(mensaje)
largo = sock.recv(4)
data = sock.recv(int.from_bytes(largo, byteorder="big"))
print(data.decode("UTF-8"))
sock.close()
```

# Comunicación directa - MPI

- ◆ Es un "estándar" para comunicación explícita entre procesos en sistemas distribuidos.
  - ◆ No es oficialmente un estándar, pero al momento de implementar, se respetan ciertas reglas.
- ◆ MPI promueve un modelo *Single Program, Multiple Data* (SPMD)
  - ◆ El mismo código ejecutado por distintos procesos, cada uno con su propia memoria.
- ◆ Soporta:
  - ◆ Comunicación punto a punto.
  - ◆ Comunicación colectiva entre varios nodos.

# Comunicación directa - MPI

Las llamadas de MPI se dividen en cuatro clases:

- ◆ Llamadas utilizadas para inicializar, administrar y finalizar comunicaciones.
  - ◆ MPI\_Init, MPI\_Finalize, MPI\_Comm\_size y MPI\_Comm\_rank
- ◆ Llamadas utilizadas para transferir datos entre un par de procesos.
  - ◆ MPI\_Send, MPI\_Recv
- ◆ Llamadas para transferir datos entre varios procesos.
  - ◆ MPI\_Barrier, MPI\_Bcast
- ◆ Llamadas utilizadas para crear tipos de datos definidos por el usuario.
  - ◆ MPI\_Type\_struct



# Comunicación directa - MPI (Python)

```
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
    print(f"Proceso 0 envía mensaje")
    comm.send("Waku Waku", dest=1)
elif rank == 1:
    msg = comm.recv(source=0)
    print(f"Proceso 1 recibió: {msg}")
```

```
mpiexec -n 2 python ejemplo.py
```

```
> Proceso 0 envía mensaje
> Proceso 1 recibió: Waku Waku
```

# Poniendo a prueba lo que hemos aprendido 🧐

**Supongamos** que su profesor hará una conferencia por Zoom para hablar de anime. Pero, como no tiene Zoom Pro, no va a grabar esa conferencia. Además, para mantener la privacidad de usuarios, Zoom será un intermediario que recibirá la información y reenviará la transmisión a todos los presentes sin que el profesor sepa realmente sus identidades.

¿Cuál de las siguientes afirmaciones describe de forma más precisa la naturaleza del intercambio de información?

- a) Es una comunicación persistente, asincrónica y acoplado referencialmente.
- b) Es una comunicación acoplada referencialmente y utiliza protocolo TCP para el envío de cada *frame*.
- c) Es una comunicación transitoria, asincrónica y acoplada temporal.
- d) Es una comunicación desacoplada temporal y referencialmente.
- e) Es una comunicación persistente y que utiliza el protocolo UDP para el envío de cada *frame*.

# Poniendo a prueba lo que hemos aprendido 🧐

**Supongamos** que su profesor hará una conferencia por Zoom para hablar de anime. Pero, como no tiene Zoom Pro, no va a grabar esa conferencia. Además, para mantener la privacidad de usuarios, Zoom será un intermediario que recibirá la información y reenviará la transmisión a todos los presentes sin que el profesor sepa realmente sus identidades.

¿Cuál de las siguientes afirmaciones describe de forma más precisa la naturaleza del intercambio de información?

- a) Es una comunicación persistente, asincrónica y acoplado referencialmente.
- b) Es una comunicación acoplada referencialmente y utiliza protocolo TCP para el envío de cada *frame*.
- c) Es una comunicación transitoria, asincrónica y acoplada temporal.**
- d) Es una comunicación desacoplada temporal y referencialmente.
- e) Es una comunicación persistente y que utiliza el protocolo UDP para el envío de cada *frame*.

# Próximos eventos

## Próxima clase

- ◆ Comunicación entre nodos II
  - ◆ Formas de comunicarnos sin recurrir directamente al paso de mensaje.
  - ◆ Estudiaremos *Remote Procedure Call (RPC)*. Importante para la Tarea 1.

## Evaluación

- ◆ Control 1, se publicó ayer. Recomendando hacerlo con tiempo y con PPT en manos.
- ◆ Control 2, se publica el otro martes y evalúa esta clase y la siguiente.
- ◆ Tarea 1 *spoiler*: levantar más de un servidor TCP con *socket* que no se quede bloqueado ante múltiples clientes conectados.

---

# IIC2523

# Sistemas Distribuidos

— Hernán F. Valdivieso López —  
(2025 - 2 / Clase 03)

---

# Créditos (animes utilizados)

K-On!

