Programación Avanzada
IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattan

Anuncios

Jueves 12 de octubre 2023

1. Hoy tendremos la cuarta experiencia.
2. Ya se encuentra publicada la Tarea 2 del curso.
3. El martes 17 es el *midterm*.
4. Encuesta de Carga Académica ¡Respóndanla!

Evaluación Temprana de Cursos (ETC)

Aspectos que han presentado problemas 🤧

- *Flipped Classroom*
- Demasiados contenidos por semana
- Más ayudantes por sala
- Respuestas a ejercicios propuestos

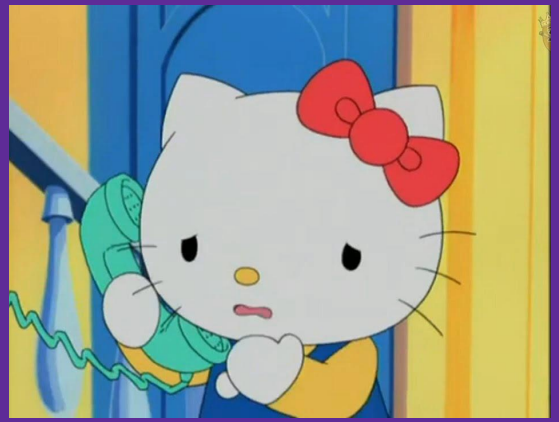
Aspectos positivos 🤩

- Aumentar plazos de actividades
- Cátedras con repasos más largos
- *Tests* para evaluaciones

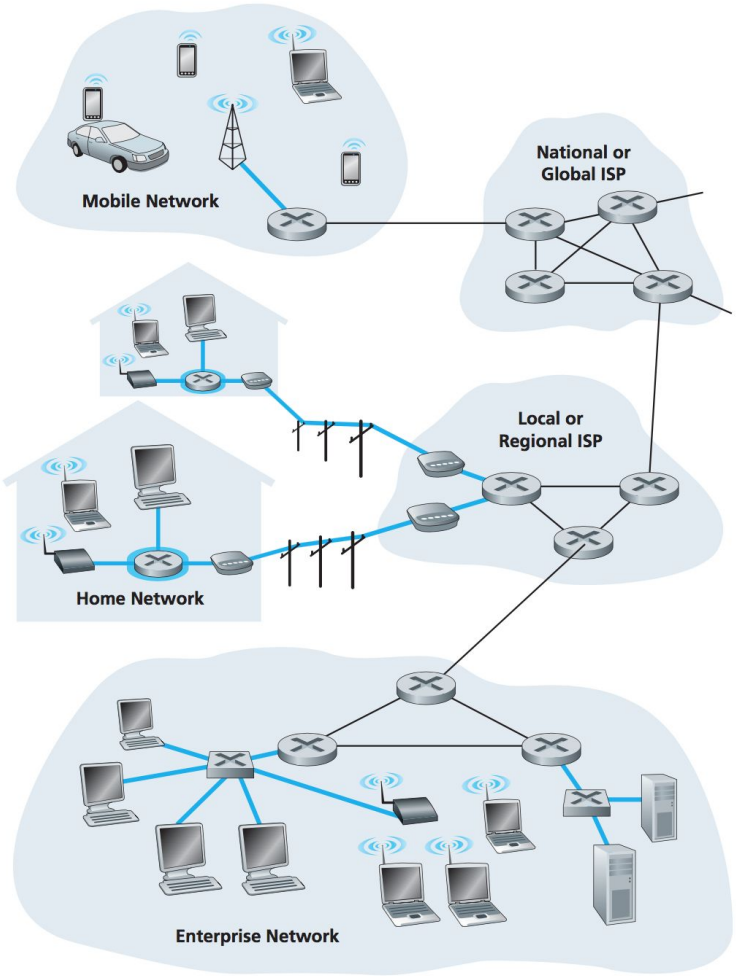
Qué puedo hacer como estudiante 🤔

- Leer los contenidos semanales con tiempo
- Participar en clase (dudas/comentarios)
- Preguntar lo que no entienda

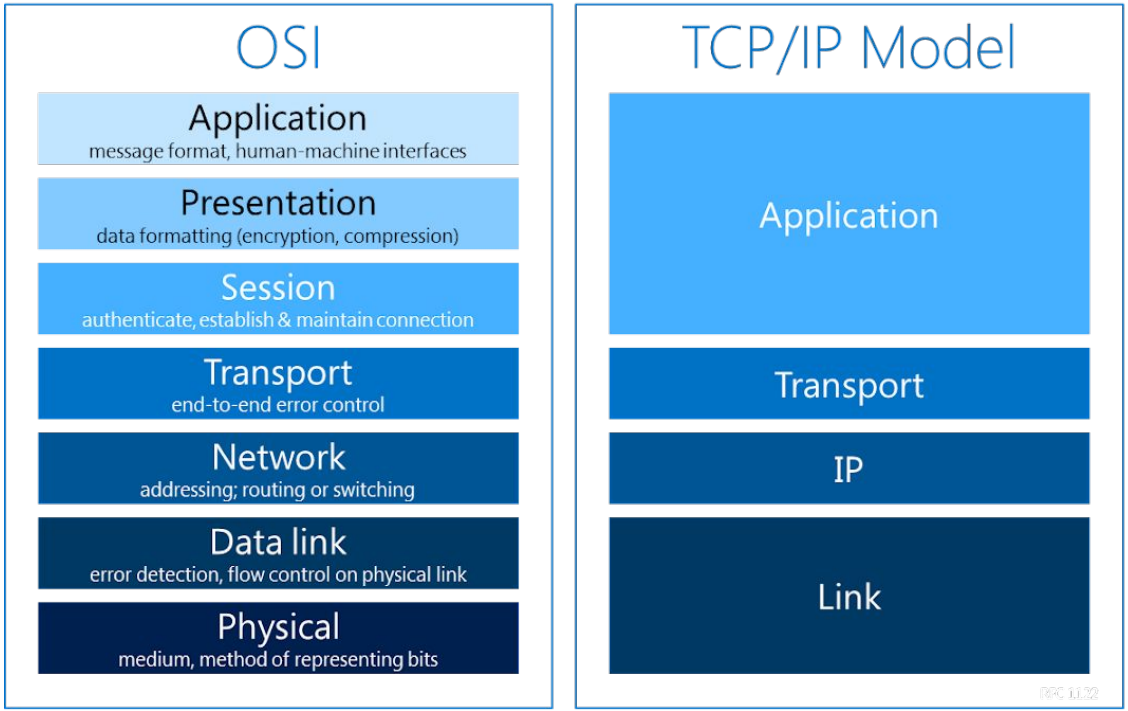
Networking



Redes


Mobile Network
National or
Global ISP
Local or
Regional ISP
Home Network
Enterprise Network

Encapsulamiento



Puertos e IPs

- **IP:** Identifica al computador

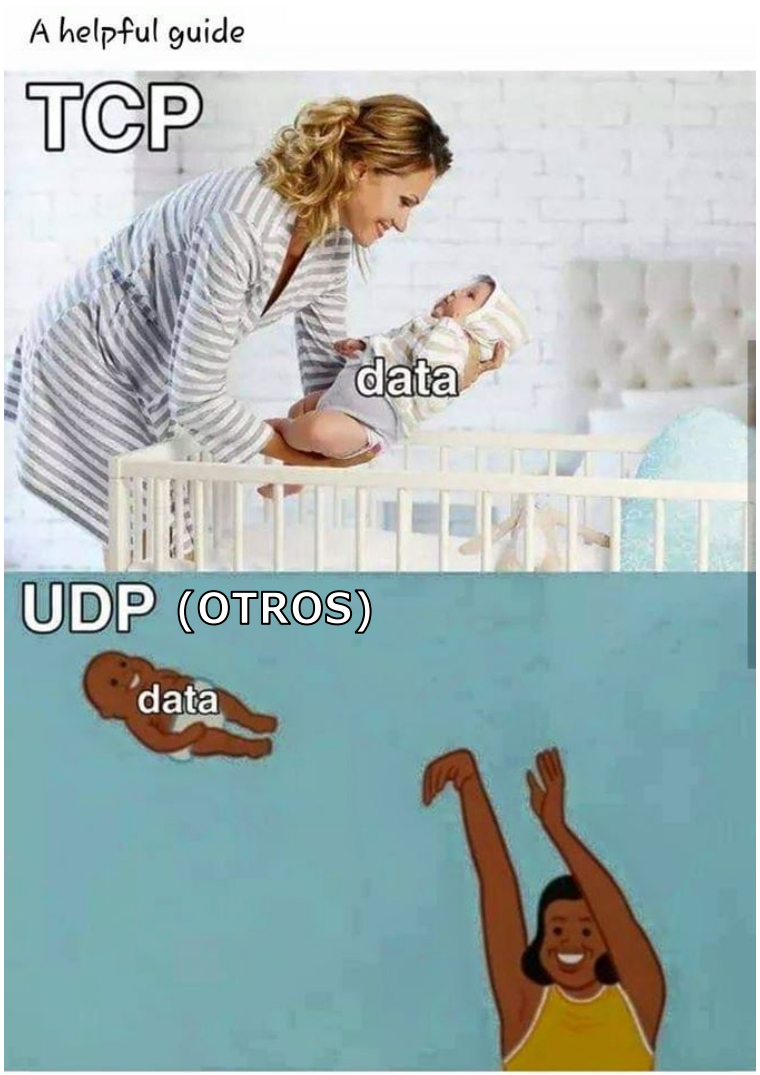
- **Puerto:** Identifica a la aplicación

Protocolos de transporte

TCP (*Transmission Control Protocol)*

- Orientado a conexión.
  - Requiere de *handshake* (establecimiento de conexión) antes de transferir datos.
- Verifica que todos los paquetes que se envían sean recibidos por el destinatario.
- Lo anterior hace que sea más lento por *overhead.*
  - Otros protocolos pueden ser más rápidos, pero no necesariamente aseguran que toda la información se envíe correctamente.
- Reserva *buffers* en *sender* y en *receiver.*
- Algunos casos de uso: Navegación web, emails, transferencia de archivos.

Existen más protocolos, pero en este curso nos enfocaremos en TCP


A helpful guide
TCP
data
UDP (OTROS)
data

Arquitectura Cliente - Servidor y *Sockets*

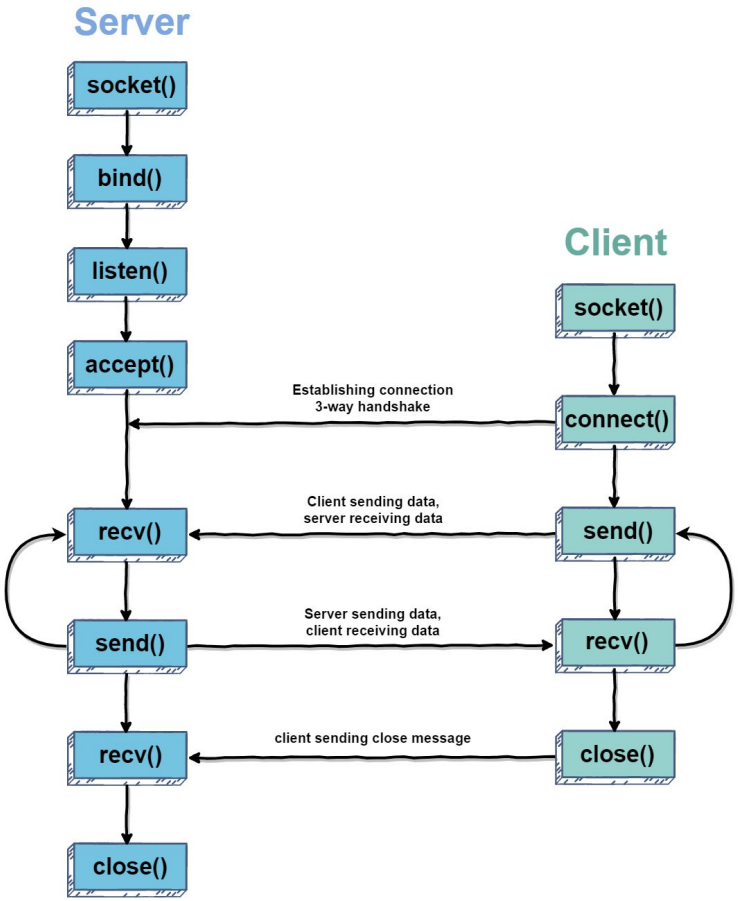
Socket

Es un **objeto del sistema operativo** que permite a un programa **transmitir y recibir datos** desde y hacia otro programa corriendo en otra máquina, o en la misma máquina pero en otro puerto.

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Flujo de comunicación con *sockets* entre Cliente y Servidor



Sockets: Servidor

Para crear el servidor es necesario enlazarlo con la dirección y el puerto deseado, y luego quedar escuchando clientes.

```
serverAddress = "127.0.0.1" # También sirve poner “localhost”
serverPort = 9999 # Valores altos tienden a estar desocupados

# ServerAddress y ServerPort son variables seleccionadas previamente.
socket.bind((serverAddress, serverPort))

# Habilitamos al socket para escuchar clientes
socket.listen()

# Aceptamos un cliente en particular.
# Este método retorna cuando un cliente en algún
# lugar se conecta a este servidor
socket_cliente, address = socket.accept()
```

Sockets: Cliente

Para conectar un cliente a un servidor debemos crear el *socket* y conectarlo al puerto y la dirección del servidor.

```
SERVER_ADDRESS = "127.0.0.1"
SERVER_PORT = 9999
cliente = socket.socket()
cliente.connect((SERVER_ADDRESS, SERVER_PORT))
```

Enviando y recibiendo información

Para enviar y recibir información (***bytes***) desde el cliente al servidor (y viceversa), se utiliza el método ***send*** o ***sendall*** para enviar datos, y el método ***recv*** para recibir bytes.

```
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.bind((ServerAddress, ServerPort))
socket.listen()
socket_cliente, address = socket.accept()
socket_cliente.send("Hola".encode("ascii"))
data = socket_cliente.recv(1024)
print(data.decode("ascii"))
```

Enviando y recibiendo información

Para enviar y recibir información (***bytes***) desde el cliente al servidor (y viceversa), se utiliza el método ***send*** o ***sendall*** para enviar datos, y el método ***recv*** para recibir bytes.

```
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.connect((ServerAddress, ServerPort))
data = socket.recv(1024)
print(data.decode("ascii"))
socket.send("Hola de vuelta!".encode("ascii"))
```

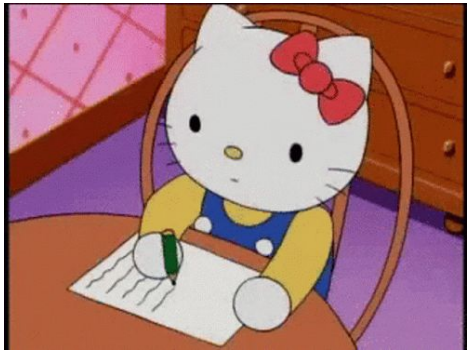
Agregando *Threads* a nuestro programa

Podemos aprovechar los *threads* de python para hacer que nuestro servidor y cliente **no se bloqueen** mientras esperan un mensaje. Para esto, podemos tener distintos *threads* encargados de distintos aspectos del programa: aceptar conexiones, escuchar y manejar los mensajes recibidos, entre otros.

En la experiencia de hoy veremos un ejemplo de esto 😎

Experiencia 4:
Vamos a ella

Midterm: Aclaraciones y Dudas



1. No necesitan traer computador, solo lápiz y papel.
2. La prueba se compone de una sección de alternativas y una de desarrollo.
3. Contenidos hasta semana previa a receso (excepciones y *bytes)*
4. ¿Dudas?

Tarea 2: Interfaces gráficas y *Networking*

1. Veamos el enunciado
2. Esta tarea tiene entrega de avance parcial, evaluada mediante *tests*.
3. La próxima semana tendremos nuestra primera sala de ayuda de la tarea.

Programación Avanzada
IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattan