

# ***Programación Avanzada***

## **IIC2233 2024-1**

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Dante Pinto - Francisca Cattán



# Anuncios



1. Hoy tendremos la cuarta experiencia.
2. El lunes se publica la última tarea enfocado en PyQt y Networking.
3. Encuesta de Carga Académica ¡Respóndanla!



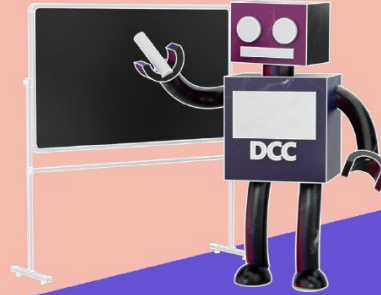
**DCC**  
Departamento de  
Ciencia de la computación

Y C3 PRESENTAN:

# DCCONOCE A TU PROFE

10 JUNIO - 3PM

INSCRIPCION:



**AUDITORIO  
ANDRÓNICO  
RONICO LUKSIC**



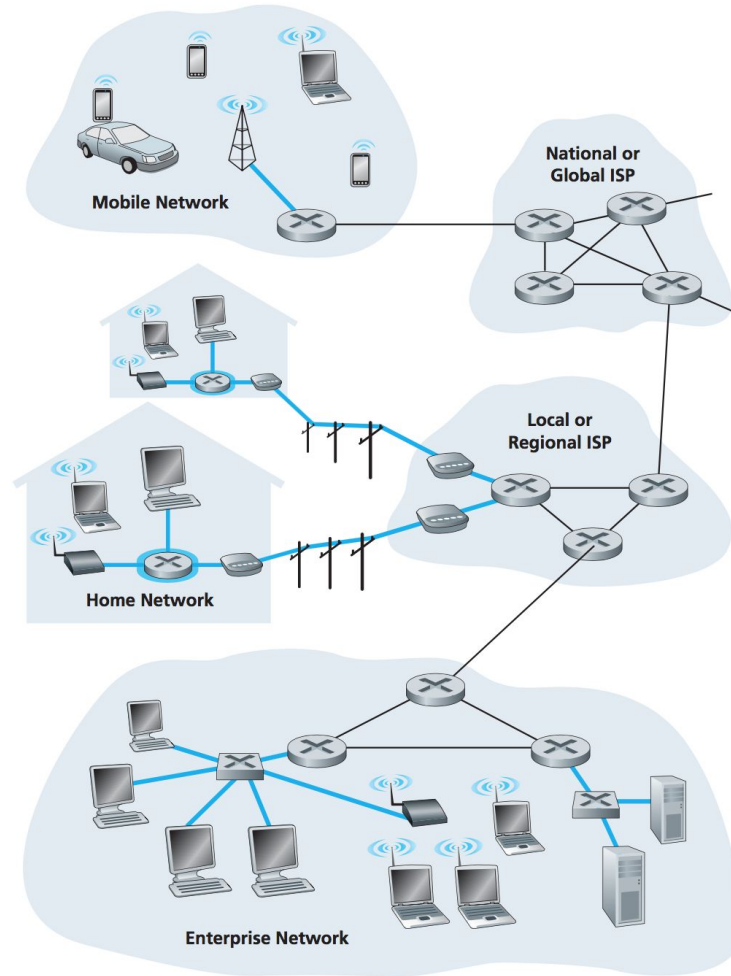
Charlas sobre <  
Hobbies <  
Docencia <  
Curiosidades <  
Investigación <



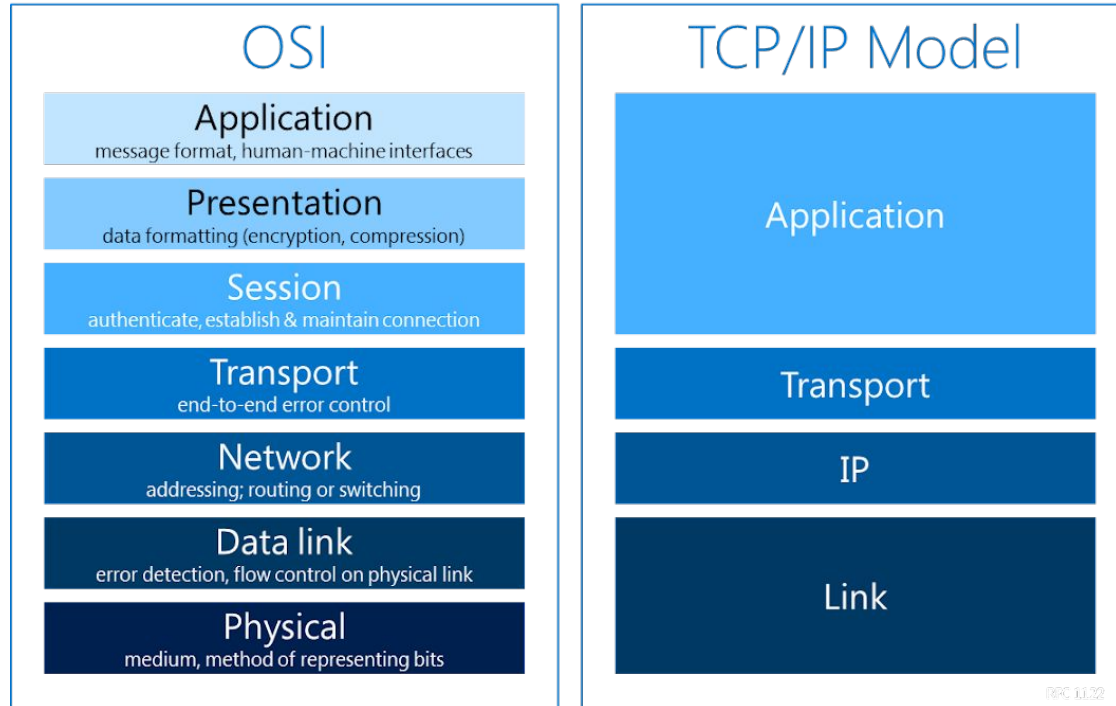
# *Networking*

---

# Redes



# 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**



# Arquitectura Cliente - Servidor *y Sockets*

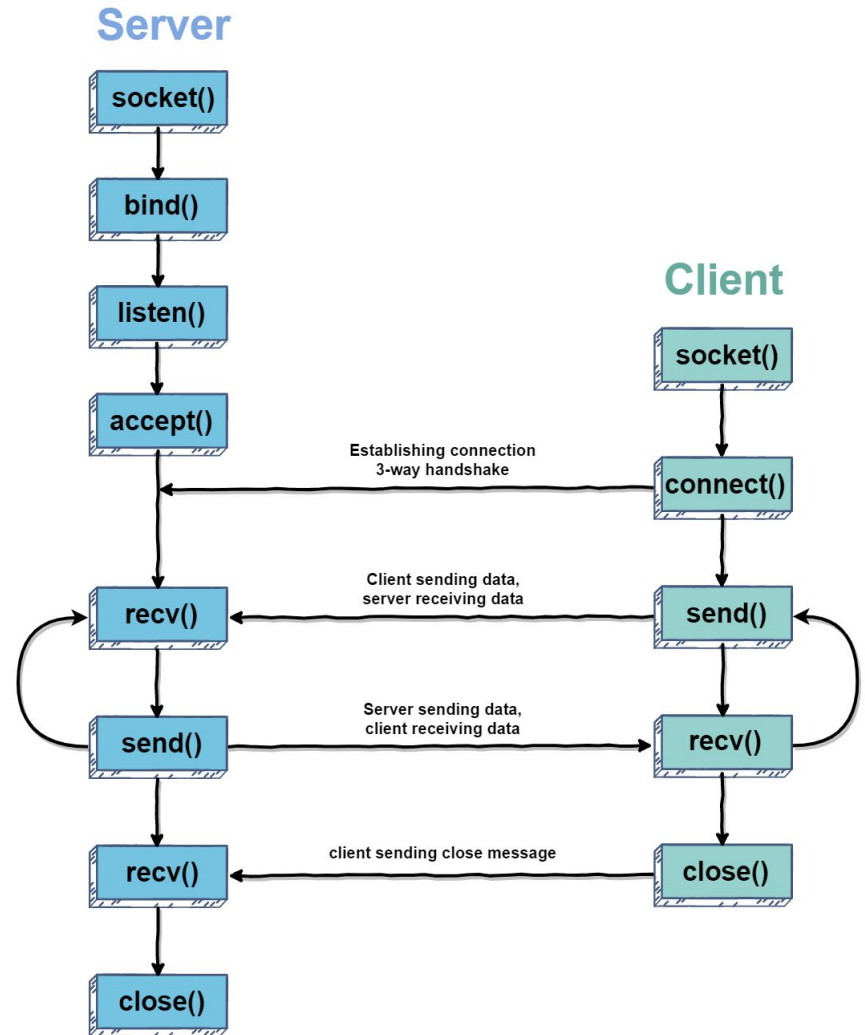
---

# ***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  
mi_socket = 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.

mi_socket.bind((ServerAddress, ServerPort)) # ServerAddress y ServerPort son las
                                             # variables def. previamente.

mi_socket.listen() # Habilitamos el socket para escuchar clientes.

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

# ***Sockets: Cliente***

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

```
ServerAddress = "127.0.0.1"  
ServerPort = 9999
```

```
cliente = socket.socket()  
cliente.connect((ServerAddress, ServerPort))
```

# 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**



# ***Programación Avanzada***

## **IIC2233 2024-1**

Hernán Valdivieso - Daniela Concha -

- Dante Pinto - Francisca Cattán

