# UT3-PROGRAMACIÓN DE COMUNICACIONES EN RED

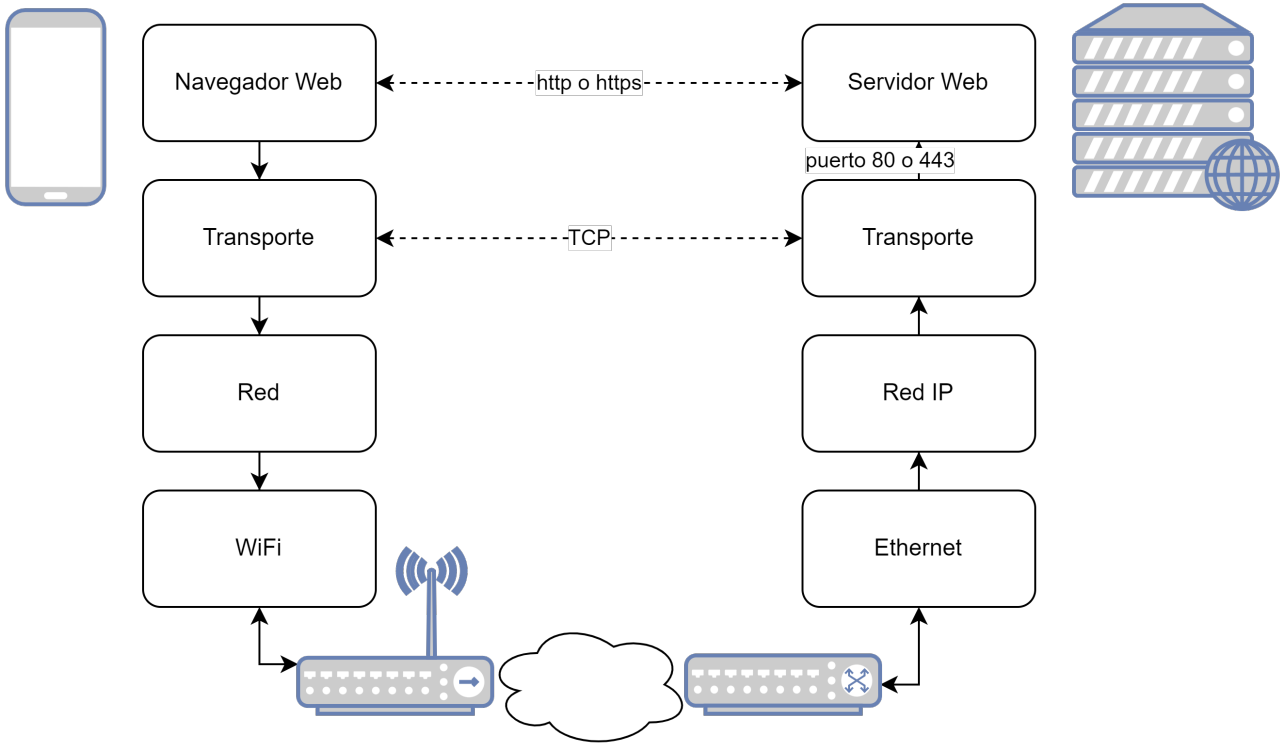
Programación de Servicios y Procesos

# OBJETIVOS

- Entender el flujo de información entre capas en TCP/IP
- Programar la abstracción Socket para la comunicaciones en red.

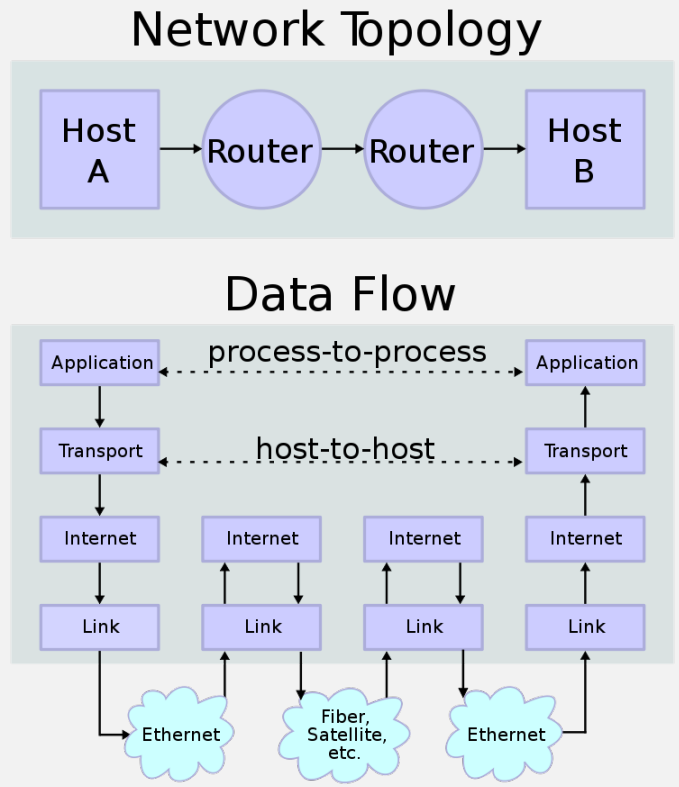
# TCP/IP

# CARGAR UNA PÁGINA WEB



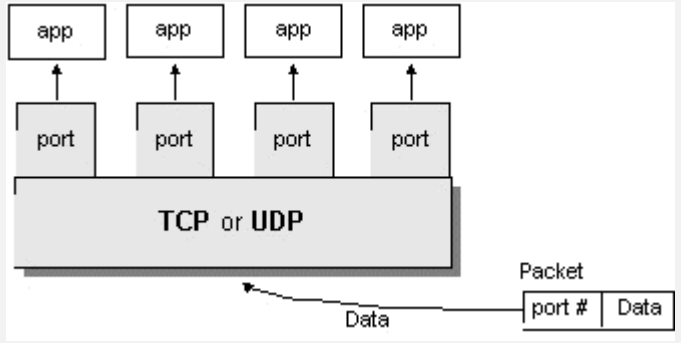
# MODELO TCP/IP

- Cuatro capas
- Aplicación: HTTP, POP, SMTP, etc. Cada cual en su puerto
- Transporte: comunicación entre equipos
  - TCP
    - Transmission Control Protocol
    - fiable, orientado a conexión, sobrecarga
    - Análogo a una llamada telefónica
  - UDP
    - User Datagram Protocol
    - no fiable, no orientado a conexión, veloz
    - Análogo al sistema postal
- Red: lleva paquetes a su destino
- Enlace: comunicaciones dentro de la propia red



# PUERTOS

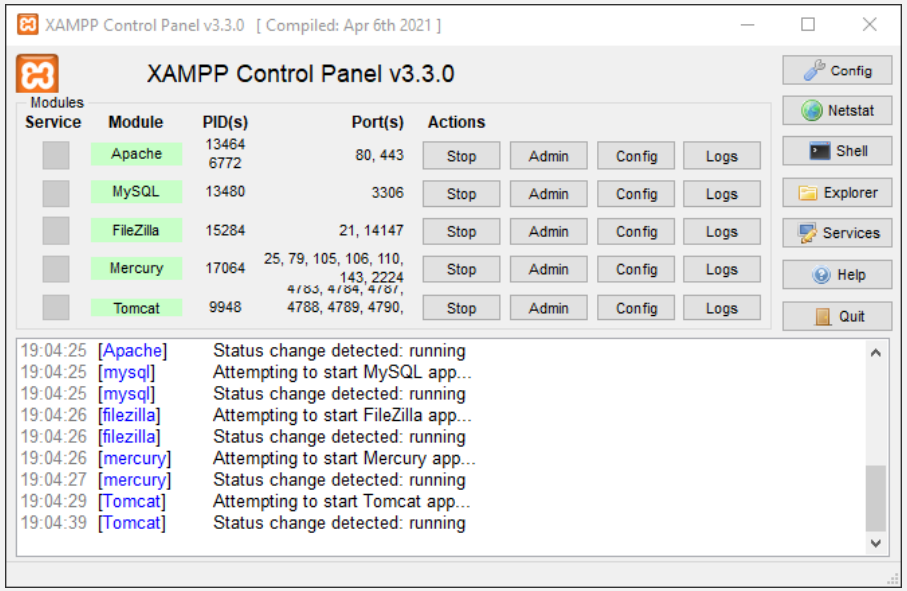
- Cada aplicación en un host tiene su propio puerto
- Hay desde 0 hasta 65.535 ($2^{16}$ o 2 bytes), asignados por ICANN
- Desde 0 a 1023 (well-known ports) y 1024 a 49.151 (registered port) se usan por aplicaciones conocidas:
  - 21 FTP, 22 SSH, 25 SMTP, 80 HTTP, 443 HTTPS, 3389 Remote Desktop Protocol
- De 49.152 a 65.535 (private ports) uso más común utilizados por los clientes
- Windows usa rango privado desde 1024 a 64511




```
C:\Users\juan>netsh int ipv4 show dynamicport tcp

Intervalo de puerto dinámico de protocolo tcp
---------------------------------------------------------
Puerto de inicio    : 1024
Número de puertos   : 64511
```

# NETSTAT




```
C:\Windows\system32>netstat -afb

Conexiones activas

  Proto  Dirección local        Dirección remota       Estado
  TCP    0.0.0.0:21             XPS-WIN:0              LISTENING
 [filezillaserver.exe]
  TCP    0.0.0.0:25             XPS-WIN:0              LISTENING
 [mercury.exe]
  TCP    0.0.0.0:79             XPS-WIN:0              LISTENING
 [mercury.exe]
  TCP    0.0.0.0:80             XPS-WIN:0              LISTENING
 [httpd.exe]
  TCP    0.0.0.0:105            XPS-WIN:0              LISTENING
 [mercury.exe]
  TCP    0.0.0.0:106            XPS-WIN:0              LISTENING
 [mercury.exe]
  TCP    0.0.0.0:110            XPS-WIN:0              LISTENING
 [mercury.exe]
  TCP    0.0.0.0:135            XPS-WIN:0              LISTENING
  RpcSs
 [svchost.exe]
  TCP    0.0.0.0:143            XPS-WIN:0              LISTENING
 [mercury.exe]
  TCP    0.0.0.0:443            XPS-WIN:0              LISTENING
 [httpd.exe]
```

# URL

# URL

- Uniform Resource Locator
- Paquete java.net
- URL.openStream() nos devuelve un Stream del que podemos leer
- Ver 1.url.download
- Ejercicio: completar 1.url.download para descargar archivos binarios

# URLENCODER

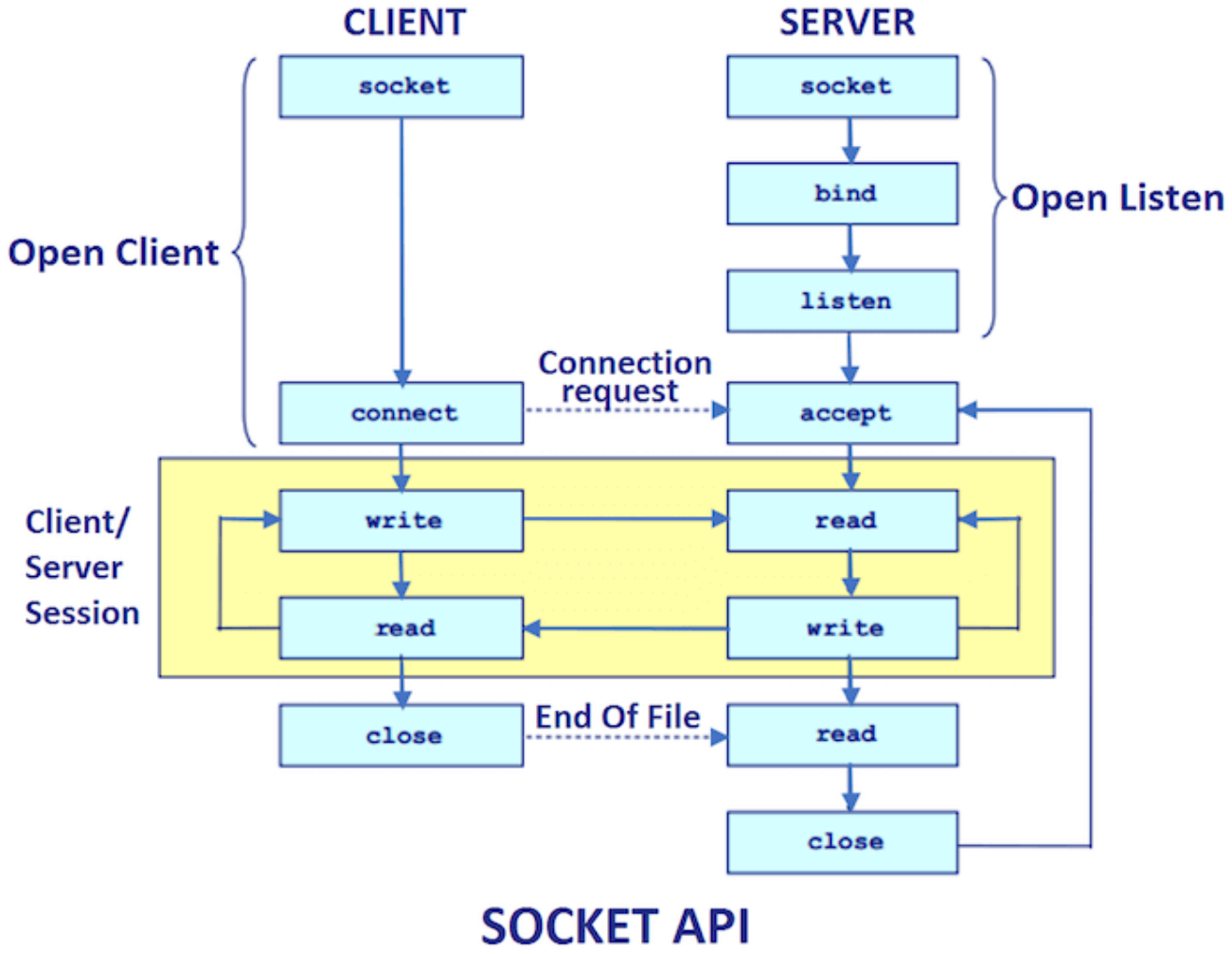
- Recuerda usar URLEncoder para los caracteres especiales en la URL
  - Alfanuméricos no se modifican
  - Caracteres especiales . - _ * no se modifican
  - Los espacios se sustituyen por +

# SOCKETS

# SOCKET

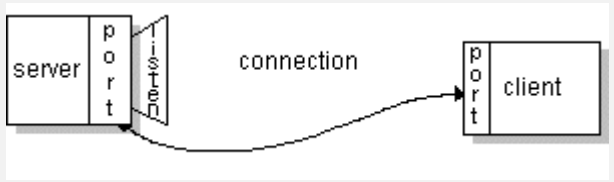
- Análogo a una tubería entre dos aplicaciones
- Abstracción de toda la pila TCP/IP y de los detalles de cada plataforma
- Nos permite escribir aplicaciones clientes y servidoras
- Se especifica con dirección IP y puerto
  - Pueden pertenecer a la misma máquina

# API DE SOCKETS



# SOCKETS

Servidor crea un socket y escucha

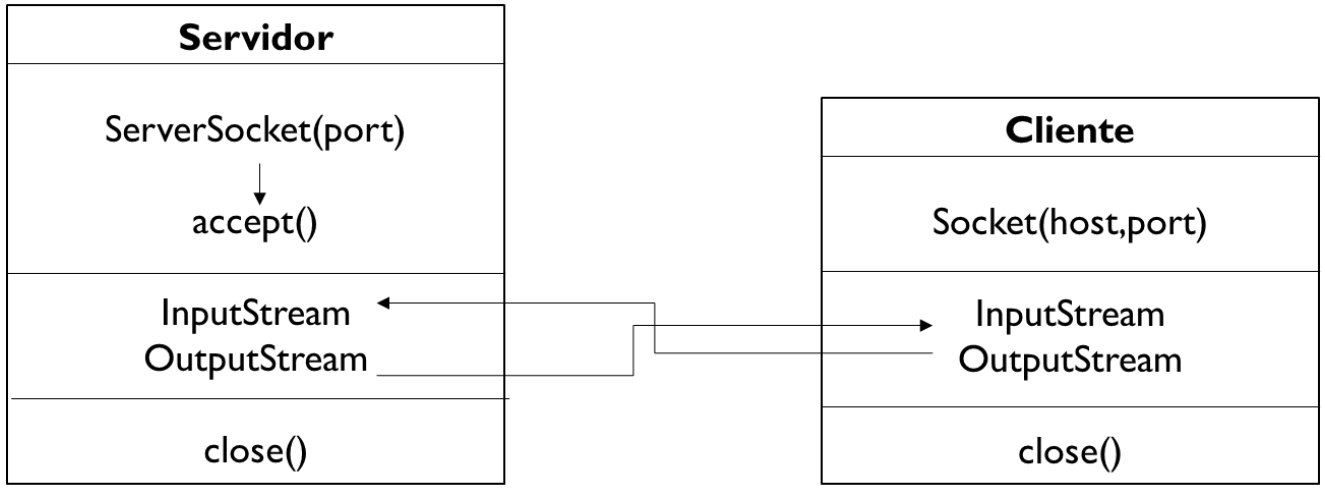



Cliente se conecta al socket especificando su propio puerto
Servidor recibe un nuevo socket con su puerto local y el remoto del cliente y sigue escuchando en el primer socket

Una vez establecida la conexión


Ambos se comunican a través del socket

# STREAMS


Servidor
ServerSocket(port)
accept()
InputStream
OutputStream
close()
Cliente
Socket(host,port)
InputStream
OutputStream
close()

# ECHO

- Ver 2.sockets.echo
- Ejemplo de comunicación cliente/servidor
- Cliente escribe en un socket
- Servidor devuelve lo que recibe (eco)
- Para terminar Ctrl+C, cerramos los Streams y el Socket
- Ejercicio: modificar el servidor de eco para que devuelva los caracteres en orden inverso.