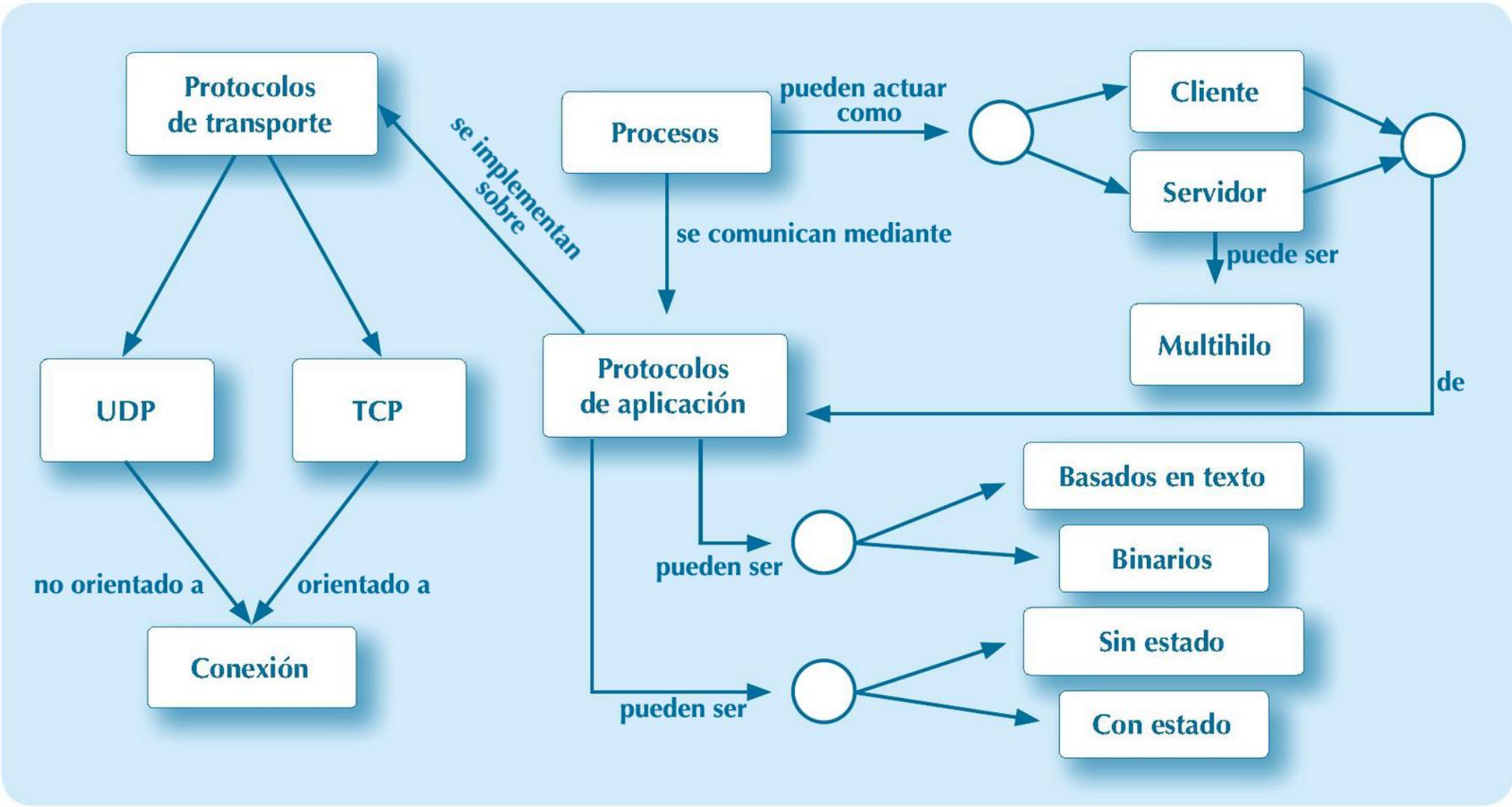
Programación de Servicios y Procesos

UD 3

Programación de Comunicaciones en Red


Protocolos de transporte
se implementan sobre
Procesos
pueden actuar como
Cliente
Servidor
se comunican mediante
puede ser
Multihilo
de
UDP
TCP
Protocolos de aplicación
Basados en texto
Binarios
pueden ser
no orientado a
orientado a
Conexión
Sin estado
Con estado
pueden ser

Modelos de Comunicación entre Procesos

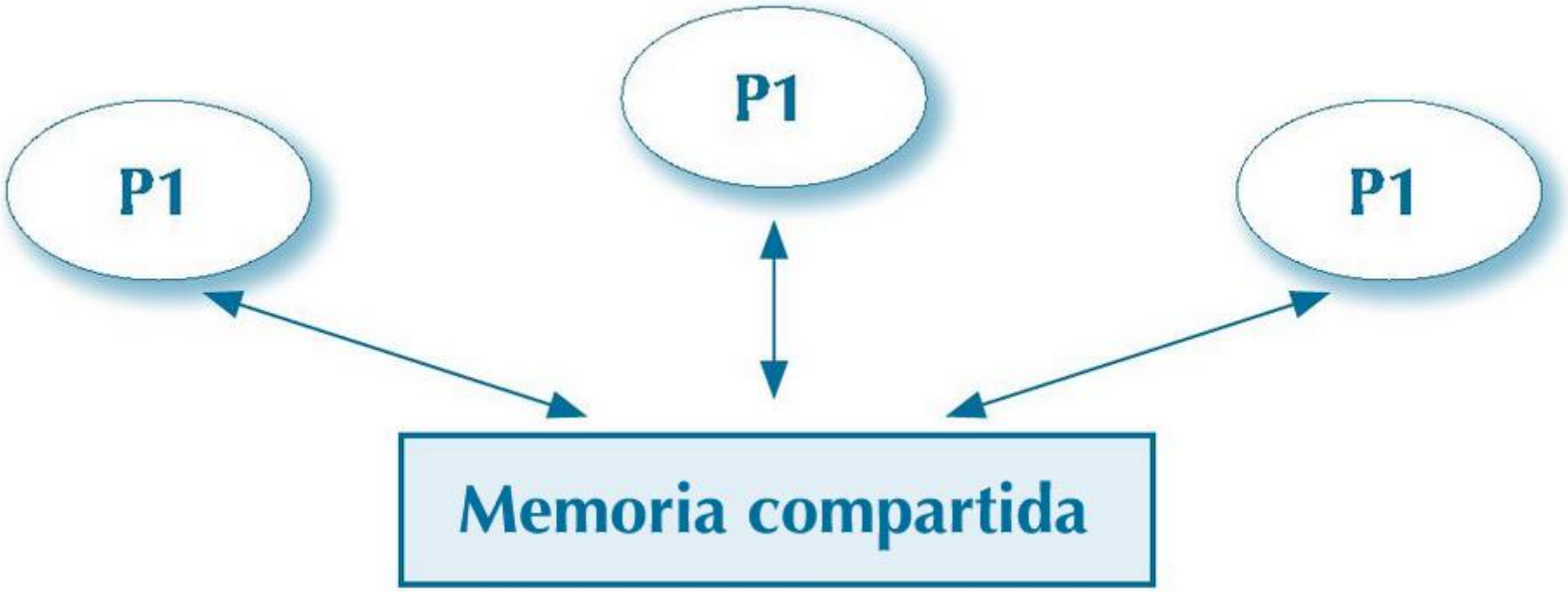
- Procesos que se comunican entre sí mediante mecanismos de E/S secuenciales. De manera que unos procesos pueden leer datos de ficheros o *streams* generados por otros
- Programación de distintos hilos de un mismo proceso que se comunican entre sí mediante memoria compartida. Se trata de una comunicación a más alto nivel y por lo tanto más flexible y natural
- Los programas se ejecutan en distintos procesadores de sistemas independientes, y se comunican a través de una red mediante protocolos estándares de comunicaciones
  - Modelo cliente servidor (C/S)
  - Modelo entre iguales o P2P (peer to peer)

Modelos de Comunicación entre Procesos

- Procesos que se comunican mediante mecanismos de E/S

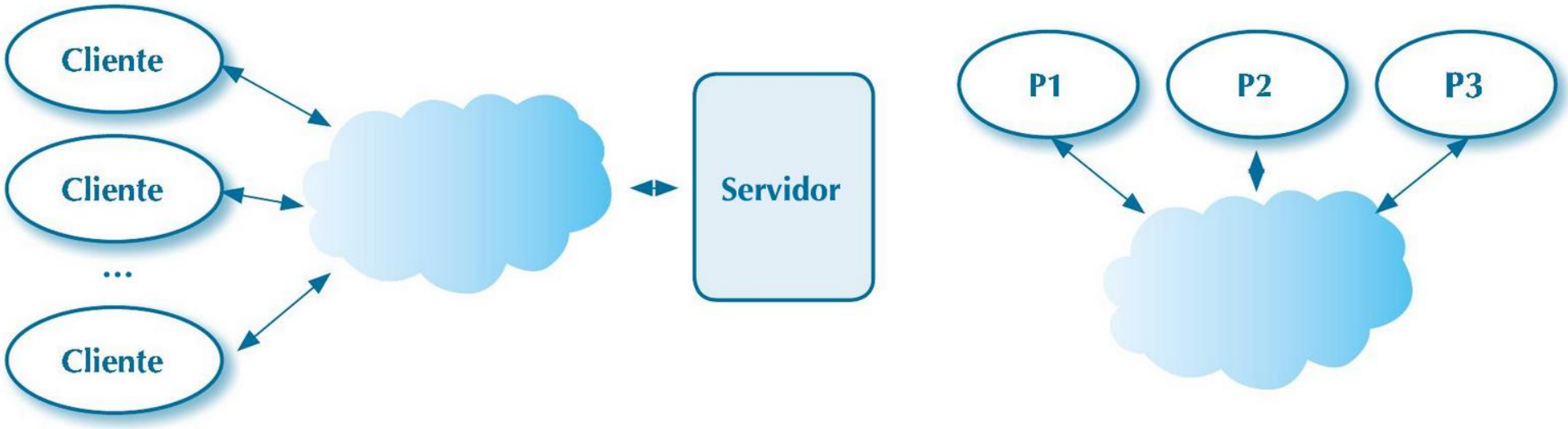
Modelos de Comunicación entre Procesos

- Hilos que se comunican mediante memoria compartida



Modelos de Comunicación entre Procesos

- Aplicaciones concurrentes para sistemas distribuidos



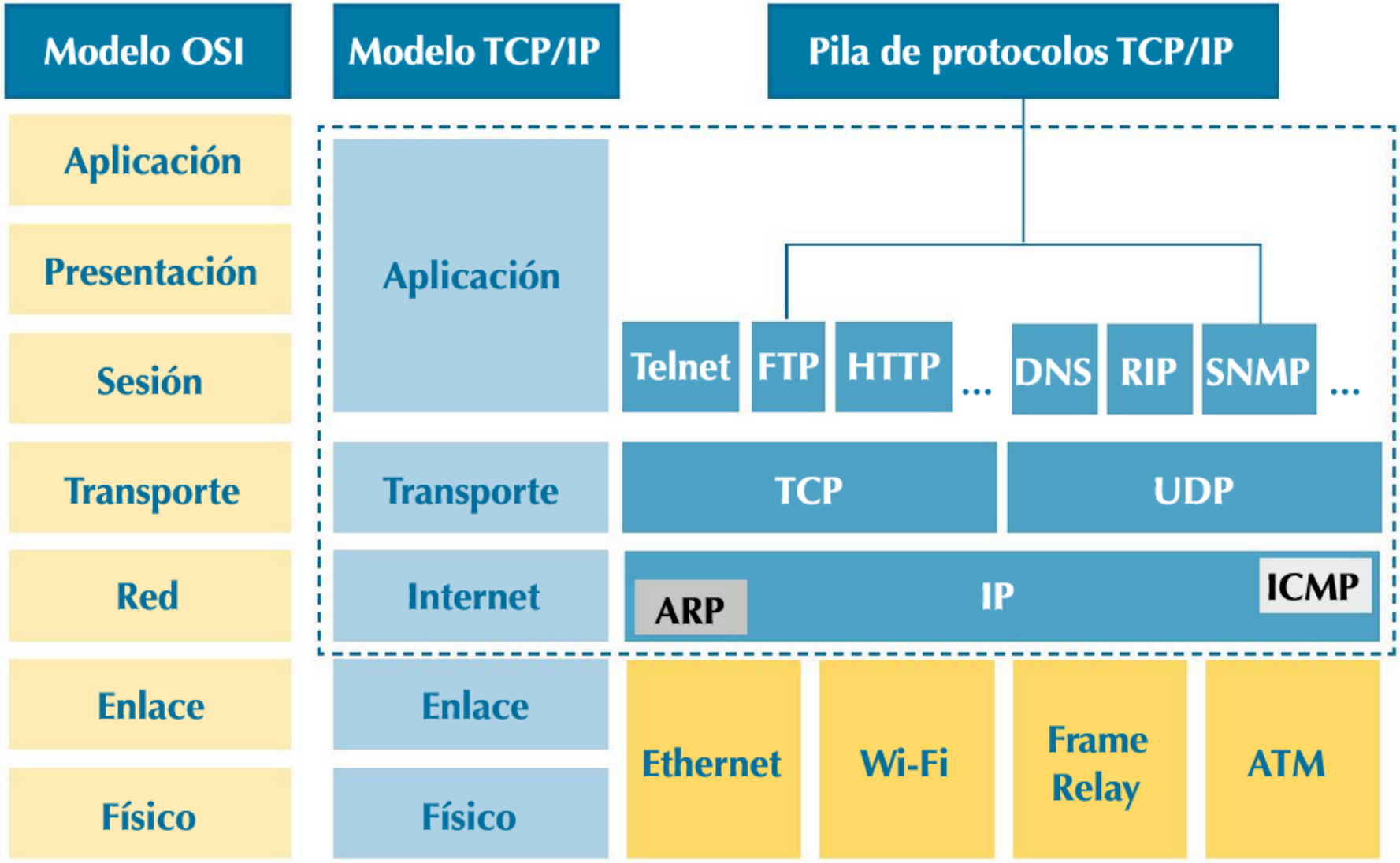
Modelos C/S y entre iguales (P2P) para comunicaciones.entre procesos.

Cambio de Paradigma

“THE NETWORK IS THE COMPUTER”

John Gage (empleado número 21 de Sun Microsystems)

Modelo de Niveles de TCP/IP



Servicios en Cada Nivel de TCP/IP (I)

Nivel	Entidades conectadas	Agrupación de datos	Servicio que proporciona el nivel
Enlace	Interfaces de red	Trama	Comunicación entre dos interfaces o tarjetas de red de una misma red local, con conexión física directa, a través de un *switch* o de varios *switches* interconectados entre sí. El nivel de enlace puede proporcionar un servicio de entrega fiable de tramas sobre un nivel físico no fiable, detectar las tramas con errores y las tramas no recibidas, y solicitar su retransmisión.
Red o internet	Hosts	Datagrama IP (paquete de datos)	Comunicación entre dos interfaces de red, *host interfaces* o hosts. El término host se puede referir a una interfaz de red conectada a una red *(host interface)* o al ordenador al que pertenece la interfaz *(host computer)*. A cada interfaz de red se le asigna una dirección IP. El nivel de red, mediante el protocolo IP, comunica distintas interfaces, mediante un servicio de entrega de paquetes de datos o datagramas, que no garantiza su entrega (pueden no llegar a su destino) ni su entrega en orden (pueden no llegar a su destino en el mismo orden en que se enviaron). Las interfaces pueden pertenecer a una misma red local o a distintas redes locales interconectadas a través de uno o más rúteres o encaminadores intermedios. Un rúter permite conectar redes distintas. Para ello, dispone de varias interfaces de red, y cada una puede estar conectada a una red distinta.

Servicios en Cada Nivel de TCP/IP (II)

Nivel	Entidades conectadas	Agrupación de datos	Servicio que proporciona el nivel
Transporte	Puertos en un host	Segmento	Transmisión de datos entre dos puertos, cada uno en un host. Existen dos protocolos en el nivel de transporte: • TCP. Servicio de transmisión fiable y con entrega de segmentos de datos en destino en el mismo orden en que se envían en origen. • UDP. Servicio de entrega de datagramas de usuario que no garantiza ni su entrega ni su entrega en orden.
Aplicación	Procesos	Datos	Transmisión fiable de una secuencia de bytes entre dos procesos, cada uno de los cuales se ejecuta en un host u ordenador distinto.

Nivel de Enlace

- Comunica interfaces de red conectadas físicamente entre sí mediante un *switch* (o varios)
- Corresponde a una tarjeta física de red
- Se identifica con una dirección MAC de 48 bits (6 bytes) única y fija que viene grabada de fábrica. (`38:2c:4a:b5:72:6b`)

Nivel de Red

- Las tarjetas se comunican a través del protocolo IP
- Dos versiones:
  - IPv4

Tipo	Longitud de máscara	Redes posibles	Rango de direcciones para hosts	Dirección de *broadcast* para una red
A	8	1 red de tipo A: 10.0.0.0/8	10.0.0.1 a 10.255.255.254	10.255.255.255
B	16	16 posibles redes de tipo B: 172.*X*.0.0/16, con 16 <= X <= 31	172.*X*.0.1 a 172.*X*.255.254	172.*X*.255.255
C	24	256 posibles redes de tipo C: 192.168.*X*.0/24, con 0 <= *X* <= 255	192.168.*X*.1 a 192.168.*X*.254	192.168.*X*.255

  - IPv6
    - Reemplaza gradualmente a la IPv4
    - Consta de 128 bits (16 bytes) (`fe80::3a2c:4aff:feb5:726b`)

Comprobar Direcciones de Enlace y Red

- **Windows**: ipconfig /all

- **Linux**: ifconfig

```
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.20  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::3a2c:4aff:feb5:726b  prefixlen 64  scopeid 0x20<link>
        ether 38:2c:4a:b5:72:6b  txqueuelen 1000  (Ethernet)
        RX packets 17568  bytes 11650217 (11.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12440  bytes 1414308 (1.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Nivel de Transporte

- Comunicación host a host entre sockets
  - Socket: Dirección IP + número de puerto
- Dos protocolos
  - **UDP** (*user datagram protocol*):
    - No orientado a conexión
    - No garantiza la entrega de datagramas
    - No garantiza la entraga en orden de los datagramas
    - Tamaño limitado a 1500 bytes (datagramas mayores se fragmentan)
  - **TCP** (*transport control protocol*)
    - Orientado a conexión (flujo de datos entre extremos de una conexión)
    - Garantiza la entrega de datos
    - Garantiza la entrega en orden de los datos

Nivel de Transporten (Rango de puertos)

Puertos del sistema o *well known ports*	0 a 1023	Asignados a determinados servicios o protocolos estándares de nivel de aplicación. Un proceso servidor debe ejecutarse con privilegios de superusuario para recibir comunicaciones desde otros procesos (clientes) por puertos en este rango. Algunos ejemplos de asignaciones de puertos para servicios estándares o protocolos de nivel de aplicación son: – FTP: puertos 22 y 23 de TCP. – HTTP: puerto 80 de TCP o 443 para HTTPS. – DNS: puerto 53 de UDP.
Puertos registrados	1024 a 49 151	Reservados por empresas y organizaciones para sus propios servicios. Por ejemplo, un servidor de bases de datos MariaDB o MySQL acepta peticiones de conexión en el puerto 3306 de TCP.
Puertos efímeros	49 152 a 65 535	Pueden usarlos libremente procesos clientes y servidores. Los programas servidores ofrecen sus servicios en puertos en los anteriores rangos (de sistema o registrados). Los procesos clientes se comunican con ellos desde puertos efímeros. Cuando un proceso servidor con el protocolo TCP recibe una petición de conexión de un proceso cliente, reserva un puerto efímero para la comunicación con ese cliente, confirma la conexión al cliente y le indica ese número de puerto. A partir de ese momento, la comunicación entre los procesos cliente y servidor se lleva a cabo entre ambos puertos efímeros.

Nivel de Aplicación

- Se implementan sobre protocolos de transporte UDP y TCP
  - TCP:
    - Cuando la comunicación ha de ser fiable
    - En transmisiones de datos de gran volumen (web)
  - UDP:
    - Prima la rapidez de transmisión sobre la fiabilidad
    - Cantidades pequeñas de datos
    - No importa el orden de recepción

Resolución de Nombres

- Local: basada en información existente en ficheros locales
  - Linux: `/etc/host`
  - Windows: `C:\Windows\System32\drivers\etc\hosts`
- DNS: basada en protocolo de nivel de aplicación sobre UDP
  - `nslookup`

```
$ nslookup www.wikipedia.org 8.8.4.4
Server:
               8.8.4.4
Address:       8.8.4.4#53

Non-authoritative answer:
www.wikipedia.org       canonical name = dyna.wikimedia.org.
Name:          dyna.wikimedia.org
Address: 91.198.174.192
Name:          dyna.wikimedia.org
Address: 2620:0:862:ed1a::1
```

JAVA.NET – CLASES DE BAJO NIVEL

Interfaces de red	`NetworkInterface:` interfaz de red.
Direcciones IP	`InetAddress:` dirección IP. `Inet4Address:` dirección IPv4. `Inet6Address:` dirección IPv6. `InterfaceAddress:` dirección IP más información relativa a la red a la que pertenece la interfaz.
Sockets	`Socket:` *socket* TCP de cliente, para conectar con un proceso servidor mediante TCP. `ServerSocket:` *socket* TCP de servidor, que acepta conexiones desde un *socket* de cliente mediante TCP. `DatagramSocket:` *socket* de UDP, que se usa para enviar y recibir datagramas de UDP.

NetworkInterface – Métodos

Método	Funcionalidad
`static Enumeration<NetworkInterface> getNetworkInterfaces()`	Devuelve todas las interfaces de red existentes en la máquina. A partir del `Enumeration` se puede obtener un `Iterator` para iterar sobre ellas.
`String getDisplayName()` `String getName()`	Devuelven el nombre de la interfaz.
`byte[] getHardwareAddress()`	Devuelve la dirección MAC en forma de array de 6 bytes (48 bits).
`Enumeration<InetAddress> getInetAddresses()` `List<InterfaceAddress> getInterfaceAddresses()`	Devuelven las direcciones IP asociadas a la interfaz. En general, será al menos una dirección IPv4 y una IPv6. Un objeto `InterfaceAddress` proporciona más información que `InetAddress`. Por ejemplo, la longitud de la máscara de red y, para IPv4, la dirección de *broadcast* parara la red a la que pertenece la interfaz.
`boolean isLoopback()`	Determina si una interfaz es de tipo *loopback*.

InterfaceAddress – Métodos

Método	Funcionalidad
`InetAddress getAddress()`	Devuelve la dirección IP.
`short getNetworkPrefixLength()`	Devuelve la longitud de la máscara de red, es decir, el número de bits dentro de la dirección IP que especifican la red.
`InetAddress getBroadcast()`	Devuelve la dirección de *broadcast* para la red a la que pertenece la dirección IP, pero solo en caso de que se trate de una dirección de IPv4. Si se trata de una dirección de IPv6, devuelve `null`.

INETADDRESS – MÉTODOS

Método	Funcionalidad
`static InetAddress getByName(String host)`	Obtiene una `InetAddress` para el host. En host se puede proporcionar una dirección IP o un nombre de host. En el primer caso, se utiliza el mecanismo de resolución de nombres para obtener una dirección IP. En el segundo, se analiza y valida la dirección IP y se obtiene un objeto de clase `Inet4Address` o `Inet6Address`.
`String getHostAddress()`	Devuelve la representación textual de la dirección IP.
`byte[] getAddress()`	Devuelve un array con los bytes que componen la dirección IP: 4 en el caso de una dirección IPv4, o 16 en el caso de una dirección IPv6.
`static InetAddress getByAddress(byte[] addr)`	Devuelve una `InetAddress`para la dirección IP dada, especificada como un array de bytes.
`boolean isReachable( int timeout)`	Determina si la dirección es alcanzable. Se especifica un tiempo máximo de espera en milisegundos `timeout`.
`boolean isSiteLocalAddress()`	Determina si la dirección es local o privada. Los rangos de direcciones para redes privadas se especifican en la RFC 1918 (https://tools.ietf.org/html/rfc1918). Como ya se ha explicado, las redes privadas de clase C para IPv4, por otra parte las más frecuentes, son 192.168.X.0/24, con 0 <= X <= 255.
`boolean isLoopbackAddress()`	Determina si la dirección IP es de una interfaz de tipo *loopback*.
`static InetAddress getLoopbackAddress()`	Devuelve la dirección de *loopback*. Esta es, como ya se ha visto: – 127.0.0.1 para IPv4. En realidad, según los estándares de internet, es cualquiera de la forma 127/8, es decir, cuyos ocho primeros bits formen el valor 127, es decir, de la forma 127.*.*.*. – ::1 para IPv6.

UDP– Datagrama