

Grado en Ingeniería del Software
Doble Grado en Matemática Computacional e Ingeniería del Software
Doble Grado en Física Computacional e Ingeniería del Software



Redes de Ordenadores

Tema 2

Dr. Constantino Malagón Luque
Dr. Rafael Socas Gutiérrez

Septiembre 2024



2

Nivel de Aplicación

1) Redes de Ordenadores e Internet

2) Nivel de Aplicación

3) Nivel de Transporte

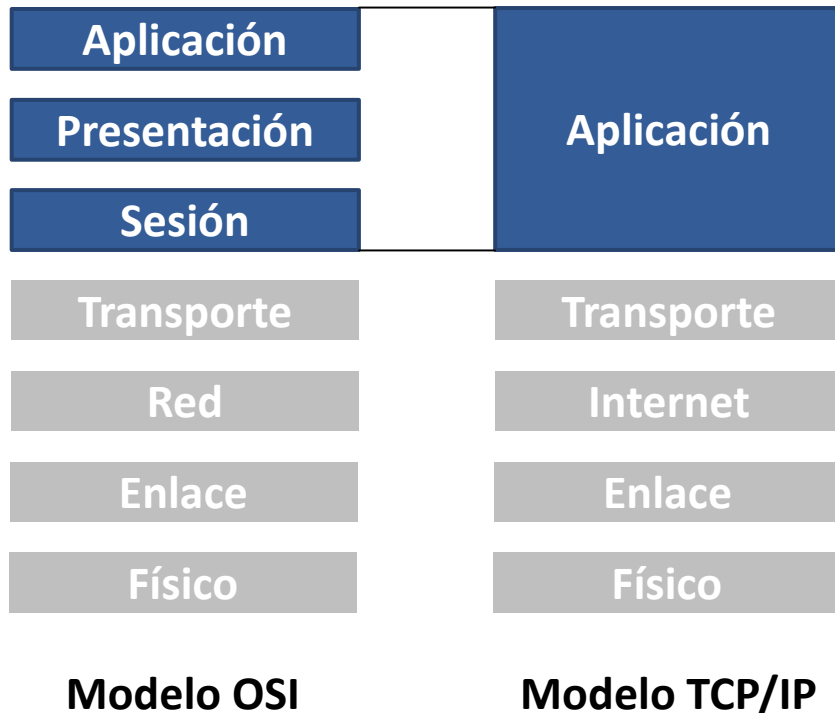
4) Nivel de Red

5) Nivel de Enlace: Redes de Acceso y LAN

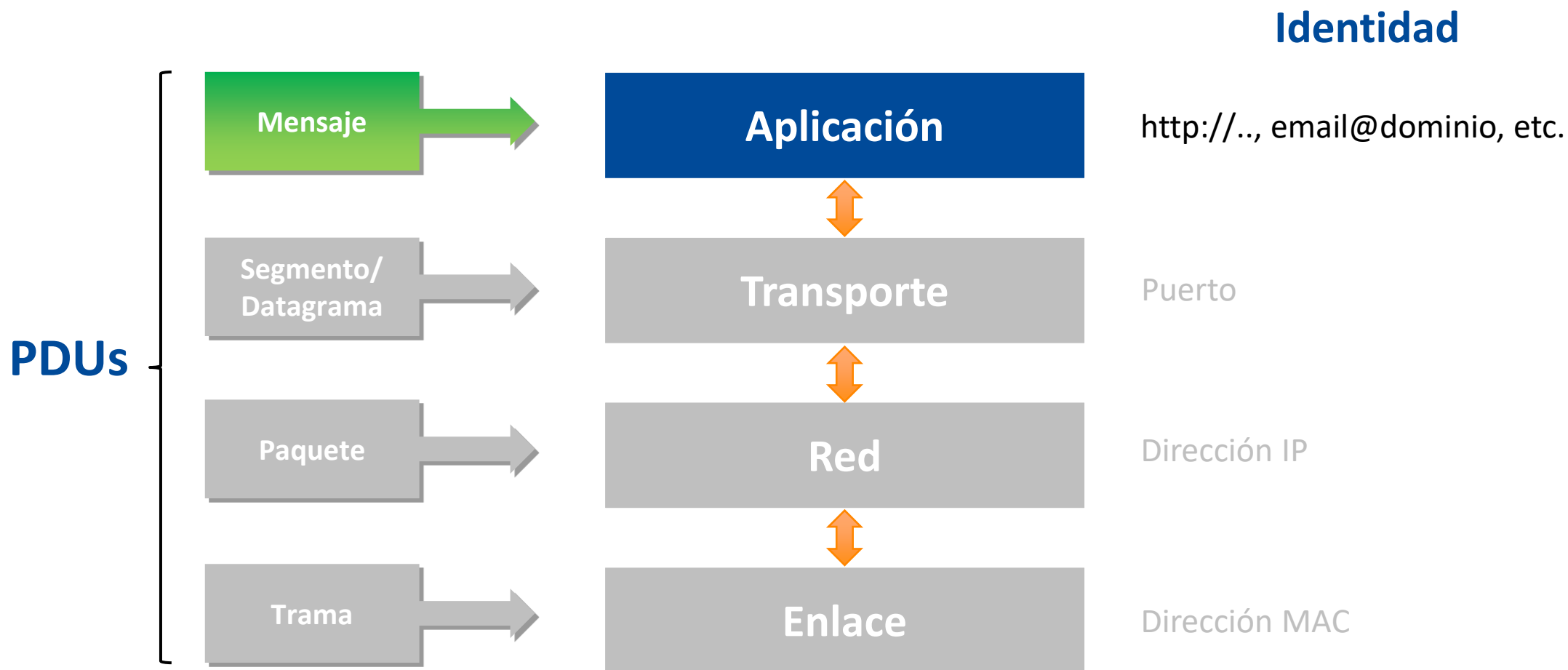
6) Redes Inalámbricas y Redes Móviles

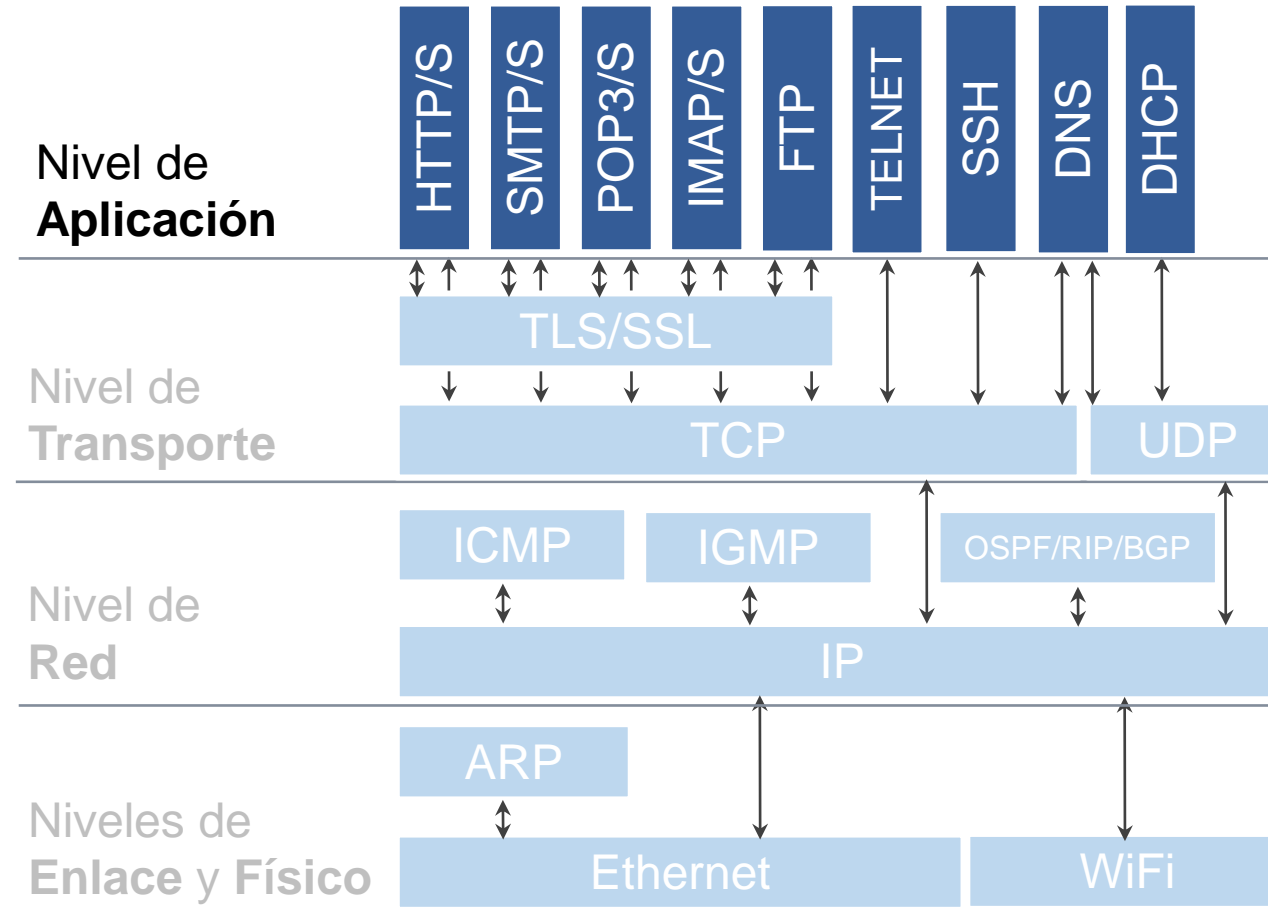
7) Seguridad en Redes de Ordenadores

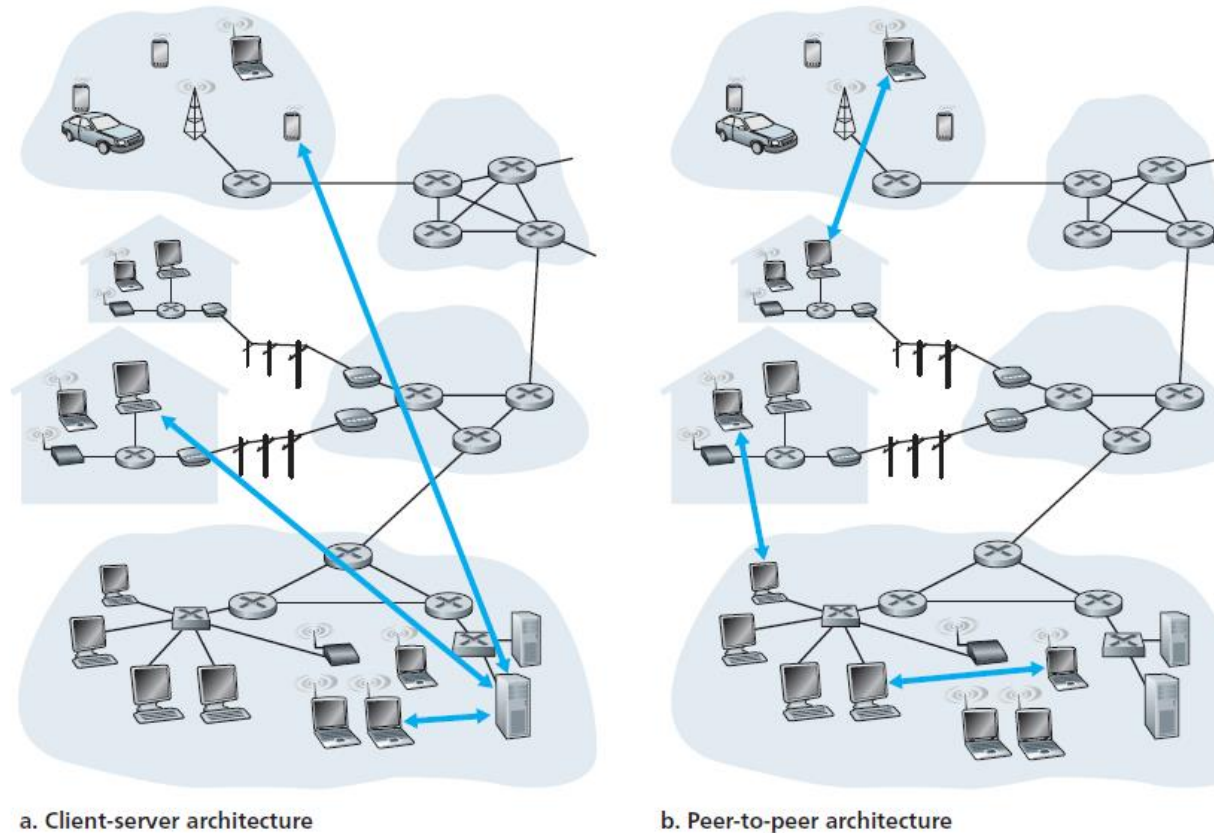
The **application layer** contains the applications with all their individual logic



- **Nivel o capa de Aplicación (Application layer)**
- Los paquetes de información de la capa de aplicación se llaman **mensajes**.
- <Protocolos>:<servicio>
 - HTTP**: publicación de páginas web
 - HTTPS**: publicación de páginas web de forma segura
 - FTP**: transferencia de ficheros
 - Telnet** : acceso remoto
 - SSH**: acceso remoto seguro
 - DNS**: resolución de nombres de Internet
 - DHCP**: asignación de direcciones IP de forma automática
 - SMTP**: correo electrónico saliente
 - IMAP / POP3**: correo electrónico entrante







We mention that some applications have hybrid architectures, combining both **client-server** and **P2P** elements. For example, for many instant messaging applications, servers are used to track the IP addresses of users, but user-to-user messages are sent directly between user hosts (without passing through intermediate servers).

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Lo primero que veremos es nuestra dirección IP en Linux con el comando **ifconfig** o **ip** (ojo, **ipconfig** en Windows)

```
rsocas@Servidor: ~  
rsocas@Servidor:~$ ifconfig  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.1.57 netmask 255.255.255.0 broadcast 192.168.1.255  
    inet6 fe80::ab66:a265:628b:3ad7 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:12:92:0b txqueuelen 1000 (Ethernet)  
    RX packets 2399 bytes 149196 (149.1 KB)  
    RX errors 0 dropped 3 overruns 0 frame 0  
    TX packets 168 bytes 17520 (17.5 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 151 bytes 12445 (12.4 KB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 151 bytes 12445 (12.4 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- ¿Y qué es una dirección IP?
- ¿Qué es una máscara de subred?
- ¿Qué es una dirección MAC?
- Si nuestra dirección IP es 192.168.1.57 con máscara de subred 255.255.255.0 diremos que es la 192.168.1.57/24 en notación CIDR
- Si nuestra dirección IP es 192.168.43.163 con máscara de subred 192.168.43.163/24 en notación CIDR
- ¿Qué es ese /24?

```
root@MVREOR1: /home/rsocas  
root@MVREOR1:/home/rsocas# ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:ac:19:af brd ff:ff:ff:ff:ff:ff  
    inet 192.168.43.163/24 brd 192.168.43.255 scope global dynamic noprefixroute enp0s3  
        valid_lft 3404sec preferred_lft 3404sec  
    inet6 fc::0:0:10::10/64 scope global noprefixroute  
        valid_lft forever preferred_lft forever  
    inet6 fe80::de6:d2ec:c35b:2a87/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
root@MVREOR1:/home/rsocas#
```

Métodos para asignar una IP estática en Linux

- Mediante la orden **ifconfig**
- Mediante la orden **ip**
- Mediante **Interfaz Gráfico**



- Mediante **VirtualBox** nuestro equipo Linux será independiente del equipo host (Windows o Macintosh)
- Máquina virtual en modo **NAT** vs modo **Bridge** (adaptador puente)
- Pondremos nuestra máquina virtual generalmente en **modo Bridge** (no NAT)

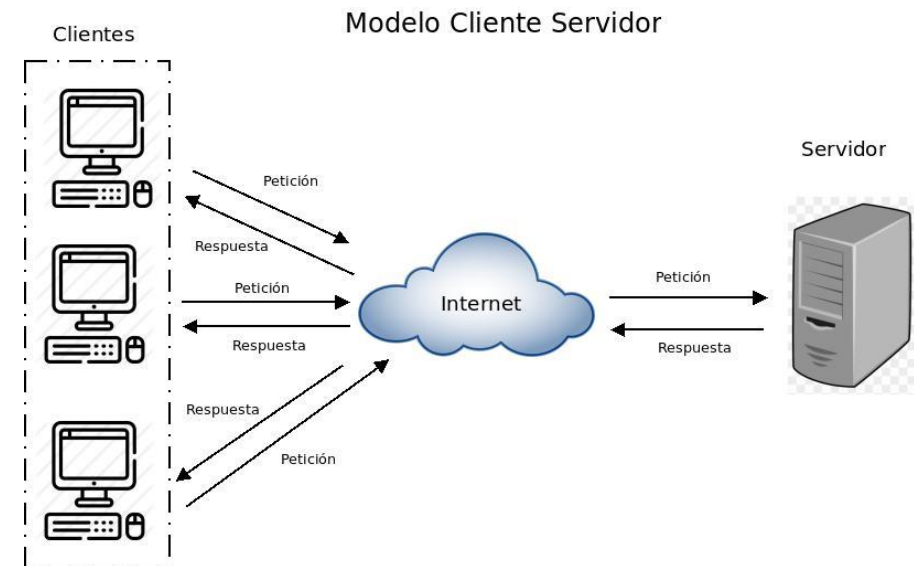
```
root@Servidor: /home/rsocas
Currently scanning: Finished! | Screen View: Unique Hosts
3209 Captured ARP Req/Rep packets, from 4 hosts. Total size: 192540
-----
IP           At MAC Address  Count  Len  MAC Vendor / Hostname
-----
192.168.1.1   78:29:ed:9f:fb:70 3206 192360 ASKEY COMPUTER CORP
192.168.1.39  a0:af:bd:e7:5c:6b 1     60  Intel Corporate
192.168.1.58  08:00:27:03:22:17 1     60  PCS Systemtechnik GmbH
192.168.1.201 8c:61:a3:5a:71:40 1     60  ARRIS Group, Inc.
root@Servidor:/home/rsocas# netdiscover -r 192.168.1.0/24
```

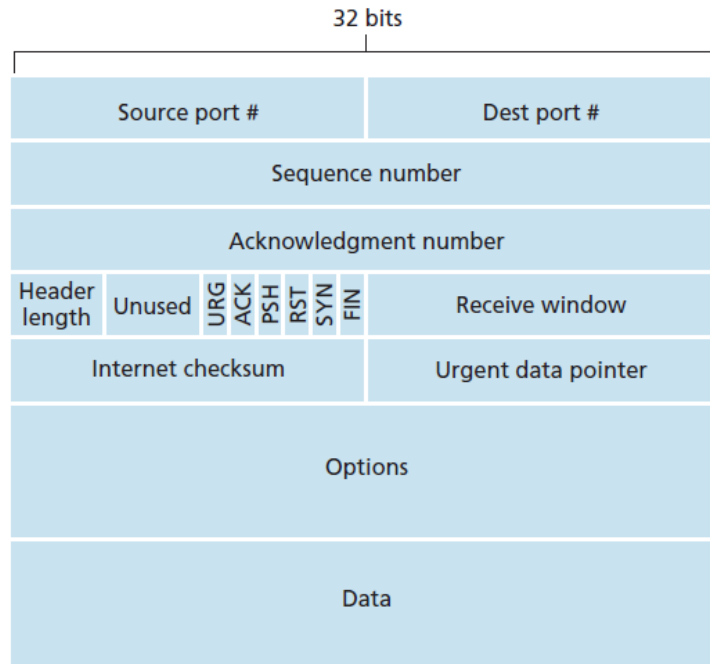
- **netdiscover**
- Vamos a ver qué equipos hay en mi red
- Para ello hay muchas herramientas, pero una de ellas es netdiscover
- **netdiscover -r 192.168.1.0/24**
- Según se vayan conectando irán apareciendo por aquí
- Estos son mis “vecinos”
- ¿Qué necesito para hablar con otros equipos que no estén en mi red?

```
rsocas@Servidor: ~  
rsocas@Servidor:~$ ping 192.168.1.58  
PING 192.168.1.58 (192.168.1.58) 56(84) bytes of data.  
64 bytes from 192.168.1.58: icmp_seq=1 ttl=64 time=0.661 ms  
64 bytes from 192.168.1.58: icmp_seq=2 ttl=64 time=1.02 ms  
64 bytes from 192.168.1.58: icmp_seq=3 ttl=64 time=1.00 ms  
64 bytes from 192.168.1.58: icmp_seq=4 ttl=64 time=0.567 ms  
^C  
--- 192.168.1.58 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3009ms  
rtt min/avg/max/mdev = 0.567/0.812/1.018/0.201 ms  
rsocas@Servidor:~$
```

- El comando **ping** es básico para saber si dos equipos en la red se ven (es decir, si pueden comunicarse entre ellos).
- Con el ping envío un paquete a un equipo con el que quiero comunicarme, y si recibo respuesta es que nos vemos.
- Si no, puede ser por varios motivos (no estamos en la misma red IP y no hay un router por medio, hay un fallo en la tarjeta, hay un firewall que lo está bloqueando,...)
- Ya veremos más adelante cómo es este proceso en profundidad
- Ping también ayuda a estimar el tráfico de la red y la capacidad de cada equipo variando el tamaño del paquete y viendo el tiempo de respuesta.
- **Opciones del ping**
 - c 4 : número de paquetes enviados
 - i 0.01: tiempo entre paquetes (sólo root <0.2)
 - t 200: TTL (por defecto a 64 en el request), cada router disminuye en 1.
- Descripción del comando con **man ping**

- **Servidor:** es un proceso que ofrece un servicio a un posible cliente.
- **Cliente:** proceso que hace uso de los servicios de un servidor.
- Un equipo puede ser cliente o servidor o ambas cosas.
- Por ejemplo, cuando visitamos una página web estamos actuando como clientes de un servidor web.
- Pueden estar en la misma máquina el cliente y el servidor.
- Se necesita un protocolo de transporte de datos:
 - **TCP – Transport Control Protocol.** Protocolo confiable y orientado a la conexión.
 - **UDP – User Datagram Protocol.** Protocolo no confiable pero muy rápido.
- Además, necesitamos un **puerto lógico** (identificador de la aplicación) en el cliente y en el servidor.
- Los puertos y protocolos para los servicios más comunes están listados en https://es.wikipedia.org/wiki/Anexo:Puertos_de_red
- Algunos ejemplos de **puertos conocidos**:
 - TCP 80 – Navegación Web HTTP
 - TCP 443 – Navegación HTTPS
 - TCP 22 – Servicio SSH
 - UDP 53 – Servicio DNS



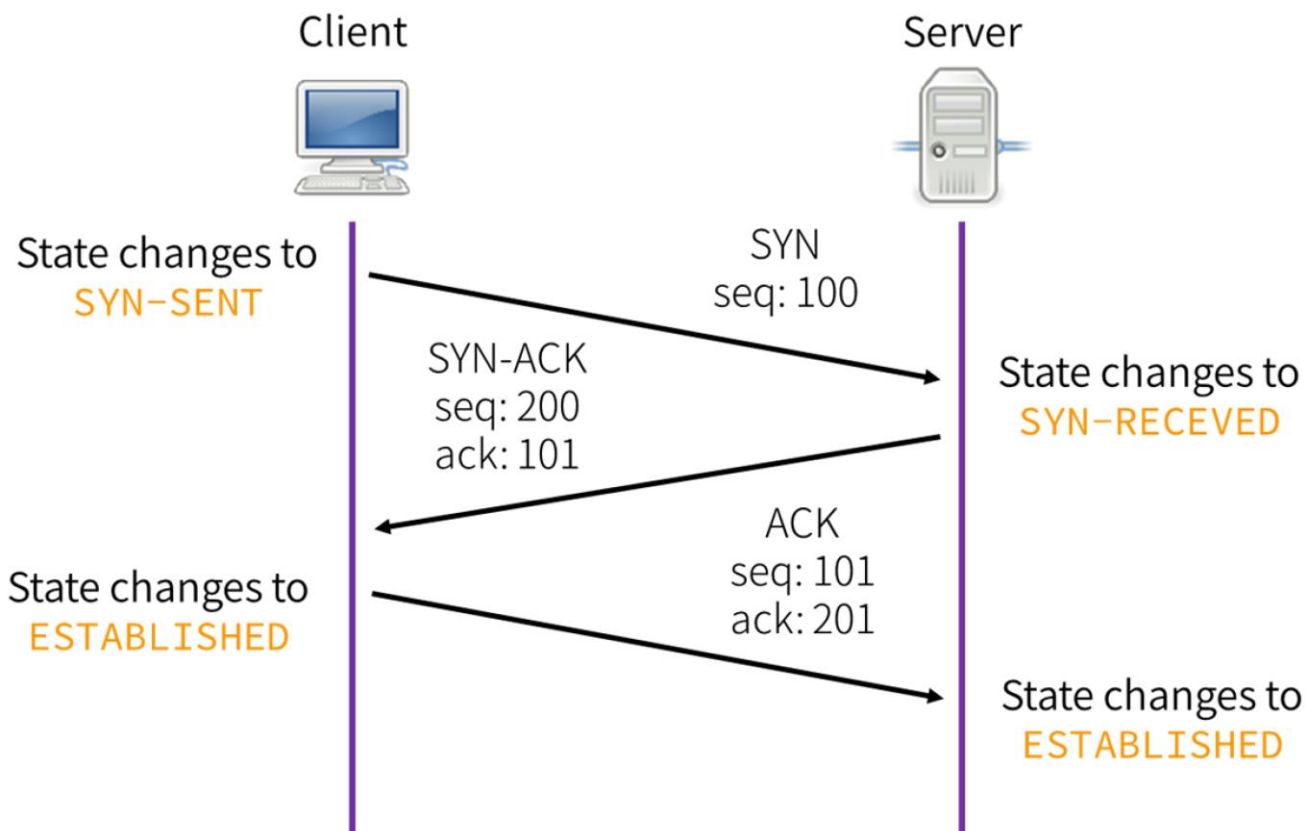


FLAGS

- **URG:** Especifica a la máquina receptora la existencia de información urgente en el flujo de datos.
- **ACK:** Se corresponde con una respuesta de correcta recepción de un paquete anterior que se envió a otra máquina.
- **PSH:** Indica a la máquina receptora que debe pasar la información a la capa de aplicación (programas) lo más rápido posible.
- **RST:** Especifica el reinicio de la conexión entre la máquina receptora y la emisora.
- **SYN:** Se utiliza para la sincronización de números de secuencia entre máquinas.
- **FIN:** Indica que debe empezar el proceso de fin de conexión.

Campos:

- Puerto origen, Puerto destino.
- Número de secuencia o sequence number: se empieza con un número aleatorio. Se va incrementando según el número de bytes enviados.
 - Ej: empieza en 34, se envían 64 bytes, luego el número de secuencia será 98 (34+64).
- Número de secuencia ACK (Acknowledgment number): Se construye con el sequence number recibido incrementado en uno. Se usa como acuse de recibo.
- Por eso se dice que TCP es confiable.
- Nota: El sequence number es efectivamente aleatorio; puede ser cualquier valor entre 0 y 4.294.967.295, incluido.
- Sin embargo, Wireshark por ejemplo, muestra relative sequence y acknowledgment numbers en vez de los valores reales. Estos números son relativos al sequence number inicial de esa conversación.
- Flags (URG, ACK, PSH, RST, SYN, FIN): Fijan la función de cada uno de los mensajes TCP.
- Checksum: Sirve para validar la integridad de los datos.
- Datos: The content sent in the TCP packet.



- **Segmento #1 - Cliente:** envía un paquete sin datos con el flag **SYN** activado (“quiero hablar contigo”).
 - **SEQ = 100** (p.e. empieza en 100).
- **Segmento # 2 - Servidor:** envía un paquete con el flag **ACK** activado (“Recibido”) y el **SYN** también activado (“vale, yo también quiero hablar contigo”).
 - Es decir, es un paquete **SYN+ACK**.
 - **ACK = 101** (es el **SEQ** del cliente + 1, para decir que lo ha recibido correctamente).
 - **SEQ = 200** (el servidor manda un 200 porque es su primer paquete).
- **Segmento # 3 - Cliente:** el servidor está esperando a que el cliente le diga que ha recibido su paquete. Al recibirlo tiene que decirle al servidor que lo ha recibido. Para ello el cliente envía un paquete con flag **ACK** activo.
 - El cliente responde con un **ACK = 201** (para decir que ha recibido el paquete del servidor).
 - Envía un **SEQ = 101** (Es su segundo paquete).
- **Se finaliza la negociación:** “empecemos a hablar”.

Nivel de Aplicación: TCP Ejemplo completo

Time	10.199.3.135	10.199.3.143	Comment
0.000000	42230	80	Seq = 0
0.000088	42230	80	Seq = 0 Ack = 1
0.000882	42230	80	Seq = 1 Ack = 1
0.001386	42230	80	Seq = 1 Ack = 1
0.001500	42230	80	Seq = 1 Ack = 94
0.058322	42230	80	Seq = 1 Ack = 94
0.059142	42230	80	Seq = 94 Ack = 335
0.073985	42230	80	Seq = 335 Ack = 94
0.074661	42230	80	Seq = 94 Ack = 612
0.084469	42230	80	Seq = 612 Ack = 94
0.085168	42230	80	Seq = 94 Ack = 1022
0.097910	42230	80	Seq = 1022 Ack = 94
0.099771	42230	80	Seq = 94 Ack = 1249
0.099831	42230	80	Seq = 1249 Ack = 94
0.100648	42230	80	Seq = 94 Ack = 1355
0.100659	42230	80	Seq = 94 Ack = 1355
0.100698	42230	80	Seq = 1355 Ack = 95
0.101423	42230	80	Seq = 1355 Ack = 95
0.102131	42230	80	Seq = 95 Ack = 1356

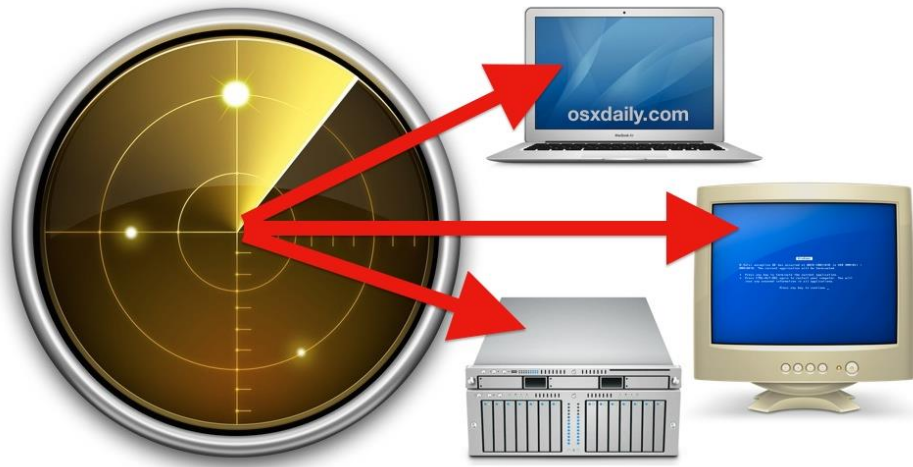
Establecimiento: Three-way handshake

Transferencia de Información Cliente/Servidor

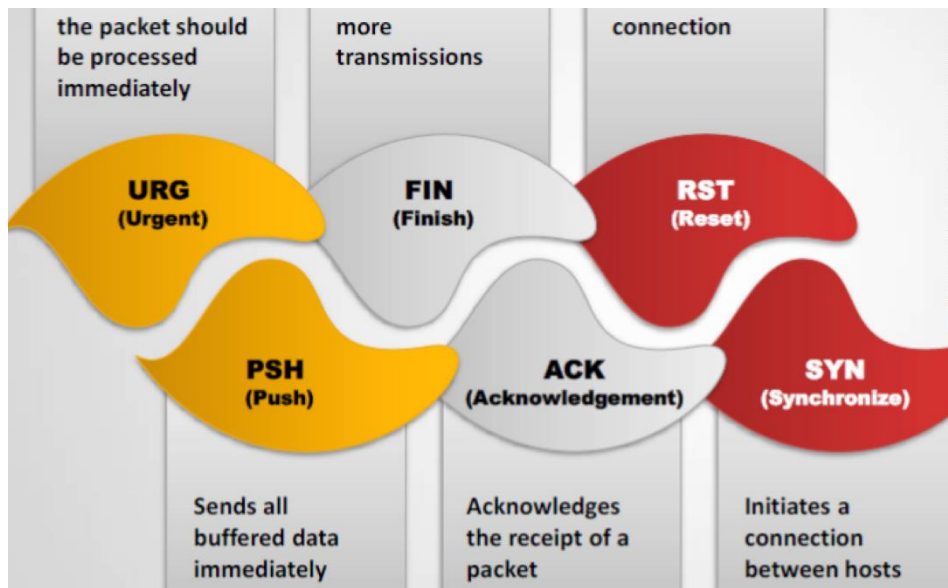
Cierre: Four-way handshake



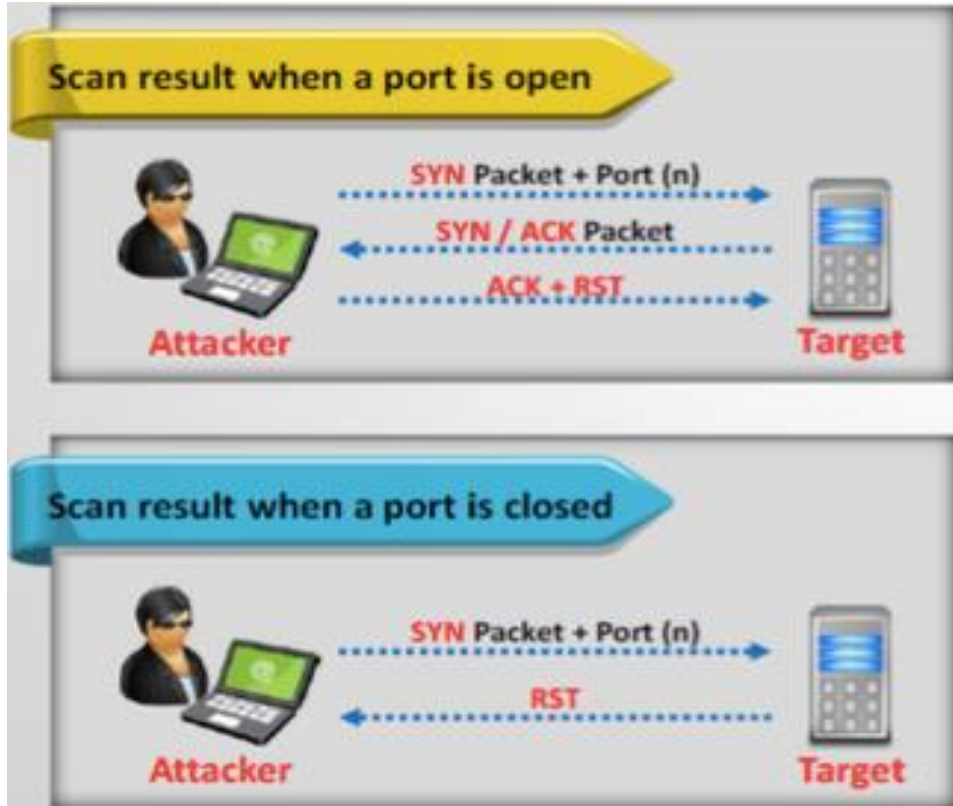
- **nmap**
- Herramienta muy importante para realizar **escaneos de equipos**.
- También hace un **descubrimiento de equipos vivos en la red como hace netdiscover**.
- www.nmap.org
- Hay una interfaz gráfica llamada **zenmap**.
- Primer paso: nmap puede hacer un barrido de pings a una red entera, para ver qué equipos están “vivos”.
 - **nmap -sn 192.168.1.0/24** (notación CIDR)
- Esto realiza un descubrimiento de equipos enviando paquetes ICMP (echo/reply) -> ping sweep, no es un escaneo de puertos, eso lo haremos después, sería como lo que hace el netdiscover.



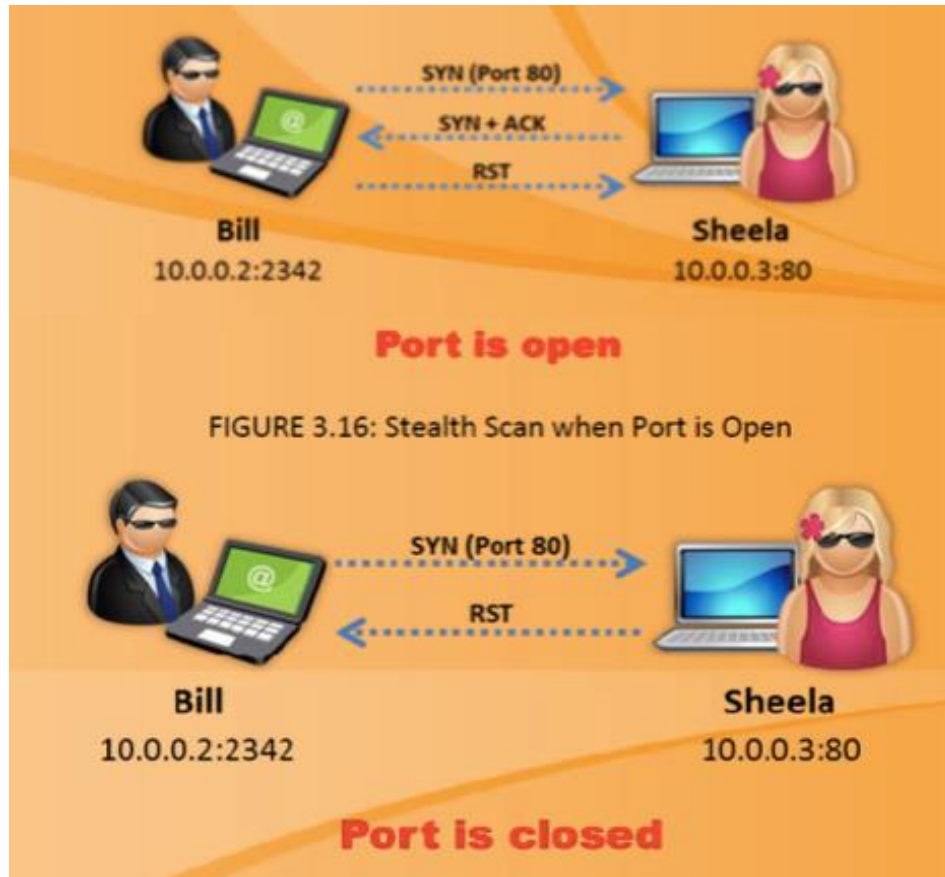
- El **escaneo de puertos** es una de las técnicas más usadas en Seguridad para descubrir servicios que puedan ser comprometidos.
- Un potencial objetivo puede ejecutar muchos servicios que escuchan en puertos conocidos.
- **Escaneando estos puertos, podemos encontrar vulnerabilidades potenciales** (por ejemplo, por bugs conocidos de ese servicio)
- **Hay diferentes técnicas** para escanear puertos, **dependiendo de qué flags enviemos al objetivo**
- ¿En qué consiste entonces un escaneo?:
 - En mandar paquetes sin datos, jugando con esos flags (por ejemplo, el SYN), como si quisiésemos abrir una conexión a un puerto (servicio) que se supone que está abierto.
 - Si me contesta, ya sé que ese puerto está abierto.
- **Según el/o los flags que mandemos tendremos un tipo u otro de escaneo.**
- Y no todos son iguales. Unos dejan más rastro que otros
- ¡Y recordad que esto que estamos haciendo es ilegal! Si es a servidores externos.



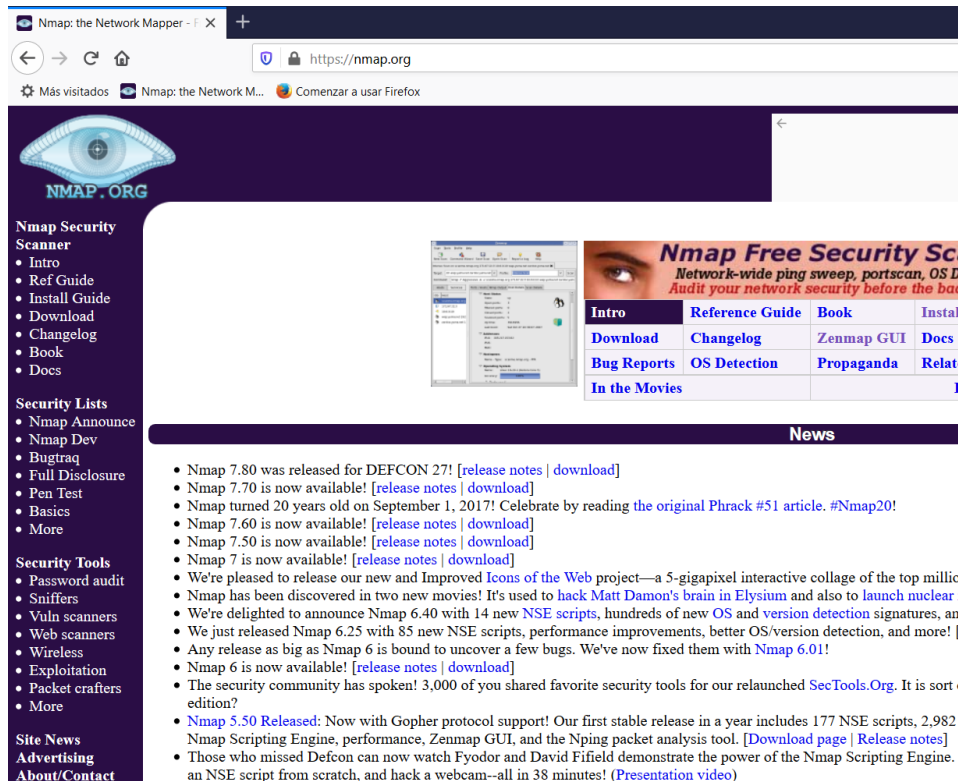
- Por lo tanto, para entender el tipo de escaneo que vamos a hacer es imprescindible conocer bien los posibles flags de los paquetes TCP.
- Hay 6 tipos:
 1. **ACK**: indica que el número de secuencia ACK es válido (acuse de recibo).
 2. **RST**: indica que la comunicación debe reiniciarse (RESET).
 3. **SYN**: gestiona la sincronización (three-way handshake).
 4. **FIN**: indica que queremos terminar la conexión.
 5. **URG**: indica que el paquete transporta datos urgentes.
 6. **PSH**: fuerza el envío de datos recibidos al nivel de aplicación (navegador) de forma inmediata, sin esperar a otros paquetes.
- TCP Flags: PSH y URG:
<https://packetlife.net/blog/2011/mar/2/tcp-flags-psh-and-urg/>



- **Full Open Scan**
- TCP connect (-sT): Se intenta crear una conexión.
 - Es la llamada normal que se hace en cualquier aplicación.
 - Ejecuto el three-way handshake completo.
 - Muy fácilmente detectable (se guarda en el syslog que no se han mandado datos después de la conexión).
 - Es el procedimiento por defecto.



- **Half-open scanning**
- TCP syn (-sS).
 - Se envía un paquete con el flag SYN, el otro envía un SYN+ACK y en lugar de aceptarlo se manda un RST (reset).
 - Es más difícil de detectar, aunque ya casi todos los sistemas de detección de intrusos lo detectan.



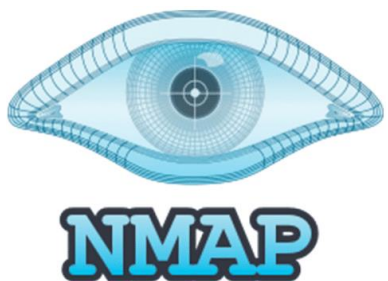
- **UDP connect (-sU):** se intenta crear una conexión mediante el envío de paquetes UDP.
 - DNS, DHCP y SNMP son algunos ejemplos de servicios que utilizan este protocolo.
 - Protocolo no confiable.
 - Con UDP no se produce el three-way handshake.
- Otros:
 - Null scan: paquetes sin flags.
 - Xmas scan: con todos los flags activos.
- El resto de las opciones en la web de nmap.



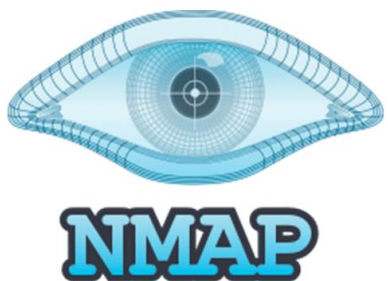
- Partimos de:
 - `nmap -nP -sS 10.10.1.13`
- Puedo detectar el sistema operativo con la opción `-O`.
 - `nmap -nP -sS -O 10.10.1.13`
- Se mandan paquetes y según la respuesta y su comportamiento se determina qué sistema operativo tiene la víctima.
- Se usa por ejemplo null scan para detectar el sistema operativo.



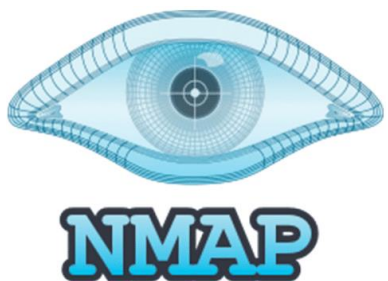
- **Optimización: limitar las IPs escaneadas.**
- Si queremos escanear solo unos equipos de la red a la que estoy conectado, una vez detectados con netdiscover hacer:
 - `nmap -Pn 10.10.0.13-20.`
 - Escanearía desde la 10.10.0.13 hasta el 10.10.0.20.
 - Con la opción `-open` solo se muestran los puertos abiertos, no los filtrados.



- **Optimización: Limitación de puertos escaneados.**
- Puedo escanear sólo uno o varios puertos para que sea más rápido:
 - `nmap -nP -sS -p 139,145 10.10.1.13` (puertos netbios y smb).
 - En un sistema Windows siempre están abiertos y son dos de las puertas de acceso más comunes.
- También `-p 1-400` (rango de puertos).
- Si ponemos `-F` hará un escaneo a los 100 puertos más conocidos.
- Si ponemos `--top-port=10` serían los 10 puertos más importantes para nmap.



- **Optimización: velocidad de escaneo.**
- La opción `-n` es obligatoria porque evita la resolución DNS de la víctima (para saber su nombre DNS), y muchas veces los equipos en una LAN no tienen nombre DNS.
- Si no tardará mucho en hacer esta resolución.
- Si no queremos dejar mucha huella y que no nos pillen, podemos usar `-T1` (escaneo más lento) o `-T5` (escaneo más rápido), pasando por `-T2`, `T3` y `T4`.
- Con `-T5` va muy rápido pero llama más la atención.
- Si quiero ver por dónde va el escaneo puedo pulsar cualquier tecla mientras lo hace y nos lo dirá.



- **Ampliando información**
- Nmap por defecto sólo muestra si tienes o no el puerto abierto, pero no dice nada sobre el servicio que se está ejecutando, su versión, etc.
- Si queremos obtener este tipo de información, usaremos la opción `-sV` (muy utilizado).
- La opción `-sV` incluye el `-O` (detección del sistema operativo).
- Aunque en este caso es mejor utilizar un escáner de vulnerabilidades como Nessus.



- **Ampliando información.**
- Una opción muy usada es `-sC`.
- En este caso, nmap utiliza unos scripts que están desarrollados para ciertos puertos (139,445) para interactuar con el servicio y sacar más información.
- La opción `-a` es como poner `-sV -sC -O`.



- **Intentar evitar que nos pillen.**
- -D 10.0.0.22,10.0.0.25: se usa para que parezca que el escaneo se realiza desde estas IP, y no solo desde la mía.
- La respuesta también les llegará a estos equipos señuelos.
- Los señuelos deben estar activos (verlos en netdiscover).
- Probarlo y verlo con el Wireshark.

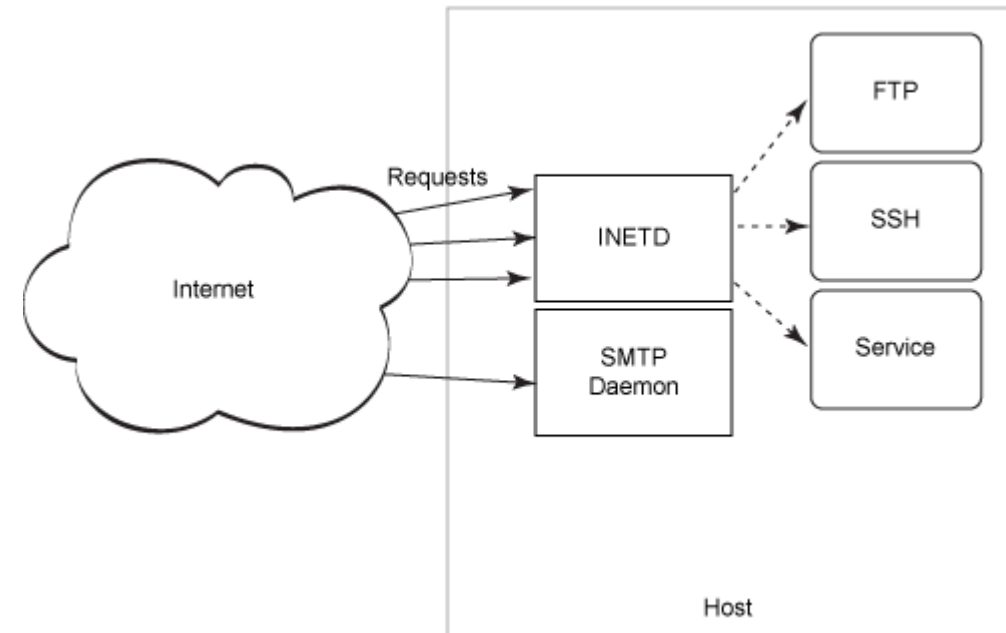
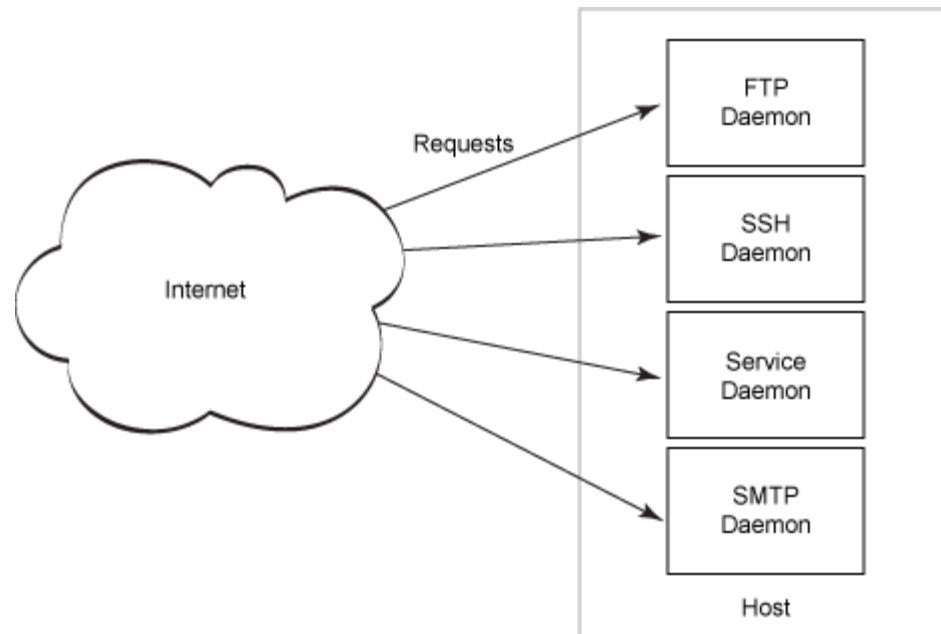


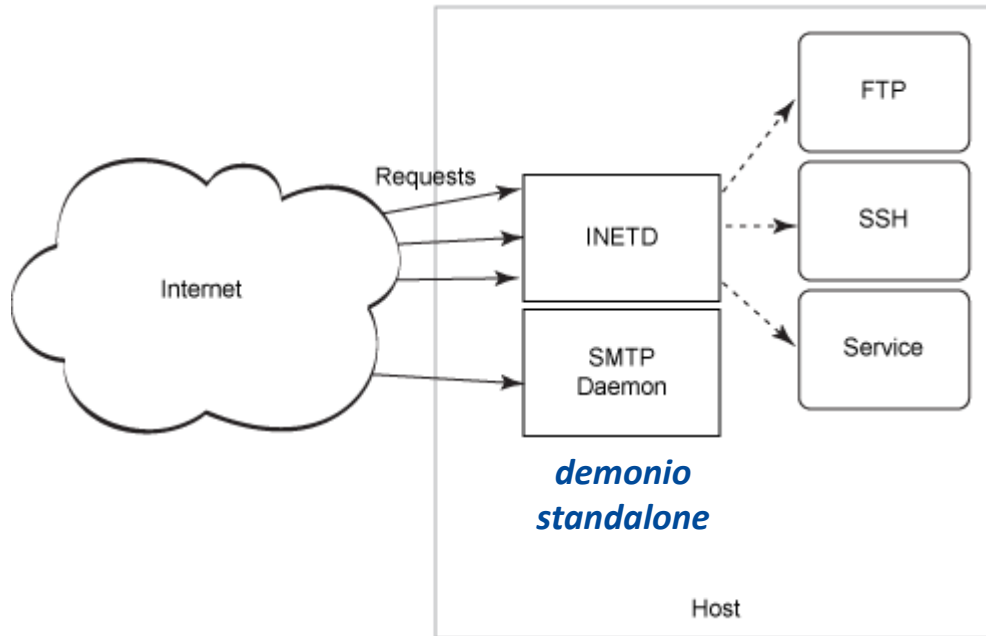
- Si es un servidor web, se puede utilizar un simple telnet para saber qué servidor web está corriendo.
- ¿Qué es telnet?
- Ejemplo:
 - telnet 192.168.1.13 80
- Y siempre nos queda la posibilidad de usar zenmap (la interfaz gráfica de nmap).

Ejemplos de USO

<https://www.tecmint.com/nmap-command-examples/>

Given a list of services, **inetd** watches those services' ports and protocols for requests. When activity occurs, inetd maps the incoming request to standard input (stdin), stdout output (stdout), and standard error (stderr) and launches the proper daemon. The service processes the data and terminates. inetd **keeps resource consumption to a minimum and makes daemons easier to write.**





Existen dos tipos de demonios en Linux:

1. Los demonios dependientes y gestionados por el súper demonio de red o súper servidor. Es este súper demonio el que escucha y arranca los demonios al llegar una petición al puerto correspondiente. Se elegirán en este caso los demonios **ftpd** y **telnetd**, que deberán arrancarse mediante el súper demonio de red **inetd**.
2. Los demonios **standalone**, llamados así porque funcionan de forma independiente del súper demonio de red. Trabajaremos con **SSH**, (Open SSH).

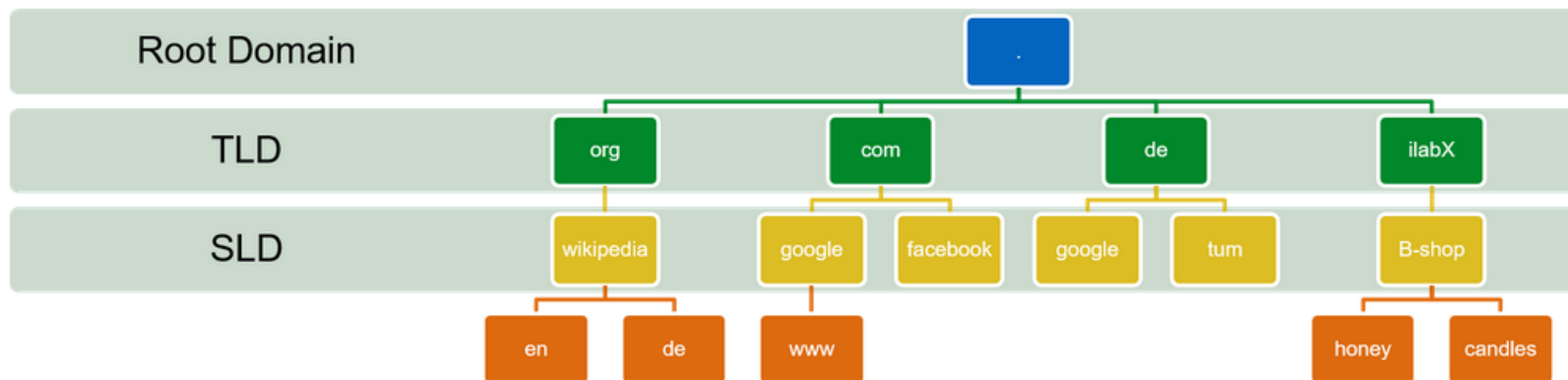
El **Domain Name System** (DNS) consiste en **tres** componentes principales:

1. El **domain namespace**,
2. **Name servers** y
3. **Resolvers**.

- El **domain namespace** es un árbol jerárquico que fija como se construye el espacio de nombres estructurado. Define los **domain names** como por ejemplo www.google.com.
- **Name servers** son los responsables de almacenar información de ciertos namespace. Todos los namespaces que son controlados por un name server se denomina zona. Esta información podría p.e. incluir el mapeo de los nombres de dominio hacia direcciones IP o servidores de nombres responsables de otras partes del namespace.
- Finalmente, los **resolvers** son responsables de consultar servidores para obtener información sobre los espacios de nombres de los que son responsables y proporcionar esta información a los clientes solicitantes.



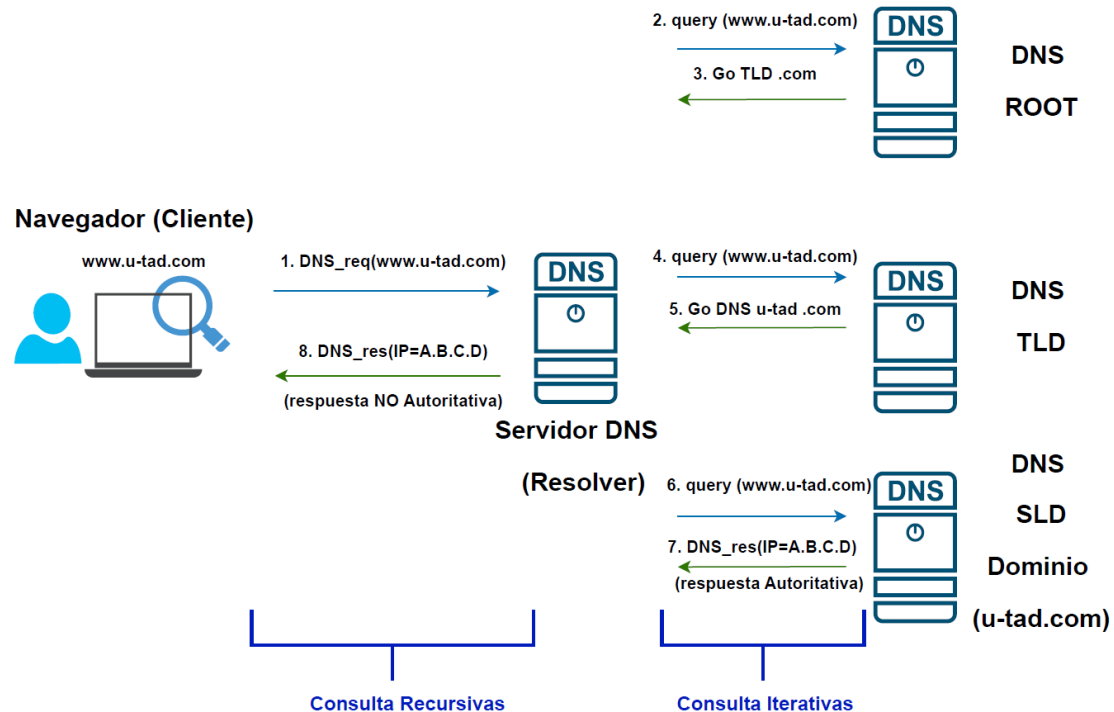
- El **Domain Name System (DNS)** es un sistema descentralizado, que se usa para nombrar los hosts conectados a Internet. El DNS se basa en estructura **jerárquica, similar a un árbol**, denominado **Espacio de Nombres de Dominio**.
- De acuerdo al **RFC920** los dominios son entidades administrativas gestionadas por una sola autoridad. Más adelante discutiremos los servidores de nombres, que pueden ser autoritativos para los dominios. En resumen, una autoridad puede ser, por ejemplo, una empresa que ejecuta la parte del DNS responsable de un subconjunto de dominios (p.e. la compañía google es la autoridad para los dominios *.google.com).
- Un **dominio** se puede identificar por su **Domain Name (Nombre de Dominio)** (p.e. el dominio .com). Cada dominio se puede dividir en subdominios, que son autónomos y pueden ser administrados por diferentes autoridades que su dominio principal. Cada nodo del árbol se llama etiqueta (**label**). Un Domain Name es una secuencia de labels, donde un **Fully Qualified Domain Name (FQDN)** es la secuencia completa de labels desde un nodo en el árbol hasta la raíz (root).



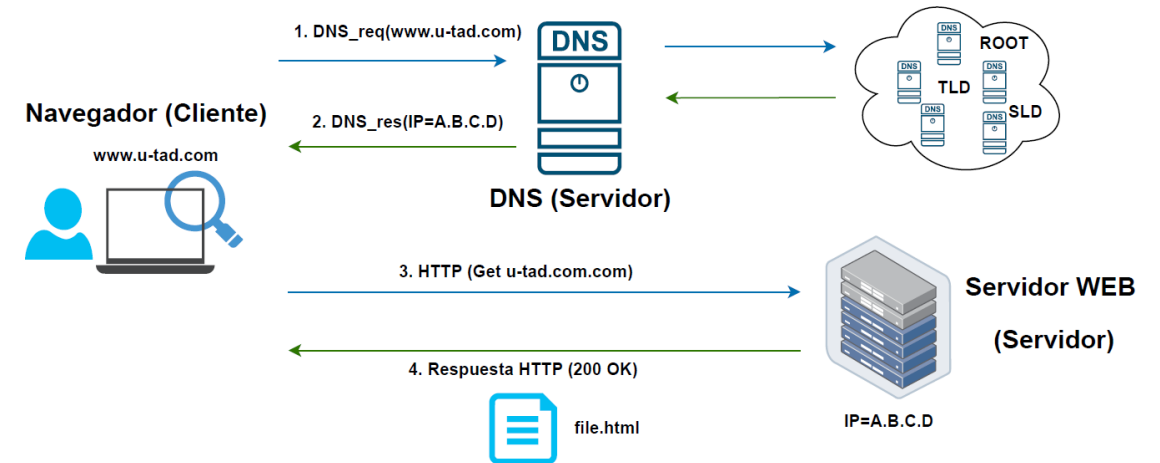
TLD: Top Level Domain

SLD: Second Level Domain

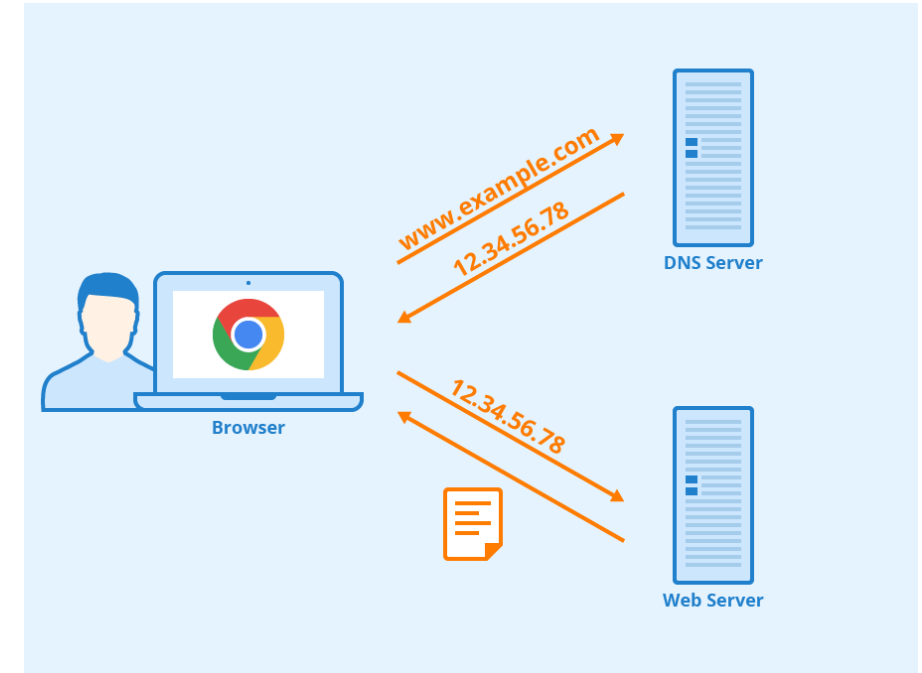
Consulta DNS: Recursiva e Iterativa



Ejem: Navegación WEB con DNS



- Los **nombres de dominio** (domain names) son traducidos a **direcciones IP**. La entidad responsable de este proceso en el servicio DNS se denomina **name servers (NS)**. Igual que otro servidor, un NS tienen un **domain name** y una **dirección IP**.
- Un name server es un servidor DNS responsable para una **zona**. Una zona se puede ver como parte de un dominio. Teniendo en cuenta el dominio de ejemplo vlab.ilabx, podría haber un servidor de nombres responsable de administrar vlab.ilabx y su subdominio www.vlab.ilabx. Pero podría no ser responsable de mooc.vlab.ilabx, ya que este dominio tiene su propio servidor de nombres.
- Una zona se describe en **Zone File** del servidor de nombres, que existe en el servidor de nombres responsable y consiste en un conjunto de **Resource Records (RRs)**. Un RR identifica un recurso en la zona administrada. Hay varios tipos de RRs, p.e. **NS**, **A** or **AAAA**.



En la siguiente tabla se muestran algunos ejemplos de registros de recursos (RRs):

Name	TTL	class	type	data
vlab.ilabx.	3600	IN	NS	ns.vlab.ilabx.
ns.vlab.ilabx.	3600	IN	AAAA	2001:DB8:1234::1:101
www.vlab.ilabx.	3600	IN	A	203.0.113.1
www.vlab.ilabx.	3600	IN	AAAA	2001:DB8:1234::2:102

- **Type NS:** el **nombre es un dominio** (e.g. vlab.ilabx) y el **dato** es el **nombre del servidor DNS autoritativo** que es el responsable de gestionar ese nombre de dominio.
- **Type AAAA:** el **nombre es el nombre de dominio** y el contiene la **dirección IPv6** que mapea este dominio.
- **Type A:** igual que type AAAA, pero para **direcciones IPv4**.
- **TTL:** indica cuánto tiempo otros hosts que consultan estos registros pueden almacenarlos localmente en caché antes de que se actualicen (3600 segundos en este caso).
- **Class:** normalmente se establece en IN (Internet). Existen otras clases, pero rara vez se utilizan y no son relevantes para nosotros.

```
$ORIGIN example.com.
$TTL 86400
@      IN      SOA     dns1.example.com. hostmaster.example.com. (
        2001062501   ; serial
        21600        ; refresh after 6 hours
        3600         ; retry after 1 hour
        604800       ; expire after 1 week
        86400 )      ; minimum TTL of 1 day
;
;
      IN      NS       dns1.example.com.
      IN      NS       dns2.example.com.
dns1   IN      A        10.0.1.1
      IN      AAAA     aaaa:bbbb::1
dns2   IN      A        10.0.1.2
      IN      AAAA     aaaa:bbbb::2
;
;
@      IN      MX       10 mail.example.com.
      IN      MX       20 mail2.example.com.
mail   IN      A        10.0.1.5
      IN      AAAA     aaaa:bbbb::5
mail2  IN      A        10.0.1.6
      IN      AAAA     aaaa:bbbb::6
;
;
; This sample zone file illustrates sharing the same IP addresses
; for multiple services:
;
services IN A        10.0.1.10
      IN AAAA     aaaa:bbbb::10
      IN A        10.0.1.11
      IN AAAA     aaaa:bbbb::11

ftp     IN CNAME     services.example.com.
www     IN CNAME     services.example.com.
;
;
```

Conceptos Generales

- **\$TTL:** ajusta el valor del Time To Live (TTL) predeterminado para la zona. Es el tiempo en segundos que se da a los servidores de nombres para determinar cuánto tiempo los registros de recursos de la zona serán válidos. Un registro de recursos puede contener su propio valor TTL, que tendrá prioridad sobre esta directiva.
- **NS:** registro NS (Name Server), este registro identifica a los servidores DNS que la zona .
- **A:** registro A (Host), este registro identifica cada equipo (su hostname) con su dirección IP.
- **CNAME (Alias):** este registro permite asignar un alias a un determinado host (por ejemplo, permite el alias www).
- **MX (Mail Exchanger) :** este registro identifica al servidor de correo de la zona.

Wireshark · Packet 22 · Wi-Fi


```
> Frame 22: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface \Device\NPF_{0CE9E...}
> Ethernet II, Src: IntelCor_e7:5c:6b (a0:af:bd:e7:5c:6b), Dst: SamsungE_a5:3a:0d (38:2d:e8:a5:3a:0d)
> Internet Protocol Version 4, Src: 192.168.43.210, Dst: 192.168.43.1
> User Datagram Protocol, Src Port: 58811, Dst Port: 53
▼ Domain Name System (query)
  Transaction ID: 0x81ee
  > Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  ▼ Queries
    ▼ www.u-tad.com: type A, class IN
      Name: www.u-tad.com
      [Name Length: 13]
      [Label Count: 3]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      [Response In: 23]
```

CNAME: Alias al nombre de dominio real


Wireshark · Packet 23 · Wi-Fi

```
> Frame 23: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits) on interface \Device\NPF_{0CE9E...}
> Ethernet II, Src: SamsungE_a5:3a:0d (38:2d:e8:a5:3a:0d), Dst: IntelCor_e7:5c:6b (a0:af:bd:e7:5c:6b)
> Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.210
> User Datagram Protocol, Src Port: 53, Dst Port: 58811
▼ Domain Name System (response)
  Transaction ID: 0x81ee
  > Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 2
  Authority RRs: 2
  Additional RRs: 0
  ▼ Queries
    > www.u-tad.com: type A, class IN
  ▼ Answers
    > www.u-tad.com: type CNAME, class IN, cname u-tad.com
    > u-tad.com: type A, class IN, addr 107.6.180.147
  ▼ Authoritative nameservers
    > u-tad.com: type NS, class IN, ns ns49.domaincontrol.com
    > u-tad.com: type NS, class IN, ns ns50.domaincontrol.com
    [Request In: 22]
    [Time: 0.068014000 seconds]
```



 Calle Playa de Liencres, 2 bis
(entrada por calle Rozabella)
Parque Europa Empresarial
Edificio Madrid
28290 Las Rozas, Madrid

 900 373 379  info@u-tad.com

 [SOLICITA MÁS INFORMACIÓN](#)



CENTRO ADSCRITO A:

 **Universidad
Camilo José Cela**

PROYECTO COFINANCIADO POR:

