Capítulo 21

DNS

DNS es el servicio de nombres para redes TCP/IP, es decir, el sistema que asocia nombres únicos a direcciones IP, de forma similar a como una agenda asocia nombres a números de teléfono. Y esto es necesario —o muy conveniente— por varios motivos:

- Las direcciones IP son difíciles de recordar para las personas, especialmente las IPv6.
- Es sencillo asociar significados a los nombres, mientras que las direcciones no lo tienen. Eso facilita la identificación de los servicios que proporcionan determinados nodos (*p. ej.* `mail.example.com` probablemente aloja un servidor de correo).
- Tener nombres permite mantener referencias fijas para los nodos, permitiendo que las direcciones cambien. Eso permite al proveedor migrar el servidor (moverlo a otro nodo) sin que los usuarios, u otros servicios, lo noten[1].
- Un mismo nombre puede asociarse a varias direcciones, lo que proporciona redundancia, y con ello mejora de la disponibilidad, tolerancia a fallos y balanceo de carga.

Formalmente se denomina Sistema de Nombres de Dominio (Domain Name System). Pero ¿qué es un 'dominio'? La definición oficial resulta un tanto... recursiva [37] y nada intuitiva. Por eso aquí lo vamos a enfocar de otro modo.

DNS define un espacio de nombres jerárquico organizado como un árbol invertido, es decir, con la raíz arriba. Cada nodo de este árbol tiene un nombre asociado. El dominio es el subárbol que se obtiene a partir de cualquiera de esos nodos. El *nombre de dominio* para un nodo se obtiene concatenando los nombres de los nodos que van desde ese hacia arriba hasta

[1] Esto se denomina *transparencia de localización*.

la raíz, y por convenio esos nombres se separan con puntos. Por ejemplo, en la Figura 21.1, el dominio `ietf.org` está formado, aparte del propio nodo `ietf`, por los nodos `www`, `mail` y `dev`.

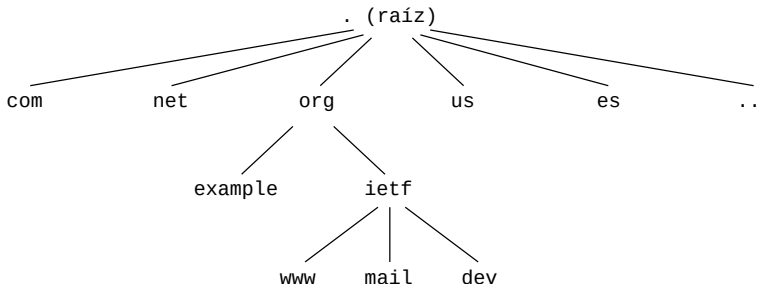

FIGURA 21.1: Estructura jerárquica de los nombres de dominio

Cualquier dominio puede tener subdominios si se toman subárboles a partir del nodo base que lo define. Es fácil entender porqué todo lo relacionado con DNS suena recursivo: el árbol es una estructura recursiva por definición.

Como puede ser confuso, aclaremos que DNS da nombre a varias cosas. Es el nombre del propio servicio distribuido que traduce nombres a direcciones, el del protocolo que define los mensajes que se intercambian los servidores y clientes, y también se utiliza para referirse a los servidores que implementan este servicio: Cuando se habla de «un DNS» se entiende que hablamos de un servidor concreto, que puede ser un proceso o un nodo, en función del contexto.

Conceptualmente, el servicio DNS funciona como una base de datos distribuida y redundante que almacena la información en forma de **registros** nombre–tipo–valor. Esto convierte el protocolo asociado en un mecanismo de consulta. A estas consultas las llamamos en español «resolución de nombres».

La resolución de nombres es una tarea muy frecuente, las aplicaciones lo hacen constantemente como parte de su funcionamiento. Lo hacen cada vez que demandan cualquier servicio de red. Pero también disponemos de herramientas para hacer esas consultas manualmente, y resultan muy útiles para diagnosticar problemas, probar entender la configuración o estudiar su funcionamiento. Una de estas herramientas es `dig`. Aquí puedes ver un ejemplo básico de uso:

```
$ dig www.ietf.org

; <<>> DiG 9.20.15-1~deb13u1-Debian <<>> www.ietf.org
;; global options: +cmd
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12494
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;ietf.org.            IN    A

;; ANSWER SECTION:
www.ietf.org.        113    IN    A    104.16.45.99
www.ietf.org.        113    IN    A    104.16.44.99

;; Query time: 7 msec
;; SERVER: 1.1.1.1#53(1.1.1.1) (UDP)
;; WHEN: Tue Jan 06 14:33:58 UTC 2026
;; MSG SIZE  rcvd: 69
```

Sin embargo `dig` produce mucha información que puede no ser relevante. Por suerte, acepta varias opciones para limitar su salida. Ademas puedes escribir algunas de esas opciones en el archivo `~/.digrc` para que el programa las aplique siempre y así no tener que escribirlas cada vez. Por ejemplo, para este capítulo usamos un archivo `~/.digrc` que contiene lo siguiente:

```
+nocmd
+noedns
+noquestion
+nostats
```

Con eso, si repites el comando anterior, la salida será algo así:

```
$ dig www.ietf.org
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6684
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; ANSWER SECTION:
www.ietf.org.        113    IN    A    104.16.45.99
www.ietf.org.        113    IN    A    104.16.44.99
```

Ahora que la salida es manejable, analicemos su contenido. Este comando en concreto pregunta al servidor DNS configurado en tu sistema acerca del dominio `www.ietf.org`. La respuesta (`ANSWER SECTION`) informa de dos registros tipo A. Estos indican las direcciones IPv4 asociadas: `104.16.44.99` y `104.16.45.99`. Quizá te sorprenda que sean dos y no una, pero eso es perfectamente posible y, de hecho, bastante común. Eso mejora la disponibilidad del servicio, porque el cliente normalmente usará la primera, pero si falla, empleará la segunda.

Además, el servidor puede reordenar esos registros cada vez que se le piden, mejorando el balanceo de la carga entre ambos nodos. Repite el comando varias veces y fíjate en el orden en que aparecen las direcciones cambia en

cada ocasión. Más adelante veremos cómo se saca provecho de esta característica.

Ya que estamos, veamos qué significa la información que aparece en la cabecera:

- `opcode: QUERY`: Se trata de una consulta estándar.
- `status: NOERROR`: La consulta se ha ejecutado sin errores.
- `flags: qr rd ra ad`:
  - `qr`: El mensaje es una respuesta (*query response*).
  - `rd`: El cliente ha solicitado resolución recursiva (*recursion desired*).
  - `ra`: El servidor soporta resolución recursiva (*recursion available*).
  - `ad`: Los datos están autenticados (*authenticated data*).

Antes hemos hablado del «servidor DNS configurado en tu sistema». Hablamos de la dirección de un servidor —en realidad suelen ser dos— que el SO utiliza para realizar las consultas DNS. Normalmente se obtienen automáticamente con DHCP, así que no es algo que nos suela preocupar. En § 21.3 veremos cómo se pueden fijar de forma manual, y por qué puede ser interesante.

21.1. Dominios y zonas

El nombre que acabamos de utilizar —`www.ietf.org`— es efectivamente un dominio, pero también es un subdominio de `ietf.org`, que a su vez es un subdominio de `net`. El nombre `net` es lo que se llama un dominio de nivel superior o TLD (Top Level Domain). Los TLD cuelgan directamente de la raíz del árbol, que se denota con un punto (`.`) al final del nombre, como en `www.ietf.org.` —aunque ese punto final normalmente se omite en los usos menos formales. Un nombre de dominio de este tipo (que termina en el raíz) se denomina FQDN (Fully Qualified Domain Name), que se suele traducir como *nombre de dominio completamente calificado*.

Esta nomenclatura es meramente conceptual y no tiene relación con el modo en que se almacena la información en los servidores. Para cada dominio existe al menos un servidor específico que define la correspondencia oficial entre nombres y direcciones IP para ese domino —si bien suelen ser dos por redundancia. De estos servidores se dice que son los **autoritativos** del dominio. Como ejemplo, veamos cuáles son los servidores autoritativos para el dominio `ietf.org`, pidiendo los registros NS.

```
$ dig ietf.org NS
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45975
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; ANSWER SECTION:
ietf.org.        86400    IN    NS    ken.ns.cloudflare.com.
ietf.org.        86400    IN    NS    jill.ns.cloudflare.com.
```

Miles de servidores (no autoritativos) repartidos por todo el mundo pueden resolver también el nombre `ietf.org`, pero eso es simplemente porque tienen una copia de la información que proporcionan estos servidores autoritativos.

El servidor autoritativo define la llamada *zona* del dominio, que incluye los registros asociados, y responde a consultas específicas sobre esta información. La zona puede incluir la definición de los subdominios o bien, puede delegarlos a servidores autoritativos específicos. Sin embargo, los servidores autoritativos no proporcionan información sobre dominios ajenos, únicamente sobre el que administran.

Veamos los tipos de registro esenciales que se pueden definir en una zona DNS.

Tipo	**Significado**	**Descripción**
SOA	Start of Authority	Indica el servidor autoritativo primario (*master*), correo del administrador, número de serie, etc.
NS	Name Server	Indica los servidores autoritativos de la zona o delega sus subdominios.
A	Address	Asocia un nombre con una dirección IPv4.
CNAME	Canonical Name	Define un alias para otro nombre.

CUADRO 21.1: Registros básicos para definir zonas DNS

Aparte de los NS, que ya hemos visto, el registro más importante es el SOA, que define parámetros esenciales de la zona. Veamos su aspecto real. La Tabla 21.2 detalla el significado de los campos incluidos en el registro.

```
$ dig +multiline ietf.org SOA
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49304
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; ANSWER SECTION:
ietf.org.       1800 IN   SOA jill.ns.cloudflare.com. dns.cloudflare.com. (
                2392675761 ; serial
                10000      ; refresh (2 hours 46 minutes 40 seconds)
                2400       ; retry (40 minutes)
                604800     ; expire (1 week)
                1800       ; negative TTL (30 minutes)
                )
```

Campo	Significado
Primary NS	Servidor autoritativo primario declarado para la zona: `jill.ns.cloudflare.com`
Responsible Mail	Correo del administrador, aunque no siempre tiene formato de correo electrónico.
Serial	Número de serie de la zona.
Refresh (s)	Cada cuanto un secundario comprueba si hay cambios.
Retry (s)	Tiempo de reintento si falla el refresco.
Expire (s)	Tras este tiempo sin contacto, el secundario deja de servir la zona.
Negative TTL (s)	Tiempo de caché para respuestas negativas.

CUADRO 21.2: Valores reales e interpretación del registro SOA de `ietf.org`

21.2. La zona raíz

Como para cualquier otro dominio, el dominio raíz también tiene su propia zona. La zona raíz gestiona los registros NS para todos los TLD. Está coordinada por la ICANN y la IANA, y consta de 13 servidores replicados a centenares por todo el mundo. Están nombrados desde `a.root-servers.net` a `m.root-servers.net`.

Puedes consultar la zona raíz y ver los nombres de esos servidores, y también su SOA:

```
$ dig . NS
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23022
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 0

;; ANSWER SECTION:
.            510508    IN    NS    a.root-servers.net.
.            510508    IN    NS    b.root-servers.net.
.            510508    IN    NS    c.root-servers.net.
.            510508    IN    NS    d.root-servers.net.
.            510508    IN    NS    e.root-servers.net.
.            510508    IN    NS    f.root-servers.net.
.            510508    IN    NS    g.root-servers.net.
.            510508    IN    NS    h.root-servers.net.
.            510508    IN    NS    i.root-servers.net.
.            510508    IN    NS    j.root-servers.net.
.            510508    IN    NS    k.root-servers.net.
.            510508    IN    NS    l.root-servers.net.
.            510508    IN    NS    m.root-servers.net.

$ dig . SOA
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45090
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; ANSWER SECTION:
```

```
.           86351    IN    SOA    a.root-servers.net. nstld.verisign-grs.com.
2026010600 1800 900 604800 86400
```

21.3. Resolución de nombres

En los nodos de los usuarios, al componente que realiza las resoluciones de nombres lo llamaremos *resolvedor* como calco léxico del original en inglés *local resolver* o *stub resolver*. Suele ser de una librería o servicio del SO que recibe y responde peticiones de parte de las aplicaciones a través de la llamada al sistema `getaddrinfo()` o de las librerías que cada lenguaje de programación ofrezca por encima. Por ejemplo, en Python:

```
import socket
for addr in socket.getaddrinfo("www.example.com", None):
    print(addr[4][0])
104.18.5.106
104.18.4.106
```

El módulo `socket` también utiliza automáticamente el resolvedor del SO cuando en lugar de una IP, utilizas un nombre al establecer una conexión TCP o flujo UDP, como en:

```
import socket
s = socket.socket()
s.connect(("www.example.com", 80))
```

El resolvedor determina las fuentes de información que debe usar consultando la línea `hosts` en el archivo `/etc/nsswitch.conf`, que suele tener este aspecto:

```
hosts: files dns
```

Ese «files» indica que primero debe consultar el archivo `/etc/hosts`, que contendrá algo similar a esto:

```
127.0.0.1    localhost
127.0.0.1    maine
::1     ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Si no encuentra el nombre aquí, la palabra «dns» de `nsswitch.conf` indica que debe consultar los servidores DNS configurados en el archivo `resolv.conf`, que suele tener este aspecto:

```
nameserver 1.1.1.1
nameserver 8.8.8.8
```

Estas direcciones IP corresponden a los servidores DNS llamados *recursive resolvers*, y que aquí llamaremos simplemente «servidores DNS» para distinguirlos de los autoritativos. Estos servidores son los encargados de realizar las consultas necesarias para resolver el nombre solicitado por el cliente y suelen proporcionarlos los ISP. Por eso, el contenido de este fichero normalmente se obtiene por DHCP u otros servicios locales como `systemd-resolved` o `resolvconf`.

También existe la posibilidad de editarlo manualmente si ninguno de esos servicios está en uso, o si se quiere forzar un servidor específico ajeno al ISP, como es el caso del ejemplo. La Figura 21.2 muestra cómo hacer dicha configuración en el entorno de escritorio GNOME.

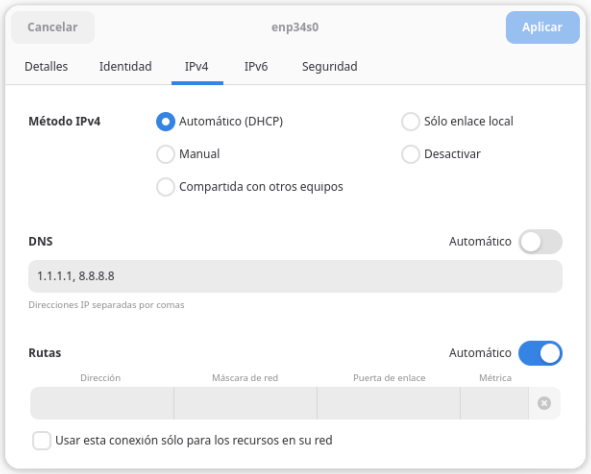

FIGURA 21.2: Configuración DNS en GNOME

La razón por la que se establecen dos de estos servidores es la habitual: redundancia. Si el primero no responde u ocasionalmente lo hace muy lento, el SO puede puede utilizar el segundo.

Normalmente el cliente pide a estos servidores una *resolución recursiva*, que implica los siguientes pasos del servidor DNS:

1. Si el nombre solicitado está en su caché, responde inmediatamente al cliente.
2. Si no está en la caché, realiza una consulta a un servidor raíz, que le indica el servidor autoritativo para el TLD solicitado.

3. Pregunta al servidor autoritativo del TLD, que a su vez delega la consulta al servidor autoritativo del dominio solicitado.
4. Realiza una consulta al servidor autoritativo específico, que puede disponer de la dirección o bien responder con otra delegación: el registro NS de un subdominio.
5. Guarda la respuesta en caché y la devuelve al cliente.

Es *recursivo* porque el servidor (y no el cliente) se encarga de seguir las delegaciones, es decir, realiza las consultas necesarias hasta llegar hasta la respuesta final. Por defecto, el cliente solicita resoluciones recursivas al servidor (RD=1 en la cabecera del mensaje DNS), pero opcionalmente es posible hacer solicitudes iterativas (RD=0). En ese caso, el servidor responde con la información de que dispone, pero no hace peticiones adicionales. Lo importante es entender que el funcionamiento típico del servidor DNS es ofrecer resolución recursiva realizando peticiones iterativas.

Para entender mejor el trabajo que realiza el servidor, podemos ver un proceso similar forzando una resolución iterativa desde el cliente:

```
$ dig +trace www.ietf.org
.            513823    IN    NS    a.root-servers.net.
.            513823    IN    NS    b.root-servers.net.
.            513823    IN    NS    c.root-servers.net.
.            513823    IN    NS    d.root-servers.net.
.            513823    IN    NS    e.root-servers.net.
.            513823    IN    NS    f.root-servers.net.
.            513823    IN    NS    g.root-servers.net.
.            513823    IN    NS    h.root-servers.net.
.            513823    IN    NS    i.root-servers.net.
.            513823    IN    NS    j.root-servers.net.
.            513823    IN    NS    k.root-servers.net.
.            513823    IN    NS    l.root-servers.net.
.            513823    IN    NS    m.root-servers.net.
;; Received 525 bytes from 1.1.1.1#53(1.1.1.1) in 7 ms

org.            172800    IN    NS    a0.org.afilias-nst.info.
org.            172800    IN    NS    a2.org.afilias-nst.info.
org.            172800    IN    NS    b0.org.afilias-nst.org.
org.            172800    IN    NS    b2.org.afilias-nst.org.
org.            172800    IN    NS    c0.org.afilias-nst.info.
org.            172800    IN    NS    d0.org.afilias-nst.org.
;; Received 812 bytes from 2001:7fe::53#53(i.root-servers.net) in 191 ms

ietf.org.        3600    IN    NS    ken.ns.cloudflare.com.
ietf.org.        3600    IN    NS    jill.ns.cloudflare.com.
;; Received 306 bytes from 199.19.54.1#53(b0.org.afilias-nst.org) in 19 ms

www.ietf.org.        300    IN    A    104.16.45.99
www.ietf.org.        300    IN    A    104.16.44.99
;; Received 177 bytes from 173.245.58.122#53(jill.ns.cloudflare.com) in 3 ms
```

Este comando se ha ejecutado teniendo configurado `1.1.1.1` en el archivo `/etc/resolv.conf`. Veamos las diferentes partes que la componen, que son las respuestas a las consultas sucesivas que hace el cliente a partir de la información que va recibiendo de cada servidor:

- Consulta al servidor raíz (**línea 2**). Responde con los 13 servidores autoritativos para el dominio raíz.
- Consulta a `i.root-servers.net` acerca del TLD `org`, que responde con 6 servidores autoritativos (**línea 17**).
- Consulta a `b0.org.afilias-nst.org` sobre `ietf.org` (**línea 25**). Responde con dos servidores autoritativos.
- Consulta a `jill.ns.cloudflare.com` para resolver `www.ietf.org`, que finalmente responde con las direcciones IP solicitadas (**línea 29**).

Después de cada sección aparece el servidor concreto al que se ha consultado y cuanto tiempo tardó en responder.

21.4. Cachés e información obsoleta

Las cachés de los servidores DNS juegan un papel clave en el funcionamiento del sistema, proporcionando un equilibrio entre disponibilidad de información actualizada y rendimiento. Cuando un servidor recibe una petición desde un cliente, primero comprueba su caché. Si tiene la respuesta y aún es válida (no ha expirado), devuelve inmediatamente ese valor y no realiza consultas adicionales. Sin la caché, los servidores continuamente tendrían que hacer consultas a los mismos servidores autoritativos, lo que conllevaría una sobrecarga inadminible para todas las partes y degradaría de forma muy notable la operación del sistema en su conjunto.

La invalidación de la caché la determina el campo TTL (expresado en segundos) asociado a cada registro. El servidor autoritativo define el valor del TTL por lo que no cambia, a menos que el administrador de la zona lo modifique manualmente. Puedes averiguar el valor del TTL tal como está definido dirigiendo la consulta directamente al servidor autoritativo (usando la `@`) obviando el servidor configurado en tu sistema.

```
$ dig @jill.ns.cloudflare.com ietf.org
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11996
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; ANSWER SECTION:
ietf.org.        300    IN    A    104.16.45.99
ietf.org.        300    IN    A    104.16.44.99
```

Hay dos pistas en esta respuesta que confirman que hemos preguntado al servidor autoritativo:

- En la cabecera aparece el flag `aa` (*authoritative answer*).
- La advertencia indicando *recursion requested but not available*, algo que ya hemos explicado es lo esperable para servidores autoritativos.

El TTL es el valor que aparece en la segunda columna junto a cada registro (300 segundos en este caso). Si repites esta misma consulta verás que siempre aparece el mismo valor, mientras que si envías la consulta al servidor proporcionado por el ISP, el valor irá disminuyendo hasta llegar a cero, momento en el que hará una nueva consulta al autoritativo y volverá a empezar desde 300. Este sistema tiene una consecuencia importante: la información que ofrecen los servidores siempre está potencialmente obsoleta[2]. ¿Cuánto? Depende completamente del valor del TTL definido en cada zona, si bien es cierto que muchos ISP imponen un valor mínimo (5 minutos) y un valor máximo (1 día).

Y así es cómo se distribuye la información a través de la red DNS. No hay ningún mecanismo que detecte cambios y sea proactivo en la distribución de actualizaciones. Cuando por ejemplo el administrador de una zona modifica la dirección IP asociada a un nombre, los servidores seguirán ofreciendo información incorrecta a sus clientes hasta que sus cachés expiren.

Es común escuchar que las asociaciones DNS «tardan en propagarse», pero en realidad solo depende de la configuración del TTL del servidor autoritativo y de los servidores de terceros que estén involucrados. Por ese motivo, cuando se realiza una migración que implica un cambio de dirección IP es buena práctica reducir el TTL horas o días antes, dando tiempo para que expiren las cachés en toda la red. Una vez terminada la migración, puede volver a aumentarse el TTL para evitar consultas demasiado frecuentes.

La información negativa, es decir, la situación en la que un servidor autoritativo dice que no conoce un nombre, también se almacena en caché. En ese caso se utiliza el valor llamado *Negative TTL* que aparece en el registro SOA (ver Cuadro 21.2). Así se evitan consultas reiteradas para nombres desconocidos a los mismos servidores.

[2]Se dice que ofrece *consistencia eventual*

21.5. Configuración de zona

La configuración de una zona en un servidor autoritativo se especifica en un archivo llamado *fichero de zona*[3] que contiene los registros que la definen.

El siguiente listado muestra una definición de zona hipotética para el dominio `example.net`[4]. Este fichero de zona se aplicaría en los servidores autoritativos y para responder a las consultas sobre ese dominio.

```
$TTL 3600
@ IN SOA ns1.example.net. ns2.example.net. (
    2026012901 ; serial
    3600       ; refresh
    900        ; retry
    604800     ; expire
    300        ; negative TTL
)

@   IN NS    ns1.example.net.
@   IN NS    ns2.example.net.

@   IN A     192.0.2.10
@   IN AAAA  2001:db8::10

www IN CNAME @
@   IN MX 10 mail.example.net.
_sip._tcp IN SRV 10 60 5060 sip.example.net.
```

Fíjate que ambos servidores autoritativos sirven la misma información, ambos definen el registro SOA que apunta al servidor primario (`ns1`). Ambos incluyen los dos registros NS.

Esta zona define también:

- El valor por defecto de TTL para los registros que no lo especifiquen explícitamente (3600 segundos).
- Registro A para el dominio base[5] `example.net` (representado por '`@`').
- Registro AAAA para el dominio base, que proporciona una dirección IPv6 asociada.
- Registro CNAME que define un alias `www.example.net` para el dominio base, es decir, devuelve la misma dirección IP.
- El registro MX que define el servidor de correo para el dominio.
- El registro SRV que define el servicio SIP sobre TCP para el dominio.

[3] O alguna estructura de datos equivalente.

[4] El dominio `example.net` existe realmente con otra configuración

[5] También se denomina en inglés *apex domain* o *root domain*, pero evitaremos esta segunda opción porque se puede confundir con el dominio raíz: '`.`'

Para que esta zona sea legítima y se pueda utilizar, la zona del `net.` debe definir obligatoriamente los mismos valores para los registros NS. Los registros correspondientes en el fichero de zona de `net.` serían algo así:

```
example.net.      3600  IN  NS    ns1.example.net.
example.net.      3600  IN  NS    ns2.example.net.
ns1.example.net.        IN  A     192.0.2.53
ns2.example.net.        IN  A     192.0.2.54
```

Los registro NS representan aquí la delegación del dominio `example.net` a los servidores autoritativos indicados (`ns1` y `ns2`). Los registros A adicionales se denominan *registros glue* y son necesarios para evitar un posible bucle de resolución cuando el nombre del servidor autoritativo (como en este caso) está dentro del dominio delegado.

21.6. Resolución inversa

ToDo!

21.7. Replicación

Algunas compañías ofrecen servidores DNS públicos que proporcionan algunas ventajas sobre los que facilitan los ISP. Esta tabla recoge los más conocidos, sus direcciones IP, compañía y propósito principal:

IP	**Compañía**	**Propósito**
8.8.8.8 / 8.8.4.4	Google	Rapidez y estabilidad.
1.1.1.1 / 1.0.0.1	Cloudflare	Privacidad y baja latencia
9.9.9.9	Quad9 Foundation	Bloqueo contra dominios phishing.
208.67.222.123 / 208.67.220.123	Cisco (OpenDNS FamilyShield)	Control parental.
208.67.222.222 / 208.67.220.220	Cisco (OpenDNS Home)	Filtrado configurable y es
185.228.168.9 / 185.228.169.9	CleanBrowsing	Bloqueo de malware.
185.228.168.168 / 185.228.169.168	CleanBrowsing	Control parental.
94.140.14.14 / 94.140.15.15	AdGuard	Bloqueo de publicidad.
84.200.69.80 / 84.200.70.40	DNS.Watch	Neutralidad y privacidad.
156.154.70.1 / 156.154.71.1	Neustar	Protección contra dominio
64.6.64.6 / 64.6.65.6	Verisign	Estabilidad y privacidad.

Cosas como la protección contra malware, phishing o control parental la consiguen simplemente manteniendo una lista negra de dominios y respondiendo con un registro A falso o un dominio que lleva a una página que informa del bloqueo. Pero ¿cómo puede un servidor DNS ofrecer baja latencia a usuarios de todo el planeta? Por lógica, el servidor ofrecerá menor

latencia a los usuarios que estén próximos geográficamente que a aquellos que estén lejos. La respuesta es que en realidad el servidor DNS no está en un nodo, sino que hay decenas o centenares de servidores (réplicas) con la misma dirección IP repartidos por el mundo en instalaciones PoP (Point of Presence) bajo el control administrativo de la compañía. De este modo, la latencia es baja para cualquier usuario, en cualquier lugar.

Pero eso nos lleva a otra cuestión aún más básica: ¿cómo puede haber centenares de nodos con la misma dirección IP? Esto es posible con *anycast*, una técnica de encaminamiento que puede llevar paquetes IP al nodo más próximo (mejor métrica) con la dirección destino solicitada. Realmente IPv4 no soporta *anycast* como sí lo hace IPv6, pero en la práctica el encaminamiento BGP puede lograr el mismo efecto.

21.7.1. CDN

Una CDN es un servicio de caché distribuida para recursos web estáticos: archivos JavaSript, CSS, imágenes, etc. El objetivo, como de costumbre, es aumentar la disponibilidad y reducir la latencia para el acceso desde los usuarios finales a estos recursos.

Para lograr esto, cuando un usuario pide la resolución del dominio de una CDN (*p. ej.* ajax.googleapis.com), el servidor autoritativo (gestionado por la CDN) responde con una dirección *anycast* o bien con la dirección IP de un PoP próximo al usuario, en cuyo caso además, puede tener en cuenta la carga a la hora de elegirlo, que es algo que no puede hacer *anycast* únicamente mediante encaminamiento BGP.

En este ejemplo se puede apreciar cómo, al caducar la caché, el servidor DNS configurado en el sistema empieza a devolver una dirección diferente.

```
~$ dig ajax.googleapis.com
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17254
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; ANSWER SECTION:
ajax.googleapis.com.    1    IN    A    142.250.200.106

\~$ dig ajax.googleapis.com
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13269
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; ANSWER SECTION:
ajax.googleapis.com.    290    IN    A    216.58.209.74
```

El sistema CDN también utiliza un mecanismo de caché con expiración, aunque en este caso se aprovechan los campos específicos de las cabeceras HTTP para determinar la validez de los recursos en caché. Cuando el objeto

solicitado no se encuentra o ha caducado, el nodo CDN *edge* lo obtiene del servidor origen o de un nodo intermedio de la red CDN. Los nodos intermedios se utilizan en redes CDN jerárquicas, algo habitual cuando esta alcanza un tamaño importante.

21.8. Transporte

El protocolo DNS original se encapsula sobre UDP y el servidor escucha en el puerto 53. Este sigue siendo el transporte más usado también hoy día, especialmente para consultas desde el cliente. El motivo es que UDP es ligero y de baja latencia, algo muy importante para un servicio como este que se utiliza a menudo en sesiones muy cortas.

Sin embargo, existen otros transportes estandarizados:

- TCP en el puerto 53, utilizado para transferencia de zona entre servidores autoritativos y para consultas cuyas respuestas que no caben en un único datagrama UDP.
- DNS sobre TLS (DoT) en el puerto 853, que proporciona privacidad y cifrando en consultas y respuestas.
- DNS sobre HTTPS (DoH) en el puerto 443, que también proporciona privacidad y cifrado, pero además permite *disimular* las consultas DNS dentro de tráfico HTTPS incluso sobre HTTP/2 o HTTP/3.
- DNS sobre QUIC (DoQ) en el puerto 853. Utiliza QUIC que es fiable, cifrado y con control de congestión sobre UDP.

21.9. Multicast DNS

Multicast DNS, abreviado como mDNS, permite la resolución de nombres en redes LAN sin necesidad de servidores. Utiliza el mismo formato de mensajes DNS estándar, pero los clientes envían las solicitudes a la dirección multicast `224.0.0.251`, puerto 5353.

Los nodos se autoasignan nombres basados en su hostname en el dominio especial `.local`. Este mecanismo es la base de *zero-configuration networking* (*zeroconf*) que se utiliza en redes LAN para la configuración y, sobre todo, descubrimiento automático de servicios y dispositivos gracias a DNS-SD.

Con DNS-SD y MDNS es posible anunciar y descubrir impresoras, servidores de ficheros, televisores, altavoces, dispositivos IoT, etc. Se utilizan nombres de servicio con el formato `_servicio._protocolo.local`, por ejemplo `_ssh._tcp.local` para un servidor SSH o `_ipp._tcp.local` para una impresora. La información se proporciona con los registros estándar PTR, SRV y TXT.

En GNU/Linux ambos protocolos se implementan con el sistema `avahi`. Por ejemplo, el siguiente comando lista los servicios que se están anunciando en la LAN:

```
$ avahi-browse -a
+ wlp1s0 IPv4 shellyplus1-c83d56432                Web Site              local
+ wlp1s0 IPv4 TV                                   AirPlay Remote Video  local
+ wlp1s0 IPv4 Google-Cast-Group-c2623452345234523   _googlecast._tcp      local
```

Y se puede obtener información detallada:

```
= wlp1s0 IPv4 TV                                   AirPlay Remote Video  local
   hostname = [tv.local]
   address = [192.168.0.173]
   port = [51620]
   txt = ["serialNumber=0CAK3SDMA87463F" "manufacturer=Samsung" "model=QRQ60"
   "flags=0x244" "fv=p20.0.1" "rsf=0x3" "features=0x7F8AD0,0x38BCB46"
   "deviceid=D5:7D:D0:DF:D5:46" "acl=0"]
```

21.10. Dynamic DNS

Un servicio de DDNS lo proporciona un servidor DNS autoritativo que ofrece al administrador la posibilidad de modificar dinámicamente los registros de la zona (específicamente los A) normalmente a través de un API. Un servicio que se ejecuta en el nodo detecta automáticamente cuando cambia su dirección IP y lo notifica inmediatamente al servidor DDNS mediante ese API para que actualice la resolución.

Esta funcionalidad obviamente es útil para nodos con direcciones IP dinámicas, como típicamente proporcionan los ISP a sus clientes domésticos o pequeñas oficinas. Esto permite disponer de un nombre de dominio público y fijo que apunta a un nodo que no dispone de una dirección IP fija.

Existen varios servicios servicios comerciales que ofrecen este servicio, tales como `no-ip.com`, `dyn.com`, `duckdns.org` o `freedns.afraid.org`. Aunque existe un protocolo estándar para este propósito [38], la mayoría usan API propietarias basadas en REST HTTP.

21.11. Servidores DNS *sinkhole*

Un sumidero DNS (*sinkhole*) es un servidor muy sencillo que se suele instalar en algún nodo de la LAN. Al igual que algunos de los servidores de la sección anterior, su objetivo es bloquear el acceso a dominios maliciosos o conocidos por participar en campañas de *phishing*, *spam*, publicidad, etc. La diferencia principal es que al ejecutarse en nuestra LAN tenemos plena capacidad para administrar con precisión el tipo de bloqueo que realizan. Algunos de los más conocidos son `Pi-hole` y `AdGuard Home`, que además están disponibles como contenedores `docker` con lo que su instalación resulta muy sencilla.

Los nombres de dominio más cortos del mundo
2.am n.nu b.bb c.cc g.gg t.tt b.cg b.mw e.mu m.tv n.tv r.tl 1.cm 2.cl 5.kg 7.ms p.ro e.tc s.ki o.ma v.pn c.sh b.mp