Capa de Aplicación, DNS

Andrés Barbieri [*]

24 de agosto de 2018

1. Aclaración para el Lector

Este texto fue generado en el año 2007 para el postgrado de Redes II y la materia Redes y Comunicaciones de la Fac. de Informática - UNLP. Si bien parte del software utilizado para los test ha quedado superado por nuevas versiones se cree que es útil para el entendimiento del protocolo de DNS básico.

2. Introducción

El servicio de DNS (Domain Name System) tiene la particularidad que, en la mayoría de los casos, no es utilizado directamente por los usuarios o aplicaciones, sino que funciona como un apoyo al resto de los servicios y sistemas de Internet. Su objetivo principal es el de traducir nombres de dominio a direcciones de Internet (direcciones IP) y, de esta forma, lograr una abstracción de las direcciones de red utilizadas internamente por los protocolos. Esto permite ubicar a un dispositivo por su nombre sin importar cual es la dirección IP que tiene asignada en ese momento. Otra ventaja es que las personas no necesitan recordar las direcciones IP. Para IPv4, recordar cuatro números no parece tan complicado, el problema surge cuando se deben recordar muchos de estos números y el caso empeora con la necesidad de recordar direcciones IPv6 (128 bits).

El sistema de DNS es un sistema distribuido de forma jerárquica, el cual está conformado por muchos servidores a lo largo y ancho de todo el mundo. Cada servidor tendrá la responsabilidad de mantener una parte dentro de la jerarquía de nombres.

El hecho de utilizar nombres en lugar de direcciones data de 1973. En esa época, se tenía un archivo global, `HOSTS.TXT`, que era mantenido por el SRI (Stanford Research Institute, hoy SRI International). Dentro del SRI, esta tarea dio lugar a la creación del NIC (Network Information Center). El servicio funcionaba de forma completamente centralizada. El archivo `HOSTS.TXT` estaba disponible para "bajar" utilizando el protocolo FTP y las modificaciones al mismo eran solicitadas vía e-mail. Cada determinado tiempo se juntaban las solicitudes de actualizaciones, se cambiaba el archivo y se generaba una nueva

[*]barbieri@cespi.unlp.edu.ar, con colaboración de Matías Robles: mrobles@info.unlp.edu.ar.

versión disponible al público. El sistema centralizado aseguraba que el NIC asignara las direcciones de forma consistente, pero tenía el problema de no ser escalable.

Para 1980, este método era muy difícil de mantener por lo que empezaron las investigaciones para reemplazarlo. El encargado de diseñar el nuevo sistema fue Paul Mockapetris, del USC (University of Southern California) Information Sciences Institute, y para fines de 1983 se generaron los documentos [RFC-882] y [RFC-883], que luego fueron reemplazados en 1987 por [RFC-1034] y [RFC-1035]. El sistema diseñado por Mockapetris trabaja de forma distribuida, pero concentra el comienzo de las búsquedas en un conjunto de servidores raíces y luego delega de forma jerárquica.

La primera implementación de un servidor de DNS fue realizada en 1984 en la Universidad de Berkeley y ejecutaba sobre Unix BSD. Más tarde, al ser re-escrita por otros estudiantes de la misma universidad, fue nombrada BIND (Berkeley Internet Name Domain). En la actualidad, hay diferentes versiones del servidor DNS mantenidas por el ISC (Internet Systems Consortium), que fue fundado por Paul Vixie en 1994. Además, ha sido portado a una gran cantidad de plataformas.

En la figura 1 se muestran dos configuraciones de un cliente de DNS en dos GUIs diferentes. A continuación de las imágenes se muestra la configuración de un cliente, en una interfaz orientada a caracteres, en un sistema Unix. Las configuraciones de los clientes, en la mayoría de los casos, son globales para todo el sistema operativo.

```
? cat /etc/resolv.conf
search algo.com
nameserver 4.2.2.2
nameserver 192.168.1.2
```

3. Sistema de Nombres

El espacio de nombres de Internet, sobre el cual trabaja el DNS, ha sido organizado de forma jerárquica a través de dominios, sub-dominios y nombres finales. Cada eslabón de la jerarquía es un dominio o sub-dominio (también llamados ZONAS) que contiene un espacio de nombres bajo esa denominación. Existen dominios de primer nivel en la jerarquía, conocidos como TLD (Top Level Domains), que están predefinidos. Actualmente, los TLDs son manejados por el IANA (Internet Assigned Numbers Authority) a través del ICANN (Internet Corporation for Assigned Names and Numbers). En un principio, eran manejados directamente por InterNIC. Si se acepta o rechaza agregar un nombre al tope de la jerarquía es decidido por el ICANN. Han surgido como alternativa otras organizaciones buscando su reemplazo, por ejemplo Open Root Server Network (ORSN) con apoyo europeo. Esta opero desde 2002 hasta 2008. Se mantenía sincronizada con los servidores del ICANN, pero trabaja con políticas independientes. Otro es OpenNIC, que acepta otros nombres en el nivel raíz, como por ejemplo: **.geek, .bbs, .free, .null**. Entre 2011 y 2012 el ICANN abrió los gTLD lanzando el "New gTLD Program" [NgTLD]. Esto permite poder registrar cualquier nombre de dominio en el tope de la jerarquía. El programa indica un proceso

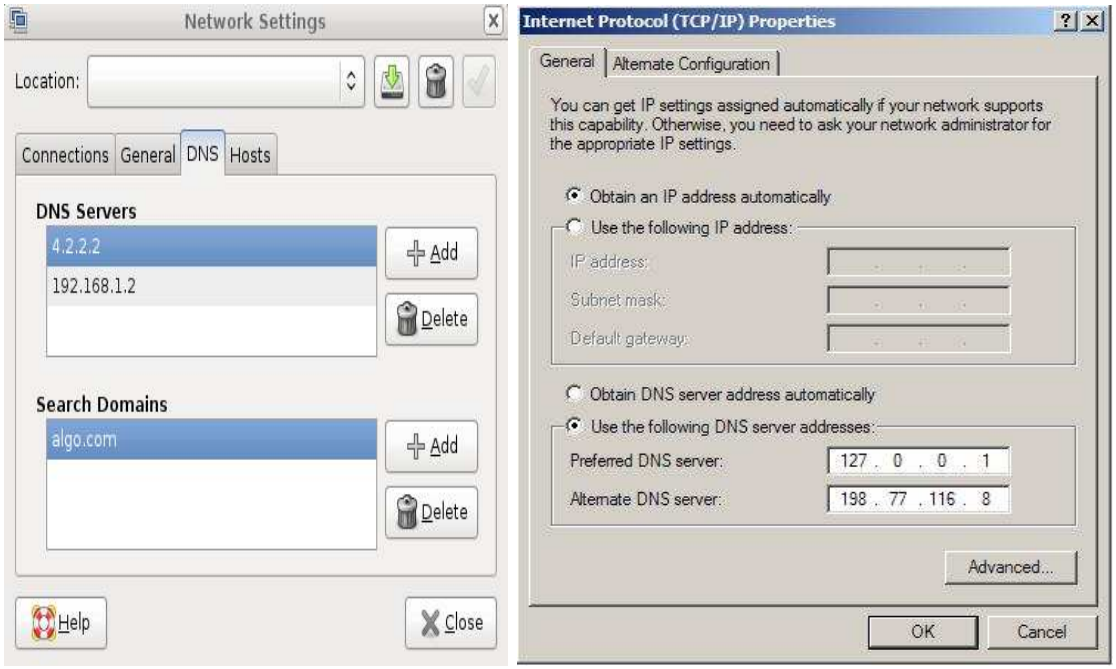

Figura 1: Configuración de DNS en Desktop GNOME (de 2008) y MS Windows.

de licitación y remate, en el cual se calcula que entrara en juego grandes sumas de dinero. Para 2015 hay más de 500 nuevos dominios, como **.beer, .paris,** etc.

El registro de los nombres no está concentrado todo en el ICANN. Éste se encarga de administrar los TLD, y luego delega registros por cada país (ccTLD). Por ejemplo, para la Argentina la entidad encargada es NIC.AR. Es también posible delegar dominios raíces a otras organizaciones.

El ICANN también estaría encargado de administrar las direcciones IPv4, direcciones IPv6 y los números de sistemas autónomos (AS). Estas tareas no se concentran todas en el ICANN, sino que son delegadas a los Registros de Internet Regionales, en inglés RIRs (Regional Internet Registry). Éstas son organizaciones que tienen a cargo diferentes regiones del mundo. Actualmente, existen las siguientes:

- American Registry for Internet Numbers (ARIN) para EE.UU., Canadá y parte del Caribe.
- Réseaux IP Européens Network Coordination Centre (RIPE NCC) para Europa, Asia Central y Media.
- Asia-Pacific Network Information Centre (APNIC) para el resto de Asia y el Pacífico.
- Latin American and Caribbean NIC (LACNIC) para América Latina y el resto del Caribe.

- African Network Information Centre (AfriNIC) para Africa.

Las organizaciones que mantienen los registros brindan servicios de consulta vía HTTP o el protocolo WHOIS [RFC-1834]. Por ejemplo, se podrían consultar los datos referentes a un dominio o un rango de direcciones IP:

```
? whois 163.10.0.0
...
inetnum:     163.10/16
status:      assigned
owner:       Universidad Nacional de La Plata
ownerid:     AR-UNLP-LACNIC
address:     50 y 115, 3er piso - CeSPI, UNLP
country:     AR
owner-c:     ZU58-ARIN
inetrev:     163.10/16
nserver:     UNLP.UNLP.EDU.AR
nsstat:      20080926 AA
nslastaa:    20080926
created:     19920701
changed:     20010530
source:      ARIN-HISTORIC

nic-hdl:     ZU58-ARIN
person:      Universidad Nacional de La Plata
e-mail:      noc@unlp.edu.ar
address:     50 esq. 115 3er piso - La Plata - BA - 1900
country:     AR
phone:       +54 221 423 6609
source:      ARIN-HISTORIC
...

? whois berkeley.edu
...
Domain Name: BERKELEY.EDU

Registrant:
   University of California at Berkeley
   Information Systems & Technology
   Communication and Network Services 2484 Shattuck #1640
   Berkeley, CA 94720-1640
   UNITED STATES
```

```
Administrative Contact:
   kenneth q lindahl
   University of California, Berkeley
   Communication and Network Services
   2484 Shattuck Ave, #1640
   Berkeley, CA 94720-1640
   UNITED STATES
   (510) 642-0866
   noc@nak.berkeley.edu

Technical Contact:
   kenneth q lindahl
   University of California, Berkeley
   Communication and Network Services
   2484 Shattuck Ave, #1640
   Berkeley, CA 94720-1640
   UNITED STATES
   (510) 642-0866
   noc@nak.berkeley.edu

Name Servers:
   ADNS1.BERKELEY.EDU      128.32.136.3
   ADNS2.BERKELEY.EDU      128.32.136.14
   UCB-NS.NYU.EDU
   PHLOEM.UOREGON.EDU
   DNS2.UCLA.EDU
...
```

3.1. TLD (Top Level Domains)

Los TLD más importantes se podrían clasificar en 3 grupos:

gTLD, Generic TLD: contienen dominios con propósitos particulares, de acuerdo a diferentes actividades. Para 1980, los gTLD eran: **.com, .edu, .gov, .int, .mil, .net** y **.org**, pero sólo **.com, .net** y **.org** tenían libre el registro, el resto estaban dedicados, únicamente, a organizaciones de los Estados Unidos u organizaciones Internacionales. Debido a esto, se creía que todos los registros (incluso los 3 libres) se encontraban en Estados Unidos. Las políticas de uso y registro pueden estar definidas por el ICANN, en ese caso se llaman **Unsponsored TLD**, o, si son definidas por otra organización a la cual se le delega el control se los llama **Sponsored TLD**. Para 2008, según [gTLD-ICANN], los gTLD son los mostrados en la figura 2. Los dominios **.com** aún hoy son administrados por empresas lucrativas como VeriSign.

ccTLD Country-Code TLD: contienen dominios delegados a los diferentes paises del mundo. Los códigos de los paises están codificados en dos símbolos, habitualmente letras, y actualmente se admiten símbolos o letras de otros alfabetos. Se encuentra por ejemplo **.ar (Argentina), .tv (Tuvalu), .zw (Zimbabwe), .uk (Reino Unido), .ru (Rusia), .us (Estados Unidos), etc**. En la figura 3 se muestran algunos ejemplos [ccTLD-ICANN]. Algunos países como el caso de Tuvalu (.tv) o Colombia (.co) delegan la administración de dominios a empresas/organizaciones para obtener ganancias.

.ARPA TLD: es un dominio especial, llamado de infraestructura, usado internamente por los protocolos, creado para resolución de reversos: de direcciones a nombres. Se muestra integrado en la figura 3.

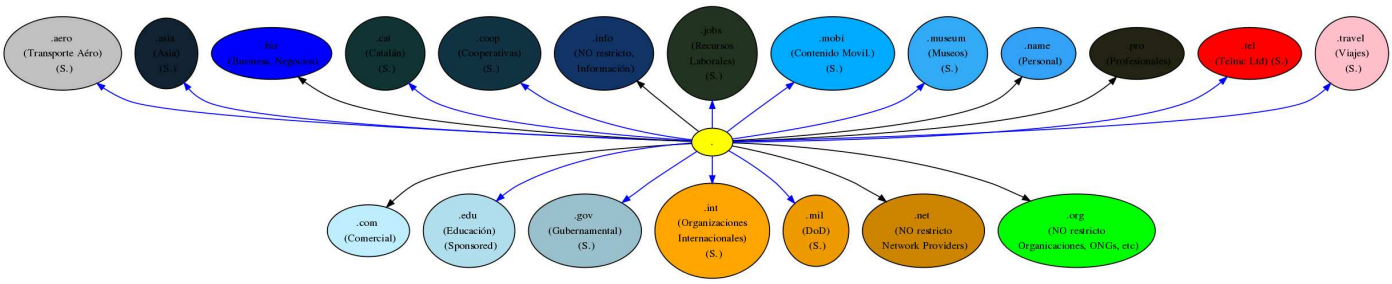

Figura 2: Generic TLD hasta 2008.

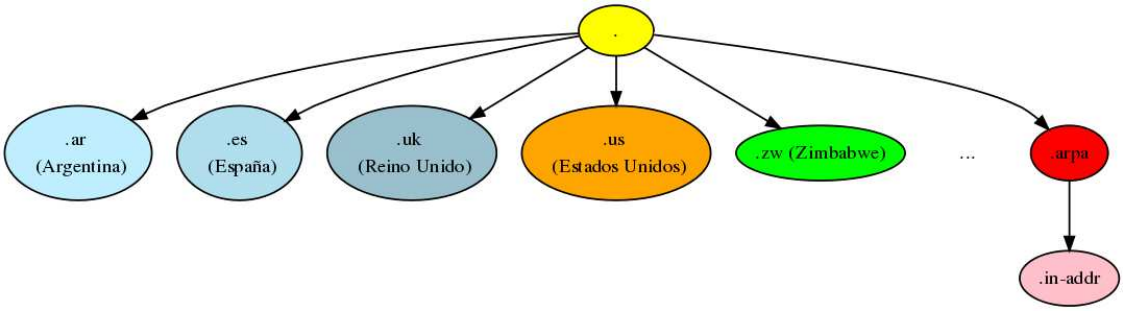

Figura 3: Country Code y ARPA TLD.

Un nombre de dominio es una lista de labels (etiquetas) separadas por puntos que van desde el nodo (el label más a la izquierda) hasta la raíz del árbol (el punto). No son case-sensitive y cada label puede tener como máximo 63 caracteres. La cantidad máxima de labels es 127 y el nombre no puede tener más de 255 caracteres en total. Al nombre completo se lo identifica como FQDN (Full Qualified Domain Name) y se distingue por terminar con un punto (dot). Si no termina con punto se le puede agregar/concatenar,

al momento de hacer la búsqueda, un espacio del domino. Si el nombre sólo tiene un label se le agrega un domino, si tiene más eslabones, generalmente, se lo considera como un FQDN.

```
paraguil (No FQDN)
paraguil.cities.org. (FQDN)
paraguil.cities.org (Considerado FQDN)
```

Para los siguientes nombres de dominio de ejemplo se muestra la jerarquía en la figura 4.

```
web.ru.
www.net.
ada.info.unlp.edu.ar.
ww.unlp.edu.ar.
www.l.google.com.
other.com
30.8.10.163.in-addr.arpa.
```

3.2. Delegación de Sub-dominios/Zonas

Una vez definido el espacio de nombres, y su administración, es necesario asignarlo a diferentes servidores de manera distribuida. Para 1993, había 8 servidores, llamados **ROOT Servers (Servidores Raíces)**, encargados de atender la raíz del servicio. Estos eran administrados por InterNIC y se encargaban de delegar cada una de las zonas generadas para los TLD, tanto gTLD como ccTLD. La delegación consiste en saber las direcciones IP de los servidores que se encargan de resolver (o sub-delegar) las zonas de manera **autoritativa**. Un servidor **autoritativo** tiene toda la información para una zona, puede producir cambios sobre la misma y es el que tiene la última versión.

En la actualidad, existen 13 **ROOT Servers** distribuidos en todo el mundo, 7 de los cuales no son servidores únicos, sino que trabajan con redundancia y las réplicas están distribuidos geográficamente. Con esta redundancia, combinada con **Ruteo Anycast**, se logra que los requerimientos se atiendan según la proximidad, logrando mejores tiempos de respuesta. La cantidad de servidores raíces está acotada a 13 debido a la limitación de 512 bytes de un mensaje de DNS sobre UDP. A continuación se listan los servidores raíces disponibles en el año 2015.

```
.  518400  IN      NS      A.ROOT-SERVERS.NET. # Versign-grs.com
.  518400  IN      NS      B.ROOT-SERVERS.NET. # ISI.edu
.  518400  IN      NS      C.ROOT-SERVERS.NET. # Cogent.com (ANYCAST)
.  518400  IN      NS      D.ROOT-SERVERS.NET. # UMD.edu (Univ. Maryland)
.  518400  IN      NS      E.ROOT-SERVERS.NET. # NASA.gov
.  518400  IN      NS      F.ROOT-SERVERS.NET. # ISC.org (ANYCAST)
.  518400  IN      NS      G.ROOT-SERVERS.NET. # NIC.mil
.  518400  IN      NS      H.ROOT-SERVERS.NET. # ARMY.mil
```

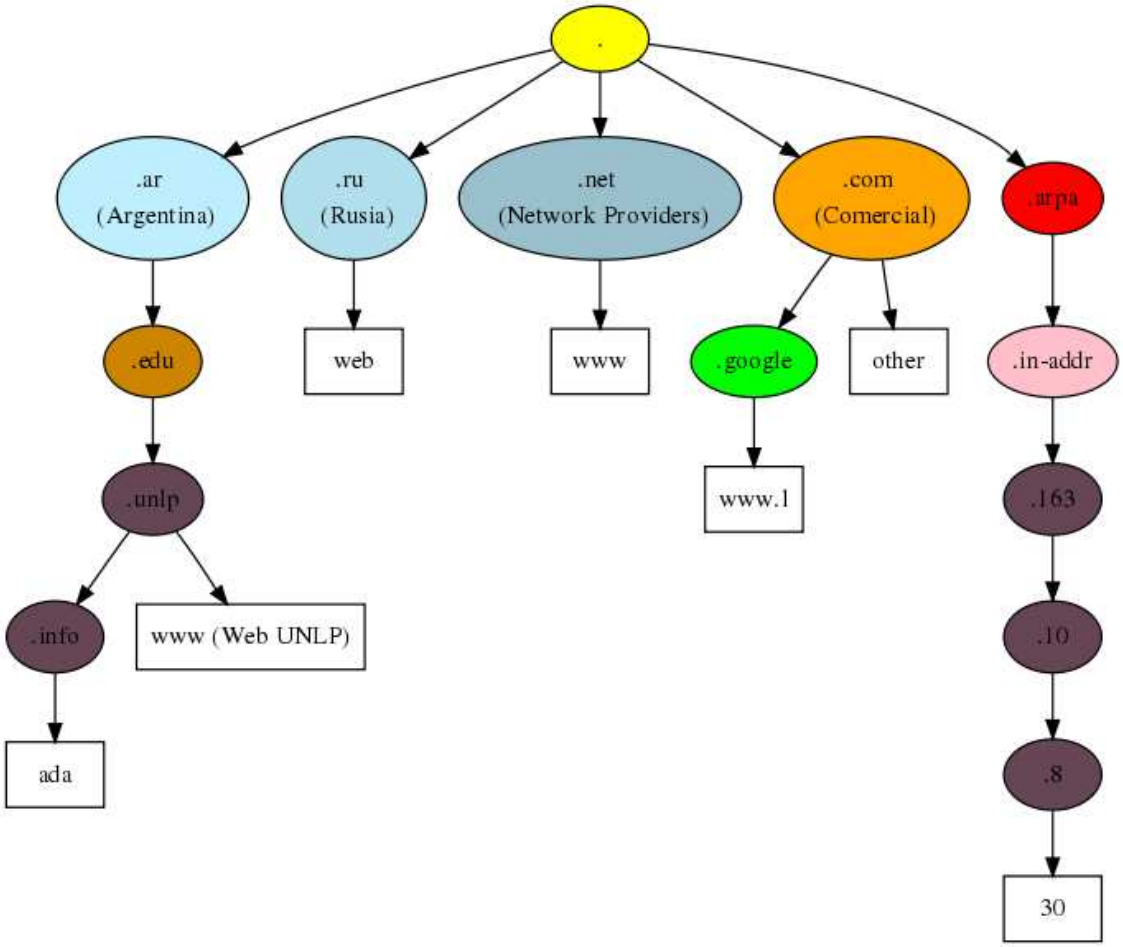

Figura 4: Ejemplos de Nombres de Dominio

```
.   518400  IN      NS      I.ROOT-SERVERS.NET. # NIC.ddn.mil (ANYCAST)
.   518400  IN      NS      J.ROOT-SERVERS.NET. # Versign-grs.com (ANYCAST)
.   518400  IN      NS      K.ROOT-SERVERS.NET. # RIPE.net (ANYCAST)
.   518400  IN      NS      L.ROOT-SERVERS.NET. # ICANN.org (ANYCAST)
.   518400  IN      NS      M.ROOT-SERVERS.NET. # WIDE.ad.jp (ANYCAST)

A.ROOT-SERVERS.NET.     3600000 IN      A       198.41.0.4
A.ROOT-SERVERS.NET.     3600000 IN      AAAA    2001:503:ba3e::2:30
B.ROOT-SERVERS.NET.     3600000 IN      A       192.228.79.201
C.ROOT-SERVERS.NET.     3600000 IN      A       192.33.4.12
D.ROOT-SERVERS.NET.     3600000 IN      A       128.8.10.90
```

```
E.ROOT-SERVERS.NET.     3600000 IN      A       192.203.230.10
F.ROOT-SERVERS.NET.     3600000 IN      A       192.5.5.241
F.ROOT-SERVERS.NET.     3600000 IN      AAAA    2001:500:2f::f
G.ROOT-SERVERS.NET.     3600000 IN      A       192.112.36.4
H.ROOT-SERVERS.NET.     3600000 IN      A       128.63.2.53
H.ROOT-SERVERS.NET.     3600000 IN      AAAA    2001:500:1::803f:235
I.ROOT-SERVERS.NET.     3600000 IN      A       192.36.148.17
J.ROOT-SERVERS.NET.     3600000 IN      A       192.58.128.30
J.ROOT-SERVERS.NET.     3600000 IN      AAAA    2001:503:c27::2:30
```

Esta lista se puede obtener vía una consulta de DNS o usando FTP.

```
? wget ftp://ftp.internic.net/domain/named.root
```

```
? dig -t ns . @j.root-servers.net
```

En las figuras 5 se muestra la distribución de los **ROOT Name Servers** en el mundo para 2015.

3.3. Registro de un nombre de dominio

En el proceso entran en juego diferentes entidades:

Registrant: cliente/entidad final, el que requiere registrar el nombre de dominio.

Registry: repositorio autoritativo responsable por el funcionamiento y la información en un TLD particular. Debe proveer la infraestructura de los servidores para resolver el servicio desde la raíz.

Registrar: interfaz entre el Registry y el Registrant. Puede proveer servicios adicionales al Registrant.

Depende de cada dominio como serán las políticas. Las políticas siempre deben estar en el marco de las definidas por el ICANN. Para un TLD debe *existir solo una autoridad responsable*, el Registry. Esta a menudo abre el servicio en *varios Registrars* como interfaz de registro. Los Registrant contactan a un Registrar para la operación. En la figura 6 se muestra esquemáticamente la operación.

En la República Argentina el registro de los nombres de dominio `.com.ar, .org.ar, net.ar` y otros esta bajo la responsabilidad de **NIC.ar**: http://www.nic.ar, organización que dependía anteriormente del Ministerio de Relaciones Exteriores, Comercio Internacional y Culto, indicado a menudo como la Cancillería (La Cancillería es parte del Ministerio). Actualmente depende de la Secretaría Legal y Técnica de Presidencia de la Nación, que es una parte de la Secretaría General de la Nación. NIC.ar cumpliría el rol de Registry. Para el registro de dominios `.edu.ar` el trámite debe realizarse ante la ARIU (Asociación de Redes Inter Universitaria) de la Argentina (la delegación a este organismo lo realiza el Ministerio de Exterior). Para registrar dominios `.com` esto se debe realizar ante una empresa autorizada por el ICANN, Registrars. Para algunos dominios se pude hacer el trámite directamente sobre el Registry.

Para registrar un dominio `.com.ar` es necesario hacerlo ante NIC.ar indicando la Entidad Registrante (Registrant), de la Persona Responsable y los DNS correspondientes a los servidores que resolverán el nombre de dominio. Puede suceder que el dominio al tiempo de registro no tenga asociados los servidores, esto puede realizarse más tarde realizando la Transferencia de Dominio. Para la transferencia se requieren al menos 2 (dos) servidores de nombres autoritativos para el dominio registrado. Para realizar el trámite requiere un casilla de e-mail a la cual se le notifica el estado del mismo. No existen Registrars, pero ISPs pueden cumplir un rol similar ofreciendo el servicio. No hay acuerdo entre Registry, **NIC.ar** e ISP.
Para la obtención de un bloque de direcciones IP en nuestra región geográfica se debe tramitar ante un ISP local o directamente ante LACNIC. Para otros países fuera de nuestra región se debe contactar el RIR adecuado. En el caso de hacerlo ante un ISP el bloque sería delegado por el ISP, sin ser propio.

3.4. Servidores de DNS

Los tipos de servidores de DNS se pueden clasificar en 3 grupos, no siendo estos disjuntos, es decir, un servidor puede pertenecer a más de un grupo.

Servidor Raíz: es un servidor que cumple el rol descripto anteriormente en la sub-sección: Delegación de Sub-dominios/Zonas.

Servidor Autoritativo: es un servidor que tiene a cargo una zona o sub-dominio de nombres. Es el responsable del mantenimiento de la misma. Pueden existir uno o más servidores responsables de una zona. A su vez, un servidor puede ser responsable de varias zonas. Si un sub-sub-dominio es delegado dentro de esta zona, el servidor debe saber a qué servidor(es) sub-delegarlo.

Servidor Local: es un servidor que es consultado dentro de una red, es el encargado de resolver los nombres solicitados por los clientes. Mantiene una cache con los nombres que va resolviendo. Al mismo tiempo, puede ser **Servidor Autoritativo** de la zona o espacio de nombres de los equipos de su red. Habitualmente también se los llama **Servidor Recursivo** porque responde estas consultas a los clientes internos o **Servidor Cache** porque almacena temporalmente los registros que obtiene.

Otra solución en lugar de usar un DNS local (tanto en la red como en el propio equipo), es utilizar un **Open DNS/Open NS**, estos son servidores de DNS globales que permiten utilizase como locales admitiendo consultas recursivas. Una lista de estos se podrá encontrar en: `http://www.dnsserverlist.org/`. Algunos de estos servicios funcionan en un esquema anycast.
Existe otro tipo de servidor de DNS llamado **Forwarder Name Server**. Éste es un servidor que interactúa directamente con los **ROOT Name Servers** y con el resto del sistema de DNS, haciendo de intermediario o proxy entre los servidores locales y el mundo exterior. Este tipo de configuración se utiliza, habitualmente, por cuestiones de seguridad. Sólo se permiten/restringen consultas de DNS que salgan de la red local desde los **Forwarder Name Server**. Hay implementaciones livianas (lightweight) de Forwarders que combinan un DNS proxy con un DHCP server y un servicio de NAT. Un ejemplo de esto es [Dnsmasq] que es muy útil para una intranet de tamaño pequeño o mediano.

4. Protocolo DNS

4.1. Funcionamiento del Protocolo

4.1.1. Cliente/Servidor y Resolver

El protocolo de DNS puede trabajar sobre dos protocolos de transporte: TCP y UDP, siendo UDP el más utilizado. El número de puerto usado por el servidor es el 53 para ambos casos. El cliente puede escoger cualquier puerto no privilegiado. El protocolo UDP es utilizado en las consultas y en la mayoría de las respuestas. TCP se utilizará en caso de que la respuesta UDP exceda los 512 bytes o en el caso particular de una **Transferencia de Zona** (copia de la base da datos de nombres de un servidor primario o master a uno secundario -antiguamente llamados esclavos-).

```
? grep domain /etc/services
domain          53/tcp                          # name-domain server
domain          53/udp
```

El funcionamiento del DNS se realiza en un esquema cliente/servidor, siendo el cliente cualquier aplicación que requiera la resolución de nombres. El código (instrucciones de máquina) del cliente se lo agrupa en un módulo llamado **Resolver**. El **Resolver** se lo podría considerar como un agente encargado de resolver los nombres a solicitud del cliente, por ejemplo: un servidor web, el cliente telnet, un navegador web, un servidor de mail, etc. Este agente, generalmente, no se implementa como un servicio activo, sino como un conjunto de rutinas encapsuladas en una biblioteca de funciones que se link-edita conjuntamente con la aplicación. En el caso de sistemas Unix, el `resolver(3)(5)` es un conjunto de funciones incluidas en la **C library (libc)**. Las funciones básicas provistas por la `libc` para la resolución de nombres son:

```
#include <netdb.h>
#include <sys/socket.h>       /* for AF_INET */

struct hostent *gethostbyname(const char *name);
struct hostent *gethostbyaddr(const void *addr, socklen_t len, int type);
struct hostent *gethostent(void); /* System V/POSIX extension */
```

Este tipo de resolver no realiza ninguna forma de caching y se lo suele llamar **Stub/Dumb Resolver**. El encargado del caching, en este caso, es el **Servidor Local**. Existen otras implementaciones que tienen un resolver activo, **Smart Resolver**, funcionando en cada equipo como si fuese un **Servidor Local**, que permiten realizar caching u ofrecer funcionalidades extras.

4.1.2. Resolución de Nombres, Iterativo vs. Recursivo

Existen dos formas posibles en las cuales se puede resolver una consulta de nombre; iterativo y recursivo. En la figura 8 se muestran los dos métodos.

1. Cuando una aplicación requiere un servicio del sistema de DNS, por ejemplo resolver un nombre a su correspondiente dirección IP, invocará la función ofrecida por el resolver. Suponiendo que éste es un **Stub Resolver**, sólo buscará en su configuración la lista de servidores de nombres locales.

2. Luego, enviará la consulta al primero de éstos. El **Servidor Local** será el responsable de responderle de forma afirmativa con una o más direcciones IP, o de forma negativa indicando que no encontró lo consultado.

3. Si el primer servidor no responde (no está activo), el resolver podrá tratar de resolver la consulta utilizando el siguiente de los servidores locales configurados.

4. Los servidores locales podrán responder utilizando su cache o saliendo a buscar la consulta hacia los **ROOT Name Servers**. Existe la posibilidad que los servidores locales no hablen, directamente, con los **ROOT Name Servers** y lo hagan sólo a través de un **Forwarder Name Server**. También, es posible que el servidor local sea un **Servidor Autoritativo** y recibe una consulta de un nombre sobre el cual él tiene autoridad. En este caso, responde desde su base de datos de nombres indicando que es una **Respuesta Autoritativa**, caso contrario, puede responder tomando los datos de la cache, indicando que es una **Respuesta NO Autoritativa**.

5. El resolver delega, por completo, la responsabilidad en el **Servidor Local** y éste se debe encargar de resolver la consulta. Esto es lo que se llama una **Consulta Recursiva**. Sólo los servidores locales deberían admitir consultas recursivas desde los equipos de su misma red. Los **ROOT Name Servers** no deberían admitir prácticamente consultas recursivas.

6. El **Servidor Local**, una vez que tiene la consulta y se ha fijado que no es por un nombre local ni está en su caché, deberá intentar resolverla externamente. Podrá intentar resolverlo de forma **Recursiva**, preguntando a otro servidor de nombres y delegando la resolución en éste. Como, en general, las consultas **Recursiva** están sólo habilitadas de forma local, deberá resolverla con una **Consulta Iterativa** encargándose el mismo.

7. Comenzará preguntando de forma iterativa a los **ROOT Name Servers**. Estos no responderán con la resolución del nombre, sino que se fijarán en la consulta, comenzando del último label del nombre (el de más arriba en la jerarquía), cuáles son los servidores encargados de este dominio y responderán con una lista de los mismos. Es posible que tengan conocimiento de más labels, lo que permitirá enviar una respuesta más cercana y acortar los saltos de la búsqueda.

8. Luego, el **Servidor Local**, a partir de la lista de servidores de nombres para el primer eslabón de la consulta, preguntará al primero de estos, nuevamente, por el nombre completo. Si este servidor es **No autoritativo**, como sucedía con los **ROOT Name Servers**, para el sub-dominio indicado responderá con otra lista de servidores que tienen delegado el sub-dominio. De esta forma, irá bajando en la jerarquía hasta llegar a un **Servidor Autoritativo** para el sub-dominio consultado.

9. Cuando la consulta llegue a un **Servidor Autoritativo**, éste resolverá el nombre de la consulta a la correspondiente dirección IP y se la entregará al **Servidor Local**.

10. El **Servidor Local**, una vez con la consulta resuelta, la guardará en la caché y responderá al **Resolver**.

11. Las sucesivas consultas, por el mismo nombre, se podrán resolver directamente desde la caché hasta que venza el tiempo de vida de la entrada en la caché. Si se vence este tiempo, se elimina la entrada de la cache y el proceso se deberá ejecutar nuevamente.

4.2. Estructura de los Mensajes de DNS

El mensaje de DNS tiene un encabezado fijo y un cuerpo variable.

Identificador: valor aleatorio generado por el cliente y copiado por el servidor en la respuesta. Valor entre 0 y 64K-1. Usado por el cliente para asociar las consultas con las respuestas. Además, el cliente pude usar diferentes números de ports UDP para agregar entropía a las consultas. También se lo conoce como **Transaction ID**.

Flags: se codifican varias indicaciones. Indica si es una consulta o una respuesta (QR), si es autoritativa o no (AA), si la respuesta esta truncada o no (TC) -no entra en un mensaje UDP, deberá usarse TCP-, si es recursiva o no (RD) habilitada por el cliente, recursión permitida o no (RA) habilitada por el servidor, código de operación, si es estándar una notificación, un update, etc.

Contadores varios: de cantidad de consultas, respuestas, autoritativas o no que lleva el mensaje.

Cuerpo: respuestas y/o consultas, más datos adicionales. Dentro de las preguntas y respuestas se pueden trabajar con diferentes tipos de información.

4.3. Servicios y Registros de DNS

Un servidor de DNS almacena la información con la que trabaja en una base de datos (DB) local. En la mayoría de los casos no es una base de datos relacional, aunque podría usarse. La DB es una estructura propia del sistema que se genera en memoria una vez que arranca el servicio y se configura desde los archivos que residen en el sistema de archivos local. Dentro de la base de datos, la información se organiza en registros llamados Resource Record (RR). Cada uno de estos registros puede guardar diferente tipo de información. Ejemplos de estos tipos se describen a continuación.

4.3.1. Registros A (Address)

Son registros que mapean de un nombre de domino a una dirección IP. Son los más comunes. Pueden existir varios registros **(A)** con el mismo nombre, por ejemplo, para realizar balance de carga de un servidor muy accedido. Para aprovechar esto, el servidor de DNS debería responder con una lista de direcciones IP, siempre alternando el resultado con algún criterio, por ejemplo de forma circular.

Tomando el dominio cities.org[1] como prueba se puede configurar:

[1]Existe un domino con este nombre registrado por service@stenzel.org, lo mejor hubiera sido usar el dominio .test

```
# less /etc/bind/db.cities.org
...
berlin.cities.org.        IN        A     172.20.1.100
brasilia.cities.org.      IN        A     172.20.1.5
paraguil-br0.cities.org.  IN        A     172.20.1.1
...
```

En la clasificación de RR, antes de los **Tipos** existen **Clases**. En este documento se trabajará, únicamente, con la clase INTERNET **(IN)**. Las clases permiten utilizar el protocolo de DNS no sólo para IP. A continuación, usando la herramienta `dig(1)`, se muestra un ejemplo de balance para los servidores de Google.

```
? dig +nocomments +nostats www.google.com

; <<>> DiG 9.4.1-P1.1 <<>> +nocomments +nostats www.google.com
;; global options:  printcmd
;www.google.com.                        IN      A
www.google.com.         42938   IN      CNAME   www.l.google.com.
www.l.google.com.       300     IN      A       74.125.47.103
www.l.google.com.       300     IN      A       74.125.47.104
www.l.google.com.       300     IN      A       74.125.47.147
www.l.google.com.       300     IN      A       74.125.47.99

? dig +nocomments +nostats www.google.com

; <<>> DiG 9.4.1-P1.1 <<>> +nocomments +nostats www.google.com
;; global options:  printcmd
;www.google.com.                        IN      A
www.google.com.         43199   IN      CNAME   www.l.google.com.
www.l.google.com.       300     IN      A       74.125.47.147
www.l.google.com.       300     IN      A       74.125.47.99
www.l.google.com.       300     IN      A       74.125.47.103
www.l.google.com.       300     IN      A       74.125.47.104
```

4.3.2. Registros PTR (Pointer)

Estos son registros que mapean direcciones IP a nombres de dominio. Son el inverso de los registros **(A)**, por esto, se los suele llama reversos. Trabajan en el dominio especial **in-addr.arpa**. Por ejemplo:

```
# less /etc/bind/db.172
...
1.1.20   IN      PTR     paraguil-br0.cities.org.
5.1.20   IN      PTR     brasilias.cities.org.
```

```
100.1.20 IN      PTR     berlin.cities.org.
...
5.1.19   IN      PTR     sucre.lat.org.
1.1.19   IN      PTR     paraguil-tap2.lat.org.
...
```

En el primer ejemplo, la red 172.0.0.0/8 no está sub-delegada y existe un dominio unificado. Se podría delegar dando como resultado una sub-zona para la red 172.20.0.0/16.

```
# less /etc/bind/db.172.20
...
1.1   IN      PTR     paraguil-br0.cities.org.
5.1   IN      PTR     brasilias.cities.org.
100.1 IN      PTR     berlin.cities.org.
...
```

Los registros **(PTR)** deben estar en un sub-árbol (dominio) separado que, como se mencionó más arriba, se llama **in-addr.arpa**. Esto se debe a que la búsqueda se realiza usando la dirección IP y no el nombre. Aunque, la información existe en los registro directos **(A)**, no se puede generar un mecanismo de búsqueda organizado, ya que dada una dirección IP no hay forma de saber en donde, qué zona, está asignada. Para este propósito, está el sub-árbol **in-addr.arpa** que organiza las direcciones por octeto de las direcciones IP, generando un árbol con la estructura mostrada en la figura 10. De esta manera, se provee un mecanismo a modo de índice de búsqueda.

Como una medida de seguridad, las consultas de reversos o registros **(PTR)** suelen ser realizadas por los servidores al recibir conexiones TCP. El servidor intentará resolver la dirección IP, desde donde recibe la conexión, para comprobar que es una conexión válida, no está spoofeada. Existen servidores que con solo comprobar la existencia del registro **(PTR)** les alcanza para considerar que no hay spoofing. En otros casos más estrictos, los servidores, además, exigen que el valor de este registro concuerde con el registro **(A)**. Si esto no sucede, podrían rechazar la conexión. Más adelante, se muestra un ejemplo.

4.3.3. Registros CNAME (Canonical Name)

Son registros que mapean de un nombre de domino a otros nombres. Se los conoce como aliases, debido a que dado un nombre indican el nombre original. Esta funcionalidad podría, también, resolverse con varios registros **(A)**. El nombre, al que apunta, debería ser el nombre original, conocido como canónico.

```
# less /etc/bind/db.cities.org
...
ftp.cities.org.          IN    CNAME    berlin.cities.org.
www.cities.org.          IN    CNAME    berlin.cities.org.
...
```

Si no se usasen registros **(CNAME)** se podría generar la siguiente configuración:

```
# less /etc/bind/db.cities.org
...
berlin.cities.org.        IN        A     172.20.1.100
ftp.cities.org.           IN        A     172.20.1.100
www.cities.org.           IN        A     172.20.1.100
...
```

4.3.4. Registros HINFO (Hardware Info)

Son registros que mapean de un nombre de domino al hardware y sistema operativo que un equipo posee. No son utilizados habitualmente por cuestiones de seguridad y ahorro de memoria. El formato que lleva el registro debe respetarse, son dos valores: primero un string para el hardware y luego otro para el sistema operativo. Los strings para los valores son arbitrarios. Ejemplo:

```
# less /etc/bind/db.cities.org
...
berlin.cities.org.        IN     HINFO     Pentium-II/256  GNU/Linux
brasilia.cities.org.      IN     HINFO     MIPS-R10000/512 IRIX6.5
...
```

4.3.5. Registros TXT (Textual)

Son registros que mapean de un nombre de domino a información extra asociada con el equipo que tiene dicho nombre, por ejemplo pueden indicar finalidad, usuarios, etc. No son utilizados habitualmente. Se los puede ver en uso asociando una clave pública, utilizando por ejemplo IPSec con un esquema de Opportunistic Encryption.

```
# less /etc/bind/db.cities.org
...
berlin.cities.org.  IN  TXT "Servidor Web y FTP"
; RSA 2048 bits berlin.cities.org
berlin.cities.org.  IN  TXT "X-IPsec-Server(10)=172.20.1.100" " AQOF8tZ2...+buFuFn/"
...
```

4.3.6. Registros MX (Mail Exchanger)

Son registros que indican, para un nombre de domino, cuáles son los servidores de mail SMTP encargados de recibir los mensajes para ese dominio. De esta forma, no es necesario especificar el servidor completo de mail donde se encuentra la casilla destino, alcanza con indicar, solamente, el dominio. El servidor de mail SMTP que envía el mensaje deberá consultar, vía el servicio de DNS, cuáles son los servidores SMTP receptores para el dominio dado. Se puede detallar una lista de servidores y asignarles prioridades. Los valores más bajos son las mejores prioridades. Así, al momento de hacerse el envío del mensaje, si el servidor primario no está activo se podría recurrir a otro de menor prioridad para enviar

el e-mail. Con la asignación de prioridades con igual valor se podría lograr un balance de carga entre los distintos servidores SMTP.

Suponiendo que se desea enviar un e-mail a la cuenta `<pepe@gmail.com>`, los servidores SMTP destinos en orden, de acuerdo a su prioridad, serían los siguientes:

```
gmail.com          IN    MX     5 gmail-smtp-in.l.google.com.
gmail.com          IN    MX    10 alt1.gmail-smtp-in.l.google.com.
gmail.com          IN    MX    10 alt2.gmail-smtp-in.l.google.com.
gmail.com          IN    MX    50 gsmtp147.google.com.
gmail.com          IN    MX    50 gsmtp183.google.com.
```

En el ejemplo del dominio cities.org se podría tener un esquema como el siguiente:

```
# less /etc/bind/db.cities.org
...
cities.org.               IN       MX 1   brasilia.cities.org.
cities.org.               IN       MX 10  berlin.cities.org.
...
```

siendo el servidor `brasilia.cities.org` el más prioritario.

4.3.7. Registros NS (Name Server)

Los registros **(NS)** indican los servidores de nombre autoritativos para una zona o sub-dominio. A partir de esto, se puede lograr una delegación de sub-dominios. A diferencia que los registros **(MX)**, estos no llevan asociado una prioridad, todos los servidores tienen la misma precedencia. Para lograr un mejor balance, al ir respondiendo se debería ir rotando el orden con el que se entregan los servidores autoritativos para una zona. Al tener varios servidores para un mismo dominio no es necesario configurar a todos con los mismos datos. El software de DNS permite asignar roles de **Servidor Primario** y **Servidor(es) Secundario(s)**. De esta forma, sólo se requiere configurar el primario y luego el secundario obtendrá una copia de la base de datos del servidor maestro/primario. Es importante que los servidores estén actualizados, por eso debe existir algún mecanismo para mantenerlos sincronizados. El **Servidor Primario** debería avisar a todos los **Servidores Secundarios** cuando se realiza un cambio, así estos re-copian la base de datos de nombres desde el servidor maestro. La comunicación entre servidores se realiza vía el mismo protocolo de DNS, con la diferencia que se hace sobre TCP en lugar de UDP, esto se debe a que, generalmente, los datos transmitidos ocupan más de 512 bytes (máximo para DNS sobre UDP). Esta operación se llama **Transferencia de Zona** o, en inglés, **Zone Transfer**.

Se recomienda sobre una configuración de un servidor de DNS, en el cual se delegan las zonas, y las direcciones IP de los servidores delegados están en la zona delegada, agregar para cada registro **(NS)** un registro **(A)**, indicando la dirección IP del servidor delegado. Esto se utiliza para agilizar las búsquedas y ahorrar una doble consulta. Estas entradas **(A)** se las conoce como **GLUE RECORDS**, ya que "pegan"

un nombre de un servidor de nombres con su correspondiente dirección IP. En el ejemplo de cities.org se podría tener dos zonas: **cities.org** y **trees.cities.org**.

```
# less /etc/bind/db.cities.org
...
; ## ZONA RAIZ
cities.org.             IN         NS berlin.cities.org.
cities.org.             IN         NS brasilia.cities.org.

; ## BRAISLIA NO NECESITA GLUE RECORD ##

; ## ZONA delegada
trees.cities.org.       IN         NS brasilia.cities.org.
trees.cities.org.       IN         NS berlin.cities.org.
trees.cities.org.       IN         NS oak.trees.cities.org.

; ## GLUE RECORD ##
oak.trees.cities.org.   IN         A 192.168.40.1
...
```

4.3.8. Registros SOA (Start Of Authority)

Los registros **(SOA)** se crean por cada zona o sub-zona que brinda el servicio de DNS. En este registro se especifican los parámetros globales para todos los registros del dominio o zona. Sólo se admite un registro **(SOA)** por zona. Los parámetros que comprende son los siguientes.

Nombre del servidor: nombre del servidor de nombres. En general, se llaman **ns.**...

E-mail del administrador: dirección de e-mail del administrador, reemplazando el símbolo arroba "@" por un punto ".".

Número de Serie: número de serie de la DB. Se debe actualizar cada vez que se hace alguna modificación al dominio. Se sugiere un formado YYYYMMDDSS. Tamaño de 32 bits.

Refresh Time: cada cuánto tiempo los servidores secundarios deben refrescar desde el maestro/primario. Tamaño de 32 bits. Según la [RFC-1912], se recomienda entre 1200 a 43200 segundos (horas).

Retry Time: el tiempo que debe esperar un servidor secundario en intentar contactar nuevamente al primario, ante fallos de conexión, cuando necesita hacer un refresh. Tamaño de 32 bits. Según la [RFC-1912], se recomienda entre 180 a 900 segundos (minutos).

Expiry Time: el tiempo que deja de ser autoritativa una zona. Luego de este tiempo, el servidor secundario debe chequear el Serial de la SOA del maestro y, si cambio, realizar una **Transferencia de Zona**: AFXR/IXFR. Tamaño de 32 bits. Según la [RFC-1912], se recomienda entre 1209600 a 2419200 segundos (semanas).

TTL: en un principio se utilizaba como el Tiempo de Vida de un registro en una caché. Ahora se utiliza como el TTL de Cacheo Negativo: si se pregunta por un valor y el servidor autoritativo respondió que no lo tiene, no se volverá a preguntar por el tiempo de este valor. Tamaño de 32 bits. El TTL se puede indicar para cada uno de los RRs o definir uno global, depende de la implementación. Este valor no es parte del SOA. En el caso de BIND 9, se define con la macro `$TTL`.

Ejemplo:

```
$TTL    604800 ; ### TTL global para todos
cities.org.       IN      SOA     berlin.cities.org. root.berlin.cities.org. (
                      2008092901         ; ## Serial
                          604800         ; ## Refresh
                           86400         ; ## Retry
                         2419200         ; ## Expiry
                          604800 )       ; ## Negative Cache TTL
```

El TTL se aplica de forma global, pero se puede re-escribir para registros en particular. Por ejemplo si no se desea que se haga caching por mucho tiempo se puede poner un tiempo de expiración de segundos. Por ejemplo:

```
# less /etc/bind/db.cities.org
...
paraguil-br0.cities.org.  30 IN        A    172.20.1.1
...
```

Google para los registros `A` utiliza en 2009 tiempos de 300 segundos que se va decrementando.

```
? dig +nocomment www.google.com
...
;www.google.com. IN A
www.google.com. 301240 IN CNAME www.l.google.com.
www.l.google.com. 4 IN A 209.85.195.104
www.l.google.com. 4 IN A 209.85.195.147
www.l.google.com. 4 IN A 209.85.195.99
...

? dig +nocomment www.google.com
...
;www.google.com. IN A
www.google.com. 301235 IN CNAME www.l.google.com.
www.l.google.com. 300 IN A 209.85.195.99
www.l.google.com. 300 IN A 209.85.195.104
www.l.google.com. 300 IN A 209.85.195.147
```

```
...

? dig +nocomment www.google.com
...
;www.google.com. IN A
www.google.com. 301231 IN CNAME www.l.google.com.
www.l.google.com. 296 IN A 209.85.195.104
www.l.google.com. 296 IN A 209.85.195.147
www.l.google.com. 296 IN A 209.85.195.99
...
```

4.4. Ejemplos con el servicio de DNS

Para los siguientes ejemplos se utilizarán los comandos `dig(1)`, `host(1)` y `nslookup(1)`.

4.4.1. Ejemplos de Consulta de Registros A

En los primeros casos se ve que se realiza una consulta recursiva y que el servidor responde como autoritativo. La captura queda almacenada en el archivo `captures/dns-a.pcap`. Se puede ver que las consultas realizadas por el comando `host(1)` preguntan, también, por el mapeo de nombre a dirección IPv6: registro **(AAAA)**.

```
# tcpdump -n -i lo -s 1500 -w captures/dns-a.pcap port 53 &

? host -t a berlin.cities.org 127.0.0.1

Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

berlin.cities.org has address 172.20.1.100

? dig -t a berlin.cities.org @127.0.0.1

; <<>> DiG 9.3.1 <<>> -t a berlin.cities.org @127.0.0.1
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 72
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
```

```
;berlin.cities.org.              IN      A

;; ANSWER SECTION:
berlin.cities.org.       604800  IN      A       172.20.1.100

;; AUTHORITY SECTION:
cities.org.              604800  IN      NS      berlin.cities.org.
cities.org.              604800  IN      NS      brasilia.cities.org.

;; ADDITIONAL SECTION:
berlin.cities.org.       604800  IN      A       172.20.1.100
brasilia.cities.org.     604800  IN      A       172.20.1.5

;; Query time: 12 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Sep 28 17:49:40 2008
;; MSG SIZE  rcvd: 120
```

A continuación se muestran otros ejemplos usando argumentos extras y diferentes comandos.

```
? dig +nocomments +nostats -t a brasilia.cities.org @127.0.0.1

; <<>> DiG 9.3.1 <<>> -t a brasilia.cities.org @127.0.0.1

;brasilia.cities.org.           IN      A
brasilia.cities.org.     604800  IN      A       172.20.1.5
cities.org.              604800  IN      NS      berlin.cities.org.
cities.org.              604800  IN      NS      brasilia.cities.org.
berlin.cities.org.       604800  IN      A       172.20.1.100
brasilia.cities.org.     604800  IN      A       172.20.1.5

? dig +short -t a brasilia.cities.org @127.0.0.1
172.20.1.5

? nslookup
> server 127.0.0.1

Default server: 127.0.0.1
Address: 127.0.0.1#53
> set type=A
> brasilia.cities.org
```

```
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:   brasilia.cities.org
Address: 172.20.1.5
> exit
```

Se puede contrastar una consulta iterativa con una recursiva. Si se realiza una consulta recursiva a algún servidor raíz se puede observar que responde con la lista de servidores y no resuelve el nombre que se le solicita. En la consulta se activa el flag de QD, pero no se activa en la respuesta el de QA. Si la tuviese cacheada, la podría resolver.

```
# tcpdump -n -i any -s 1500 -w dns-rec-norec.pcap port 53 &

? dig +recurse +short www.unlp.edu.ar @192.112.36.4
```

El resultado sería similar a forzar la iteración, en esta caso no se activará el flag de RD. Si se saca la opción `+short` se ve que retorna la lista de servidores delegados.

```
? dig +norecurse www.unlp.edu.ar @192.112.36.4
 <<>> DiG 9.3.1 <<>> +norecurse +nocomments www.unlp.edu.ar @192.112.36.4
; (1 server found)
;; global options:  printcmd
;www.unlp.edu.ar.               IN      A
ar.                     172800  IN      NS      MERAPI.SWITCH.CH.
ar.                     172800  IN      NS      UUCP-GW-2.PA.DEC.COM.
ar.                     172800  IN      NS      ATHEA.ar.
ar.                     172800  IN      NS      NS1.RETINA.ar.
ar.                     172800  IN      NS      NS-AR.RIPE.NET.
ar.                     172800  IN      NS      NS.UU.NET.
ar.                     172800  IN      NS      CTINA.ar.
ar.                     172800  IN      NS      UUCP-GW-1.PA.DEC.COM.
NS.UU.NET.              172800  IN      A       137.39.1.3
NS1.RETINA.ar.          172800  IN      A       200.10.202.3
...
```

Si se realiza la consulta al servidor local, éste responderá con la dirección IP solicitada, no con la lista de servidores.

```
? dig +recurse +short www.unlp.edu.ar @127.0.0.1
163.10.0.98
```

En la captura `captures/dns.pcap` se almacena la consulta al servidor local desde otro host de la red. En el mismo, se encuentran las consultas recursivas y las iterativas. Al inicio, se ve que el servidor de DNS se copia la lista de **ROOT Name Servers** desde uno de los que tiene configurado. El proceso se ilustra en la figura 8, salvo que no existe un **Servidor Forwarder** intermedio.

```
? host www.unlp.edu.ar @172.20.1.100
Using domain server:
Name: 172.20.1.100
Address: 172.20.1.100#53
Aliases:

www.unlp.edu.ar has address 163.10.0.98
```

Al ejecutar la consulta nuevamente, en la captura no se obtiene nada de tráfico de DNS porque ya está cacheada.

```
? dig +short www.unlp.edu.ar @127.0.0.1
163.10.0.98

? dig +short www.unlp.edu.ar @127.0.0.1
163.10.0.98

# tcpdump -n -i eth0 -s 1500 port 53
listening on eth0, link-type EN10MB (Ethernet), capture size 1500 bytes

0 packets captured
0 packets received by filter
0 packets dropped by kernel
```

4.4.2. Consulta de Registros PTR

A continuación se muestran consultas de reversos. En el archivo `captures/dns-ptr.pcap` se almacenan las capturas.

```
# tcpdump -n -i lo -s 1500 -w dns-ptr.pcap port 53 &

? host -t ptr 172.20.1.100 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

100.1.20.172.in-addr.arpa domain name pointer berlin.cities.org.
```

```
? dig +short -t ptr 100.1.20.172.in-addr.arpa  @127.0.0.1
berlin.cities.org.

? nslookup
> server 127.0.0.1

Default server: 127.0.0.1
Address: 127.0.0.1#53
> set type=PTR
> 100.1.20.172.in-addr.arpa

Server:         127.0.0.1
Address:        127.0.0.1#53

100.1.20.172.in-addr.arpa       name = berlin.cities.org.

> 172.20.1.100

Server:         127.0.0.1
Address:        127.0.0.1#53

100.1.20.172.in-addr.arpa       name = berlin.cities.org.
```

Algo similar sucede al momento de conectarse a un servidor vía TCP, por ejemplo, un servidor de SSH (Secure Shell).

```
# vi /etc/ssh/sshd_config
...
#UseDNS no
UseDNS yes # Es el default
...

# /etc/init.d/ssh reload

(172.20.1.1)# ssh 172.20.1.100
root@172.20.1.100's password: ****

# tcpdump -vvvv -n -i lo -s 1500 port 53
...
11:27:42.401014 IP (tos 0x0, ttl  64, id 27447, offset 0, flags [DF],
  proto: UDP (17), length: 69) 172.20.1.100.1029 > 172.20.1.100.53:
```

```
  64442+ PTR? 1.1.20.172.in-addr.arpa. (41)
11:27:42.404919 IP (tos 0x0, ttl  64, id 5, offset 0, flags [DF],
  proto: UDP (17), length: 143) 172.20.1.100.53 > 172.20.1.100.1029:
  64442* q: PTR? 1.1.20.172.in-addr.arpa. 1/1/1 1.1.20.172.in-addr.arpa.
  PTR paraguil-br0.cities.org. ns: 172.in-addr.arpa. NS berlin.cities.org.
  ar: berlin.cities.org. A 172.20.1.100 (115)
11:27:42.409989 IP (tos 0x0, ttl  64, id 27457, offset 0, flags [DF], ...
```

Luego se muestra un ejemplo de como se recorre el árbol en el dominio ARPA para resover un reverso. Se comienza escogiendo una IP y se resuelve en el local server.

```
? host anubis.unlp.edu.ar
anubis.unlp.edu.ar has address 163.10.0.65
anubis.unlp.edu.ar has IPv6 address 2800:340:0:64::65
```

```
? host 163.10.0.65
65.0.10.163.in-addr.arpa domain name pointer anubis.unlp.edu.ar.
```

Luego se muestra el recorrido que haría el local server comenzando por G.ROOT-SERVERS.NET.

```
? dig -t ptr +nocomments 65.0.10.163.in-addr.arpa @192.112.36.4

; <<>> DiG 9.9.5-3ubuntu0.10-Ubuntu <<>> -t ptr +nocomments
  65.0.10.163.in-addr.arpa @192.112.36.4
;; global options: +cmd
;65.0.10.163.in-addr.arpa. IN PTR
in-addr.arpa. 172800 IN NS a.in-addr-servers.arpa.
in-addr.arpa. 172800 IN NS b.in-addr-servers.arpa.
in-addr.arpa. 172800 IN NS d.in-addr-servers.arpa.
in-addr.arpa. 172800 IN NS f.in-addr-servers.arpa.
in-addr.arpa. 172800 IN NS c.in-addr-servers.arpa.
in-addr.arpa. 172800 IN NS e.in-addr-servers.arpa.
a.in-addr-servers.arpa. 172800 IN A 199.212.0.73
b.in-addr-servers.arpa. 172800 IN A 199.253.183.183
c.in-addr-servers.arpa. 172800 IN A 196.216.169.10
d.in-addr-servers.arpa. 172800 IN A 200.10.60.53
e.in-addr-servers.arpa. 172800 IN A 203.119.86.101
f.in-addr-servers.arpa. 172800 IN A 193.0.9.1
a.in-addr-servers.arpa. 172800 IN AAAA 2001:500:13::73
b.in-addr-servers.arpa. 172800 IN AAAA 2001:500:87::87
```

```
c.in-addr-servers.arpa. 172800 IN AAAA 2001:43f8:110::10
d.in-addr-servers.arpa. 172800 IN AAAA 2001:13c7:7010::53
e.in-addr-servers.arpa. 172800 IN AAAA 2001:dd8:6::101
f.in-addr-servers.arpa. 172800 IN AAAA 2001:67c:e0::1
;; Query time: 200 msec
;; SERVER: 192.112.36.4#53(192.112.36.4)
;; WHEN: Fri Mar 31 16:51:55 ART 2017
;; MSG SIZE  rcvd: 429
```

```
? dig -t ptr +nocomments 65.0.10.163.in-addr.arpa @199.212.0.73

; <<>> DiG 9.9.5-3ubuntu0.10-Ubuntu <<>> -t ptr +nocomments
  65.0.10.163.in-addr.arpa @199.212.0.73
;; global options: +cmd
;65.0.10.163.in-addr.arpa. IN PTR
163.in-addr.arpa. 86400 IN NS apnic.authdns.ripe.net.
163.in-addr.arpa. 86400 IN NS ns1.apnic.net.
163.in-addr.arpa. 86400 IN NS apnic1.dnsnode.net.
163.in-addr.arpa. 86400 IN NS ns4.apnic.net.
163.in-addr.arpa. 86400 IN NS tinnie.arin.net.
163.in-addr.arpa. 86400 IN NS ns3.apnic.net.
;; Query time: 165 msec
;; SERVER: 199.212.0.73#53(199.212.0.73)
;; WHEN: Fri Mar 31 16:52:05 ART 2017
;; MSG SIZE  rcvd: 204
```

```
? dig -t ptr +nocomments 65.0.10.163.in-addr.arpa @apnic.authdns.ripe.net

; <<>> DiG 9.9.5-3ubuntu0.10-Ubuntu <<>> -t ptr +nocomments
  65.0.10.163.in-addr.arpa @apnic.authdns.ripe.net
;; global options: +cmd
;65.0.10.163.in-addr.arpa. IN PTR
10.163.in-addr.arpa. 86400 IN NS unlp.unlp.edu.ar.
10.163.in-addr.arpa. 86400 IN NS anubis.unlp.edu.ar.
;; Query time: 232 msec
;; SERVER: 2001:67c:e0::9#53(2001:67c:e0::9)
;; WHEN: Fri Mar 31 16:52:19 ART 2017
;; MSG SIZE  rcvd: 104
```

```
? dig -t ptr +nocomments 65.0.10.163.in-addr.arpa @unlp.unlp.edu.ar

; <<>> DiG 9.9.5-3ubuntu0.10-Ubuntu <<>> -t ptr +nocomments
  65.0.10.163.in-addr.arpa @unlp.unlp.edu.ar
;; global options: +cmd
;65.0.10.163.in-addr.arpa. IN PTR
65.0.10.163.in-addr.arpa. 86400 IN PTR anubis.unlp.edu.ar.
0.10.163.in-addr.arpa. 86400 IN NS unlp.unlp.edu.ar.
0.10.163.in-addr.arpa. 86400 IN NS server.cespi.unlp.edu.ar.
0.10.163.in-addr.arpa. 86400 IN NS anubis.unlp.edu.ar.
unlp.unlp.edu.ar. 7200 IN A 163.10.0.67
unlp.unlp.edu.ar. 7200 IN AAAA 2800:340:0:64::67
anubis.unlp.edu.ar. 7200 IN A 163.10.0.65
anubis.unlp.edu.ar. 7200 IN AAAA 2800:340:0:64::65
server.cespi.unlp.edu.ar. 86400 IN A 163.10.0.84
server.cespi.unlp.edu.ar. 86400 IN AAAA 2800:340:0:64::84
;; Query time: 7 msec
;; SERVER: 2800:340:0:64::67#53(2800:340:0:64::67)
;; WHEN: Fri Mar 31 16:52:30 ART 2017
;; MSG SIZE  rcvd: 277


? dig -t ptr +short +nocomments 65.0.10.163.in-addr.arpa @unlp.unlp.edu.ar
anubis.unlp.edu.ar.
```

4.4.3. Consulta de Registros CNAME

Es posible realizar consultas por registros CNAME. Las capturas de estas consultas se pueden ver en los archivos `captures/dns-cname.pcap` y `captures/dns-cname-dig.pcap`.

```
# tcpdump -n -i lo -s 1500 -w dns-cname.pcap port 53 &
? host www.cities.org 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

www.cities.org is an alias for berlin.cities.org.
berlin.cities.org has address 172.20.1.100

# tcpdump -n -i lo -s 1500 -w dns-cname-dig.pcap port 53 &
? dig +short ftp.cities.org 127.0.0.1
```

```
berlin.cities.org.
172.20.1.100
```

4.4.4. Consulta de Registros MX y NS

A continuación se muestran consultas al servidor local sobre los Mail Exchangers para el dominio propio. Las capturas se almacenan en el archivo `captures/dns-mx.pcap`.

```
# tcpdump -n -i lo -s 1500 -w dns-mx.pcap port 53 &
? host -t mx cities.org 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

cities.org mail is handled by 10 berlin.cities.org.
cities.org mail is handled by 1 brasilia.cities.org.
? dig +short -t mx cities.org @127.0.0.1
10 berlin.cities.org.
1 brasilia.cities.org.

? dig +nocomments +nostats -t mx cities.org @127.0.0.1

; <<>> DiG 9.3.1 <<>> +nocomments +nostats -t mx cities.org @127.0.0.1
; (1 server found)
;; global options:  printcmd
;cities.org.                    IN      MX
cities.org.             604800  IN      MX      1 brasilia.cities.org.
cities.org.             604800  IN      MX      10 berlin.cities.org.
cities.org.             604800  IN      NS      berlin.cities.org.
cities.org.             604800  IN      NS      brasilia.cities.org.
brasilia.cities.org.    604800  IN      A       172.20.1.5
berlin.cities.org.      604800  IN      A       172.20.1.100
```

El siguiente es un ejemplo de consultas de los Mail Exchangers para el dominios externos. Archivo de captura: `captures/dns-mx-ext.pcap`.

```
# tcpdump -n -i any -s 1500 -w dns-mx-ext.pcap port 53 &
? host -t mx gmail.com 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:
```

```
gmail.com mail is handled by 10 alt2.gmail-smtp-in.l.google.com.
gmail.com mail is handled by 50 gsmtp147.google.com.
gmail.com mail is handled by 50 gsmtp183.google.com.
gmail.com mail is handled by 5 gmail-smtp-in.l.google.com.
gmail.com mail is handled by 10 alt1.gmail-smtp-in.l.google.com.

? dig +nocomments +nostats -t mx gmail.com @127.0.0.1

; <<>> DiG 9.3.1 <<>> +nocomments +nostats -t mx gmail.com @127.0.0.1
; (1 server found)
;; global options:  printcmd
;gmail.com.                     IN      MX
gmail.com.              3177    IN      MX      10 alt2.gmail-smtp-in.l.google.com.
gmail.com.              3177    IN      MX      50 gsmtp147.google.com.
gmail.com.              3177    IN      MX      50 gsmtp183.google.com.
gmail.com.              3177    IN      MX      5 gmail-smtp-in.l.google.com.
gmail.com.              3177    IN      MX      10 alt1.gmail-smtp-in.l.google.com.
gmail.com.              345177  IN      NS      ns1.google.com.
gmail.com.              345177  IN      NS      ns2.google.com.
gmail.com.              345177  IN      NS      ns3.google.com.
gmail.com.              345177  IN      NS      ns4.google.com.
ns1.google.com.         172377  IN      A       216.239.32.10
ns2.google.com.         172377  IN      A       216.239.34.10
ns3.google.com.         172377  IN      A       216.239.36.10
ns4.google.com.         172377  IN      A       216.239.38.10
```

Los siguientes son ejemplos de consultas de servidores DNS para el dominio interno y para un dominio externo. Archivos de captura: `captures/dns-ns.pcap` y `captures/dns-ns-ext.pcap`.

```
# tcpdump -n -i lo -s 1500 -w dns-ns.pcap port 53 &
? host -t ns cities.org 127.0.0.1

Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

cities.org name server brasilia.cities.org.
cities.org name server berlin.cities.org.

? host -t ns cities.org 127.0.0.1
```

```
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

cities.org name server berlin.cities.org.
cities.org name server brasilia.cities.org.

? host -t ns trees.cities.org 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

trees.cities.org name server oak.trees.cities.org.
trees.cities.org name server brasilia.cities.org.
trees.cities.org name server berlin.cities.org

? host -t ns trees.cities.org 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

trees.cities.org name server brasilia.cities.org.
trees.cities.org name server berlin.cities.org.
trees.cities.org name server oak.trees.cities.org.

? host -t ns trees.cities.org 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

trees.cities.org name server oak.trees.cities.org.
trees.cities.org name server brasilia.cities.org.
trees.cities.org name server berlin.cities.org.

? dig +short -t ns trees.cities.org @127.0.0.1
oak.trees.cities.org.
brasilia.cities.org.
berlin.cities.org.
```

```
? dig +nocomments +nostats -t ns trees.cities.org @127.0.0.1

; <<>> DiG 9.3.1 <<>> +nocomments +nostats -t ns trees.cities.org @127.0.0.1
; (1 server found)
;; global options:  printcmd
;trees.cities.org.              IN      NS
trees.cities.org.       604800  IN      NS      brasilia.cities.org.
trees.cities.org.       604800  IN      NS      berlin.cities.org.
trees.cities.org.       604800  IN      NS      oak.trees.cities.org.
brasilia.cities.org.    604800  IN      A       172.20.1.5
berlin.cities.org.      604800  IN      A       172.20.1.100
oak.trees.cities.org.   604800  IN      A       192.168.40.1
```

Las consultas sobre servidores externos.

```
# tcpdump -n -i any -s 1500 -w dns-ns-ext.pcap port 53 &
? host -t ns gmail.com 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

gmail.com name server ns3.google.com.
gmail.com name server ns4.google.com.
gmail.com name server ns1.google.com.
gmail.com name server ns2.google.com.

? dig +short -t ns gmail.com @127.0.0.1
ns4.google.com.
ns1.google.com.
ns2.google.com.
ns3.google.com.

? dig +nocomments +nostats -t ns gmail.com @127.0.0.1

; <<>> DiG 9.3.1 <<>> +nocomments +nostats -t ns gmail.com @127.0.0.1
; (1 server found)
;; global options:  printcmd
;gmail.com.                     IN      NS
gmail.com.              344734  IN      NS      ns3.google.com.
gmail.com.              344734  IN      NS      ns4.google.com.
```

```
gmail.com.              344734  IN      NS      ns1.google.com.
gmail.com.              344734  IN      NS      ns2.google.com.
ns3.google.com.         171934  IN      A       216.239.36.10
ns4.google.com.         171934  IN      A       216.239.38.10
ns1.google.com.         171934  IN      A       216.239.32.10
ns2.google.com.         171934  IN      A       216.239.34.10
```

4.4.5. Transferencias de Zonas

Los siguientes son ejemplos de transferencias de zonas forzadas, no realizadas automáticamente entre los servidores de DNS. Se puede observar en la captura `captures/dns-axfr.pcap` que se utiliza TCP para hacer la transferencia. Este proceso de copia de los datos debería realizarse de forma automática al indicar el servidor maestro a los secundarios que se produjo un cambio.

```
# tcpdump -n -i lo -s 1500 -w dns-axfr.pcap port 53 &
? host -t axfr cities.org 127.0.0.1
Trying "cities.org"
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40298
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;cities.org.                    IN      AXFR

;; ANSWER SECTION:
cities.org.             604800  IN      SOA
    localhost. root.localhost. 1 604800 86400 2419200 604800
...
cities.org.             604800  IN      NS      berlin.cities.org.
...
cities.org.             604800  IN      NS      brasilia.cities.org.
...
cities.org.             604800  IN      MX      1 brasilia.cities.org.
...
Received 72 bytes from 127.0.0.1#53 in 137 ms

? dig +short -t axfr cities.org @127.0.0.1
localhost. root.localhost. 1 604800 86400 2419200 604800
```

```
berlin.cities.org.
brasilia.cities.org.
1 brasilia.cities.org.
10 berlin.cities.org.
berlin.cities.org.
172.20.1.1
172.20.1.100
"Pentium-II/256" "GNU/Linux"
"Servidor Web y FTP"
berlin.cities.org.
172.20.1.5
"MIPS-R10000/512" "IRIX6.5"
brasilia.cities.org.
berlin.cities.org.
oak.trees.cities.org.
192.168.40.1
localhost. root.localhost. 1 604800 86400 2419200 604800

? dig +nocomments +nostats -t axfr cities.org @127.0.0.1

; <<>> DiG 9.3.1 <<>> +nocomments +nostats -t axfr cities.org @127.0.0.1
; (1 server found)
;; global options:  printcmd
cities.org.             604800  IN      SOA     localhost. root.localhost.
                                                1 604800 86400 2419200 604800
cities.org.             604800  IN      NS      berlin.cities.org.
cities.org.             604800  IN      NS      brasilia.cities.org.
cities.org.             604800  IN      MX      1 brasilia.cities.org.
cities.org.             604800  IN      MX      10 berlin.cities.org.
ftp.cities.org.         604800  IN      CNAME   berlin.cities.org.
paraguil-br0.cities.org. 604800 IN      A       172.20.1.1
berlin.cities.org.      604800  IN      A       172.20.1.100
berlin.cities.org.      604800  IN      HINFO   "Pentium-II/256" "GNU/Linux"
berlin.cities.org.      604800  IN      TXT     "Servidor Web y FTP"
www.cities.org.         604800  IN      CNAME   berlin.cities.org.
brasilia.cities.org.    604800  IN      A       172.20.1.5
brasilia.cities.org.    604800  IN      HINFO   "MIPS-R10000/512" "IRIX6.5"
trees.cities.org.       604800  IN      NS      brasilia.cities.org.
trees.cities.org.       604800  IN      NS      berlin.cities.org.
trees.cities.org.       604800  IN      NS      oak.trees.cities.org.
oak.trees.cities.org.   604800  IN      A       192.168.40.1
cities.org.             604800  IN      SOA     localhost. root.localhost.
```

```
                                        1 604800 86400 2419200 604800
```

Si se intenta realizar una transferencia a un servidor que no permite hacer la copia desde cualquier lado se obtendrá un error. El servidor destino indicará que no acepta la conexión DNS, luego el cliente cierra la conexión TCP. Ver archivo `captures/dns-axfr-error.pcap`.

```
# tcpdump -n -i eth0 -s 1500 -w dns-axfr-error.pcap port 53

? dig +nocomments +nostats -t axfr att.com @144.160.128.140

; <<>> DiG 9.3.1 <<>> +nocomments +nostats -t axfr att.com @144.160.128.140
; (1 server found)
;; global options:  printcmd
; Transfer failed.
```

4.4.6. Transferencias de Zonas entre Master y Secundarios

En los ejemplos anteriores se mostraron transferencias de zonas forzadas por el usuario. En los siguientes ejemplos se muestra la convivencia entre un servidor Master y un Slave o Secundario. Una vez re-configurado el servidor Primario o Master y configurado el secundario, se pueden ver los ejemplos. Al final, se encuentran las configuraciones de master y slave. Primero, se inicia el master y se ve el intento de sincronizar. El slave no está ejecutando aún.

```
root@berlin:/etc/bind# /etc/init.d/bind stop
root@berlin:/etc/bind# vi named.conf.local
root@berlin:/etc/bind# /etc/init.d/bind start
Starting domain name service: named.

root@berlin:/etc/bind# tail -f /var/log/syslog | grep named
Nov 16 23:39:20 berlin named[5228]: hint zone "" (IN) loaded (serial 0)
Nov 16 23:39:20 berlin named[5228]: master zone "localhost" (IN) loaded (serial 1)
Nov 16 23:39:20 berlin named[5228]: master zone "127.in-addr.arpa" (IN) loaded (serial 1)
Nov 16 23:39:20 berlin named[5228]: master zone "0.in-addr.arpa" (IN) loaded (serial 1)
Nov 16 23:39:20 berlin named[5228]: master zone "255.in-addr.arpa" (IN) loaded (serial 1)
Nov 16 23:39:20 berlin named[5228]: master zone "cities.org" (IN) loaded (serial 1)
Nov 16 23:39:20 berlin named[5228]: master zone "172.in-addr.arpa" (IN) loaded (serial 1)
Nov 16 23:39:20 berlin named[5228]: listening on [127.0.0.1].53 (lo)
Nov 16 23:39:20 berlin named[5228]: listening on [172.20.1.100].53 (eth0)
Nov 16 23:39:20 berlin named[5228]: listening on [10.20.1.100].53 (eth0:loc)
Nov 16 23:39:20 berlin named[5228]: Forwarding source address is [::].1028
Nov 16 23:39:20 berlin named[5228]: Forwarding source address is [0.0.0.0].1029
Nov 16 23:39:20 berlin named[5229]: Ready to answer queries.
Nov 16 23:39:37 berlin named[5229]: Sent NOTIFY for "cities.org IN SOA 1" (cities.org);
```

```
                        1 NS, 1 A
Nov 16 23:39:48 berlin named[5229]: Err/TO getting serial# for "trees.cities.org"
Nov 16 23:39:48 berlin named-xfer[5230]: connect([172.20.1.5].53) for zone trees.cities.org
                        failed: Connection refused

root@brasilia:/etc/bind# tcpdump -n -i any -s 1500 -w dns-slave-listen-no-run.pcap port 53
```

Ahora se inicia el slave y se produce la sincronización.

```
root@brasilia:/etc/bind# /etc/init.d/bind start
Starting domain name service: named.

root@brasilia:/etc/bind# tail -f /var/log/syslog | grep named
Nov 16 23:42:45 brasilia named[5258]: hint zone "" (IN) loaded (serial 0)
Nov 16 23:42:45 brasilia named[5258]: master zone "localhost" (IN) loaded (serial 1)
Nov 16 23:42:45 brasilia named[5258]: master zone "127.in-addr.arpa" (IN) loaded (serial 1)
Nov 16 23:42:45 brasilia named[5258]: master zone "0.in-addr.arpa" (IN) loaded (serial 1)
Nov 16 23:42:45 brasilia named[5258]: master zone "255.in-addr.arpa" (IN) loaded (serial 1)
Nov 16 23:42:45 brasilia named[5258]: master zone "trees.cities.org" (IN) loaded (serial 1)
Nov 16 23:42:45 brasilia named[5258]: listening on [127.0.0.1].53 (lo)
Nov 16 23:42:45 brasilia named[5258]: listening on [172.20.1.5].53 (eth0)
Nov 16 23:42:45 brasilia named[5258]: listening on [10.20.1.5].53 (eth0:loc)
Nov 16 23:42:45 brasilia named[5258]: Forwarding source address is [::].1025
Nov 16 23:42:45 brasilia named[5258]: Forwarding source address is [0.0.0.0].1026
Nov 16 23:42:45 brasilia named[5259]: Ready to answer queries.
Nov 16 23:42:45 brasilia named[5259]: sysquery: sendto([192.58.128.30].53):
                Network is unreachable
Nov 16 23:42:46 brasilia named-xfer[5260]: send AXFR query 0 to [172.20.1.100].53 for
                cities.org
Nov 16 23:42:46 brasilia named[5259]: slave zone "cities.org" (IN) loaded (serial 1)
Nov 16 23:43:01 brasilia named[5259]: Sent NOTIFY for "trees.cities.org IN SOA 1"
                (trees.cities.org); 1 NS, 1 A
Nov 16 23:43:01 brasilia named[5259]: Received NOTIFY answer (AA) from 172.20.1.100 for
                "trees.cities.org IN SOA"
Nov 16 23:43:01 brasilia named[5259]: approved AXFR from [172.20.1.100].4289 for
                "trees.cities.org"
Nov 16 23:43:01 brasilia named[5259]: zone transfer (AXFR) of "trees.cities.org" (IN) to
                [172.20.1.100].4289 serial 1
```

En los logs del master queda registrada la transferencia de zona.

```
root@berlin:/etc/bind# tail -f /var/log/syslog | grep named
Nov 16 23:42:40 berlin named[5229]: approved AXFR from [172.20.1.5].2506 for "cities.org"
```

```
Nov 16 23:42:40 berlin named[5229]: zone transfer (AXFR) of "cities.org" (IN) to
                [172.20.1.5].2506 serial 1
Nov 16 23:42:56 berlin named[5229]: rcvd NOTIFY(trees.cities.org, IN, SOA) from
                [172.20.1.5].1026
Nov 16 23:42:56 berlin named[5229]: rcvd NOTIFY(trees.cities.org, IN, SOA) from
                [172.20.1.5].1026
Nov 16 23:42:56 berlin named[5229]: NOTIFY(SOA) for zone already xferring
                (trees.cities.org)
Nov 16 23:42:56 berlin named-xfer[5231]: send AXFR query 0 to [172.20.1.5].53 for
                trees.cities.org
Nov 16 23:42:56 berlin named[5229]: slave zone "trees.cities.org" (IN) loaded (serial 1)
Nov 16 23:43:19 berlin named[5229]: rcvd NOTIFY(cities.org, IN, SOA) from [172.20.1.5].1026
Nov 16 23:43:19 berlin named[5229]: NOTIFY(SOA) for non-slave zone (cities.org), from
                [172.20.1.5].1026
Nov 16 23:43:19 berlin named[5229]: rcvd NOTIFY(cities.org, IN, SOA) from
                [172.20.1.5].1026
Nov 16 23:43:19 berlin named[5229]: NOTIFY(SOA) for non-slave zone (cities.org), from
                [172.20.1.5].1026
Nov 16 23:43:24 berlin named[5229]: Sent NOTIFY for "trees.cities.org IN SOA 1"
                (trees.cities.org); 1 NS, 1 A
```

En el servidor secundario se generan, una vez finalizada la transferencia, los archivos de cache.

```
root@brasilia:/etc/bind# ls /var/cache/bind/
db.cities.org

textcolorgreenroot@brasilia:/etc/bind# cat /var/cache/bind/db.cities.org
...
cities  604800  IN      SOA     localhost. root.localhost. (
                1 604800 86400 2419200 604800 )
        604800  IN      NS      berlin.cities.org.
        604800  IN      NS      brasilia.cities.org.
        604800  IN      MX      1 brasilia.cities.org.
        604800  IN      MX      10 berlin.cities.org.
...
ftp     604800  IN      CNAME   berlin.cities.org.
paraguil-br0    604800  IN      A       172.20.1.1
berlin  604800  IN      A       172.20.1.100
        604800  IN      HINFO   "Pentium-II/256" "GNU/Linux"
        604800  IN      TXT     "Servidor Web y FTP"
...
```

```
root@berlin:/etc/bind# ls /var/cache/bind/
db.trees.cities.org

root@brasilia:/etc/bind# tcpdump -n -i any -s 1500 -w dns-slave-start.pcap port 53
```

Para mostrar el ejemplo de una notificación, se puede generar un cambio en el archivo maestro. Se envía la señal HangUp (HUP) al proceso named para que lea, nuevamente, su archivo de configuración y se entere de los cambios. Esto depende el OS, de forma más genérica se puede utilizar el comando `rndc(8)`. Los slaves o secundarios son notificados de las modificaciones por el master y se fuerza la transferencia de zona.

```
root@berlin:/etc/bind# cp db.cities.org db.cities.org.bak
root@berlin:/etc/bind# vi db.cities.org
root@berlin:/etc/bind# diff db.cities.org db.cities.org.bak
5,6c5,6
< @     IN      SOA     localhost. root.berlin.cities.org. (
<                               2         ; Serial
---
> @     IN      SOA     localhost. root.localhost. (
>                               1         ; Serial
34d33
< paraguil-br1.cities.org.  IN        A    172.20.0.1

root@berlin:/etc/bind# killall -HUP named / rndc reload

root@berlin:/etc/bind# tail -f /var/log/syslog | grep named
Nov 16 23:53:39 berlin named[5229]: reloading nameserver
Nov 16 23:53:39 berlin named[5229]: master zone "cities.org" (IN) loaded (serial 3)
Nov 16 23:53:39 berlin named[5229]: Forwarding source address is [::].1032
Nov 16 23:53:39 berlin named[5229]: Forwarding source address is [0.0.0.0].1033
Nov 16 23:53:39 berlin named[5229]: Ready to answer queries.
Nov 16 23:53:39 berlin named[5229]: reloading nameserver
Nov 16 23:53:39 berlin named[5229]: master zone "cities.org" (IN) loaded (serial 3)
Nov 16 23:53:39 berlin named[5229]: Forwarding source address is [::].1032
Nov 16 23:53:39 berlin named[5229]: Forwarding source address is [0.0.0.0].1033
Nov 16 23:53:39 berlin named[5229]: Ready to answer queries.
...
Nov 16 23:54:31 berlin named[5229]: Sent NOTIFY for "cities.org IN SOA 3" (cities.org);
                1 NS, 1 A
Nov 16 23:54:31 berlin named[5229]: Received NOTIFY answer (AA) from 172.20.1.5 for
                "cities.org IN SOA"
Nov 16 23:54:31 berlin named[5229]: approved AXFR from [172.20.1.5].2671 for "cities.org"
```

```
Nov 16 23:54:31 berlin named[5229]: zone transfer (AXFR) of "cities.org" (IN) to
                [172.20.1.5].2671 serial 3
Nov 16 23:54:31 berlin named[5229]: Sent NOTIFY for "cities.org IN SOA 3" (cities.org);
                1 NS, 1 A
Nov 16 23:54:31 berlin named[5229]: Received NOTIFY answer (AA) from 172.20.1.5 for
                "cities.org IN SOA"
Nov 16 23:54:31 berlin named[5229]: approved AXFR from [172.20.1.5].2671 for "cities.org"
Nov 16 23:54:31 berlin named[5229]: zone transfer (AXFR) of "cities.org" (IN) to
                [172.20.1.5].2671 serial 3

root@brasilia:/etc/bind# tcpdump -n -i any -s 1500 -w dns-master-slave-xfer.pcap port 53
```

4.4.7. Consultas por TCP/Flag TC

De acuerdo a la cantidad de información a transmitir se utilizará TCP o UDP, siendo en la mayoría de los caso UDP la elección. Se puede forzar el uso TCP desde el cliente en los comandos, por ejemplo la opción `+tcp` del comando `dig(1)`, o si la cantidad de información a transmitir no entra en un mensaje de DNS el servidor indiciará el uso de TCP mediante el flag: TC. A continuación se muestra el ejemplo y se indica la captura.

```
root@berlin:/tmp# tcpdump -n -i eth0 -s 1500 -w dns-truncate-host.pcap port 53
# less /etc/bind/db.cities.org
...

berlin.cities.org.        IN    TXT      "Servidor Web y FTP"
brasilia.cities.org.      IN    TXT      ("AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                                          "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA")

? host -t txt brasilia.cities.org 127.0.0.1
;; Truncated, retrying in TCP mode.
Using domain server:
```

```
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

brasilia.cities.org descriptive text "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA"
```

Resultado similar se obtendrá con el comando `dig(1)`.

```
? dig -t txt brasilia.cities.org @127.0.0.1
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.3.1 <<>> -t txt brasilia.cities.org @127.0.0.1
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61661
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;brasilia.cities.org. IN TXT

;; ANSWER SECTION:
brasilia.cities.org. 604800 IN TXT "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
...

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Apr  3 12:46:52 2017
;; MSG SIZE  rcvd: 848
```

4.4.8. Dynamic DNS (DDNS)

La actualización de los DNS de forma dinámica se estandarizó con [RFC-2136]. En la actualidad, existen varios servicios de DDNS (e.g. www.dyndns.org) que permiten utilizar una conexión SOHO (Small

Office - Home Office) con dirección IP pública, pero dinámica, rotada por el ISP (e.g. ADSL, Cable-Modém). Esto es posible gracias a la capacidad de actualización dinámica de un servidor de DNS. A continuación se muestran algunos ejemplos de mensajes dinámicos sobre los servidores de DNS de prueba. Primero se muestra el envío de una actualización que se deniega debido a la falta de privilegios.

```
root@berlin:/tmp# tcpdump -n -i eth0 -s 1500 -w dns-update-NO.pcap port 53
```

Desde el cliente.

```
? nsupdate -v update.cities.org
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:      0
;; flags: ; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
;; ZONE SECTION:
;cites.org.                     IN      SOA

;; UPDATE SECTION:
paraguil-br0.cities.org. 0      NONE    A       172.20.1.1
paraguil-br0.cities.org. 86400  IN      A       172.20.1.7

update failed: NOTAUTH
update failed: NOTAUTH
```

Si se configuran adecuadamente las claves correspondientes para las actualizaciones, este proceso es posible. Antes, se deben crear las claves en el servidor.

```
root@berlin:/etc/bind# dnskeygen -H 128 -n key-cities -h
** Adding dot to the name to make it fully qualified domain name**
Generating 128 bit HMAC-MD5 Key for key-cities.

Generated 128 bit Key for key-cities. id=0 alg=157 flags=513

root@berlin:/etc/bind#root@berlin:/etc/bind# cat Kkey-cities.+157+00000.key
key-cities. IN KEY 513 3 157 Vpfdgg8WirY2I2m4Af/pjg==
root@berlin:/etc/bind# cat Kkey-cities.+157+00000.private
Private-key-format: v1.2
Algorithm: 157 (HMAC)
Key: Vpfdgg8WirY2I2m4Af/pjg==

root@berlin:/etc/bind#root@berlin:/etc/bind# ls -l Kkey*
-rw-------  1 root root 54 2008-11-17 10:18 Kkey-cities.+157+00000.key
-rw-------  1 root root 77 2008-11-17 10:18 Kkey-cities.+157+00000.private
```

```
root@berlin:/etc/bind#root@berlin:/etc/bind# /etc/init.d/bind restart
...
Nov ...: dynamic zone file '/etc/bind/db.cities.org' is writable
...
```

También se debe reconfigurar el servidor para permitir actualizaciones con la clave generada. Es posible usar un esquema de clave pública/clave privada, aunque en este caso no se utiliza. Sólo se usa un digest o HMAC MD5.

```
root@berlin:/etc/bind# cat named.conf.local
...
key "key-cities" {
  algorithm hmac-md5;
  secret "Vpfdgg8WirY2I2m4Af/pjg==";
};

zone "cities.org" {
        type master;
        file "/etc/bind/db.cities.org";
        allow-query { any; };
        allow-transfer { 172.20.1.5; 172.20.1.1; };
        allow-update { key key-cities; };
        //allow-update { 172.20.1.1; };
};
...
```

```
root@berlin:/etc/bind# chmod 600 db.cities.org
root@berlin:/etc/bind# ls -l db.cities.org
-rw-------  1 root root 1337 2008-11-17 11:11 db.cities.org

root@berlin:/tmp# tcpdump -n -i eth0 -s 1500 -w dns-update-OK.pcap port 53
```

Con el servidor configurado correctamente y funcionando, se lanza la actualización en el cliente.

```
? vi Kkey-cities.+157+00000.key
key-cities. IN KEY 513 3 157 Vpfdgg8WirY2I2m4Af/pjg=

? nsupdate -k Kkey-cities.+157+00000.key
> server 172.20.1.100
> zone cities.org.
> update add    paraguil-br0.cities.org.  86400        A    172.20.1.7
> send
> update delete paraguil-br0.cities.org.  A 172.20.1.1
```

```
> show
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:      0
;; flags: ; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
;; ZONE SECTION:
;cities.org.                    IN      SOA

;; UPDATE SECTION:
paraguil-br0.cities.org. 0      NONE    A       172.20.1.1

> send
> quit

? host paraguil-br0.cities.org. 172.20.1.100
Using domain server:
Name: 172.20.1.100
Address: 172.20.1.100#53
Aliases:

paraguil-br0.cities.org has address 172.20.1.7
```

En el servidor, se pueden observar los cambios producidos.

```
root@berlin:/etc/bind# cat db.cities.org.log
;BIND LOG V8
[DYNAMIC_UPDATE] id 30487 from [172.20.1.1].32849 at 1226931659 (named pid 5306):
zone:   origin cities.org class IN serial 3
update: add paraguil-br0.cities.org. 86400 IN A 172.20.1.7

[DYNAMIC_UPDATE] id 31914 from [172.20.1.1].32849 at 1226931734 (named pid 5306):
zone:   origin cities.org class IN serial 3
update: delete paraguil-br0.cities.org. IN A 172.20.1.1

[INCR_SERIAL] from 3 to 4 Mon Nov 17 11:23:13 2008
```

Después de reiniciar el servicio, el log se rota, y los cambios se incorporan en el archivo de zona.

```
root@berlin:/etc/bind# /etc/init.d/bind restart
Stopping domain name service: named.
Starting domain name service: named.

root@berlin:/etc/bind# diff db.cities.org db.cities.org.bak2
1,18c1,45
```

```
< ;BIND DUMP V8
< $ORIGIN org.
< cities        604800  IN      NS      berlin.cities.org.      ;Cl=2
<       604800  IN      NS      brasilia.cities.org.    ;Cl=2
<       604800  IN      SOA     localhost. root.berlin.cities.org. (
<               4 604800 86400 2419200 604800 ) ;Cl=2
<       604800  IN      MX      1 brasilia.cities.org.  ;Cl=2
<       604800  IN      MX      10 berlin.cities.org.   ;Cl=2
< $ORIGIN cities.org.
< berlin        604800  IN      A       172.20.1.100    ;Cl=2
<       604800  IN      HINFO   "Pentium-II/256" "GNU/Linux"    ;Cl=2
<       604800  IN      TXT     "Servidor Web y FTP"    ;Cl=2
< brasilia      604800  IN      A       172.20.1.5      ;Cl=2
<       604800  IN      HINFO   "MIPS-R10000/512" "IRIX6.5"     ;Cl=2
< www   604800  IN      CNAME   berlin.cities.org.      ;Cl=2
< paraguil-br0  86400   IN      A       172.20.1.7      ;Cl=2
< paraguil-br1  604800  IN      A       172.20.0.1      ;Cl=2
< ftp   604800  IN      CNAME   berlin.cities.org.      ;Cl=2
---
> ;
> ; BIND reverse data file for cities.org
> ;
> $TTL  604800
> @     IN      SOA     localhost. root.berlin.cities.org. (
>                             3         ; Serial
>                        604800         ; Refresh
>                         86400         ; Retry
>                       2419200         ; Expire
>                        604800 )       ; Negative Cache TTL
> ;
> ; @                      IN         NS berlin.cities.org.
>
> $ORIGIN .
>
> cities.org.           IN         NS berlin.cities.org.
> cities.org.           IN         NS brasilia.cities.org.
>
> ; ## BRAISLIA NO NECESITA GLUE RECORD ##
> trees.cities.org.       IN         NS brasilia.cities.org.
> trees.cities.org.       IN         NS berlin.cities.org.
> trees.cities.org.       IN         NS oak.trees.cities.org.
>
```

```
> ; ## GLUE RECORD ##
> oak.trees.cities.org.   IN         A 192.168.40.1
>
> cities.org.               IN       MX 1   brasilia.cities.org.
> cities.org.               IN       MX 10  berlin.cities.org.
>
>
> berlin.cities.org.        IN        A    172.20.1.100
> brasilia.cities.org.      IN        A    172.20.1.5
> paraguil-br0.cities.org.  IN        A    172.20.1.1
> paraguil-br1.cities.org.  IN        A    172.20.0.1
>
> ftp.cities.org.           IN    CNAME    berlin.cities.org.
> www.cities.org.           IN    CNAME    berlin.cities.org.
>
> berlin.cities.org.        IN    HINFO    Pentium-II/256  GNU/Linux
> brasilia.cities.org.      IN    HINFO    MIPS-R10000/512 IRIX6.5
>
>
> berlin.cities.org.        IN    TXT      "Servidor Web y FTP"
>
> ; ### EOF ###

root@berlin:/etc/bind# ls *.log
ls: *.log: No such file or directory
```

4.4.9. Split DNS

Un Split o Stealth DNS es un servicio que tiene diferentes "personalidades" (responde con diferente información) de acuerdo al cliente que lo consulta o servidor que lo utiliza. Con el servicio de BIND utilizado en estos ejemplos se puede implementar a través de vistas: `view`. Este facilidad está sólo disponible a partir del BIND 9.X. A continuación se muestra un ejemplo en el cual a los sistemas de una red local se les responde con una información y, a los demás, con otra. Esto puede ser útil si se tiene una intranet con servidores que pueden ser de acceso público (mediante NAT, por ejemplo) y, también, de forma local desde la red privada. Es importante notar que, para evitar configuraciones erroneas todas las zonas deben estar definidas en todas las vistas. En las versiones nuevas de BIND este chequeo se hace al arrancar y si no esta de forma correcta no funciona el servicio.

```
root@berlin:/etc/bind# cat named.conf.local
...
acl "internal"        { 172.20.0.0/16; 127.0.0.0/8; };
```

```
view "local" {
 match-clients { "internal"; };
 recursion yes;
 zone "cities.org" {
        type master;
        file "/etc/bind/db.cities.org";
        allow-query { any; };
        allow-transfer { 172.20.1.5; 172.20.1.1; };
        allow-update { key key-cities; };
        //allow-update { 172.20.1.1; };
 };
};

view "public"
{
 match-clients { any; };
 recursion no;
   zone "cities.org"
   {
        type master;
        // Direcciones de Host públicos solamente
        file "/etc/bind/db.external.cities.org";
        allow-query { any; };
        allow-transfer { none; };
   };
};
...
```

Y el archivo de zona para la vista externa será similar al siguiente.

```
root@berlin:/etc/bind# cat db.external.cities.org
;
; BIND reverse data file for External: cities.org
;
$TTL    604800
@       IN      SOA     localhost. root.berlin.cities.org. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
; @                       IN         NS berlin.cities.org.

$ORIGIN .
```

```
cities.org.             IN        NS berlin.cities.org.
cities.org.             IN        NS brasilia.cities.org.

trees.cities.org.       IN        NS brasilia.cities.org.
trees.cities.org.       IN        NS berlin.cities.org.

cities.org.               IN      MX 1   brasilia.cities.org.
cities.org.               IN      MX 10  berlin.cities.org.

berlin.cities.org.    IN A 200.1.1.201
voluntad.cities.org.  IN A 200.1.1.200
bangkok.cities.org.   IN A 200.1.1.5
brasilia.cities.org.  IN A 200.1.1.49
paraguil.cities.org.  IN A 200.1.1.50

ftp.cities.org.           IN    CNAME    berlin.cities.org.
www.cities.org.           IN    CNAME    berlin.cities.org.

; ### EOF ###
```

Al realizar las consultas desde diferentes puntos se puede observar que el servidor responderá con diferentes datos.

```
external? dig +short brasilia.cities.org @200.1.1.201
200.1.1.49

root@berlin:/etc/bind# dig +short brasilia.cities.org @172.20.1.100
172.20.1.5
```

4.4.10. Delegación de Reverso con CIDR

El dominio de los reversos in-addr.arpa fue pensado de forma classful. Hoy en día, el direccionamiento classful ha sido reemplazado por el CIDR, convirtiéndolo en classless. Para poder delegar un sub-dominio de una subred en el dominio de los reversos, [RFC-2317] define como hacerlo. Lo que se utiliza es un registro CNAME para poder subdelegar una parte de la subred. A continuación se muestra un ejemplo.

```
root@berlin:/etc/bind# cat named.conf.local
...
zone "1.20.172.in-addr.arpa" {
        type master;
        file "/etc/bind/db.172.20.1";
        allow-query { any; };
```

```
        allow-transfer { 172.20.1.5; };
};
...
```

En el servidor delegado.

```
root@paraguil:/etc/bind# cat named.conf.local
...
zone "128/25.1.20.172.in-addr.arpa" in{
        type master;
        file "/etc/bind/db.172.20.1.128-255";
        allow-query { any; };
};

...
```

En las configuraciones propias de cada zona.

```
root@berlin:/etc/bind# cat db.172.20.1
;
; BIND reverse data file for  172.in-addr.arpa
;
$TTL    604800
@       IN      SOA     localhost. root.berlin.cities.org. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@        IN       NS     berlin.cities.org.

128/25   IN       NS     paraguil.cities.org.


1        IN      PTR     paraguil.cities.org.
5        IN      PTR     brasilias.cities.org.
100      IN      PTR     berlin.cities.org.


; IPs addresses in the subnet - all need to be defined
; except 128 and 255 since they are the subnet and broadcast
```

```
; broadcast and multicast addresses not hosts/nodes

129             IN  CNAME   129.128/25.1.20.172.IN-ADDR.ARPA. ;qualified
130             IN  CNAME   130.128/25                        ;unqualified name

; The following GENERATE directive would continue the sequence.

$GENERATE 131-254 $ IN CNAME $.128/25

; ### EOF ###
%%

root@paraguil:/etc/bind# cat db.172.20.1.128-255
;
; BIND reverse data file for  172.20.1.128-255.in-addr.arpa
;
$TTL    604800
@       IN      SOA     localhost. root.paraguil.cities.org. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;

@                   IN    NS     paraguil.cities.org.

129                 IN    PTR     quito.cities.org.
130                 IN    PTR     santiago.cities.org.
$GENERATE 131-254  $ IN    PTR     host172-20-1-$.cities.org.
; ### EOF ###
```

Al realizarse la consulta.

```
? host 172.20.1.129  172.20.1.100
Using domain server:
Name: 172.20.1.100
Address: 172.20.1.100#53
Aliases:

129.1.20.172.in-addr.arpa is an alias for 129.128/25.1.20.172.in-addr.arpa.
129.128/25.1.20.172.in-addr.arpa domain name pointer quito.cities.org.
```

4.5. Configuración Básica

Se muestra la configuración inicial del servidor DNS. Se utilizó para los ejemplos la versión BIND `named(8)` de ISC ejecutando sobre GNU/Linux.

```
# uname
Linux

# named -v
BIND 9.3.4-P1.1

# named -v
named 8.4.6-REL-NOESW
```

```
# cat /etc/bind/named.conf

include "/etc/bind/named.conf.options";

// reduce log verbosity on issues outside our control
logging {
        category lame-servers { null; };
        category cname { null; };
};

// prime the server with knowledge of the root servers
zone "." {
        type hint;
        file "/etc/bind/db.root";
};

// be authoritative for the localhost forward and reverse zones, and for
// broadcast zones as per RFC 1912

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
        type master;
```

```
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};

// add local zone definitions here
include "/etc/bind/named.conf.local";

### EOF ###
```

```
# cat /etc/bind/named.conf.local
### Created By Andres 2007-09-29 (C) ###
//
// Add local zone definitions here.


zone "cities.org" {
        type master;
        file "/etc/bind/db.cities.org";
};

zone "172.in-addr.arpa" {
        type master;
        file "/etc/bind/db.172";
};
### EOF ###
```

```
# cat /etc/bind/db.cities.org
;
; BIND reverse data file for cities.org
;
$TTL    604800
@       IN      SOA     berlin.cities.org. root.berlin.cities.org. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expiry
                         604800 )       ; Negative Cache TTL
;
; @                       IN         NS berlin.cities.org.

$ORIGIN .
```

```
cities.org.              IN         NS berlin.cities.org.
#cities.org.             IN         NS brasilia.cities.org.

; ## BRAISLIA NO NECESITA GLUE RECORD ##
trees.cities.org.        IN         NS brasilia.cities.org.
trees.cities.org.        IN         NS berlin.cities.org.
trees.cities.org.        IN         NS oak.trees.cities.org.

; ## GLUE RECORD ##
oak.trees.cities.org.    IN         A 192.168.40.1

cities.org.                IN       MX 1   brasilia.cities.org.
cities.org.                IN       MX 10  berlin.cities.org.


berlin.cities.org.         IN        A    172.20.1.100
brasilia.cities.org.       IN        A    172.20.1.5
paraguil-br0.cities.org.   IN        A    172.20.1.1

ftp.cities.org.            IN    CNAME    berlin.cities.org.
www.cities.org.            IN    CNAME    berlin.cities.org.

berlin.cities.org.         IN    HINFO    Pentium-II/256  GNU/Linux
brasilia.cities.org.       IN    HINFO    MIPS-R10000/512 IRIX6.5

berlin.cities.org.         IN    TXT      "Servidor Web y FTP"

; ### EOF ###
```

```
# cat /etc/bind/db.172
;
; BIND reverse data file for  172.in-addr.arpa
;
$TTL    604800
@       IN      SOA     berlin.cities.org. root.berlin.cities.org. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expiry
                         604800 )       ; Negative Cache TTL
;
@        IN       NS      berlin.cities.org.
1.1.20   IN      PTR     paraguil-br0.cities.org.
5.1.20   IN      PTR     brasilias.cities.org.
100.1.20 IN      PTR     berlin.cities.org.
```

```
5.1.19   IN      PTR     sucre.lat.org.
1.1.19   IN      PTR     paraguil-tap2.lat.org.

; ### EOF ###
```

```
# cat /etc/bind/db.127
;
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expiry
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
1.0.0   IN      PTR     localhost.

; ### EOF ###
```

```
# cat /etc/bind/db.local
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expiry
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
@       IN      A       127.0.0.1

; ### EOF ###
```

```
# /etc/init.d/bind start

# netstat -atunp | grep 53
tcp  0      0 10.20.1.100:53          0.0.0.0:*   LISTEN     5498/named
```

```
tcp  0      0 127.0.0.1:53            0.0.0.0:*   LISTEN      5498/named
udp  0      0 172.20.1.100:53         0.0.0.0:*               5498/named
udp  0      0 127.0.0.1:53            0.0.0.0:*               5498/named
```

4.6. Configuración Master/Slave

Se muestra la configuración del servidor DNS Master y del Slave.

```
root@berlin:/# cat /etc/bind/named.conf.local
//
// Add local zone definitions here.

zone "cities.org" {
        type master;
        file "/etc/bind/db.cities.org";
        allow-query { any; };
        allow-transfer { 172.20.1.5; };
};

zone "172.in-addr.arpa" {
        type master;
        file "/etc/bind/db.172";
        allow-query { any; };
        allow-transfer { 172.20.1.5; };
};

zone "trees.cities.org" {
        type slave;
        masters { 172.20.1.5; };
        file "/var/cache/bind/db.trees.cities.org";
};

### EOF ###
```

```
root@brasilia:/# cat /etc/bind/named.conf.local
//
// Add local zone definitions here.

zone "cities.org" {
        type slave;
        masters { 172.20.1.100; };
        file "/var/cache/bind/db.cities.org";
};
```

```
zone "trees.cities.org" {
        type master;
        file "/etc/bind/db.trees.cities.org";
        allow-query { any; };
        allow-transfer { 172.20.1.100; 192.168.40.1; };

};

### EOF ###
```

4.7. Configuración de Ejemplo de DNSmasq

```
# /etc/init.d/dnsmasq start

# less /etc/dnsmasq.conf

...
# Never forward plain names (with a dot or domain part)
domain-needed
# Never forward addresses in the non-routed address spaces.
bogus-priv

# Change this line if you want dns to get its upstream servers from
# somewhere other that /etc/resolv.conf
#resolv-file=

# Uncomment this to enable the integrated DHCP server, you need
# to supply the range of addresses available for lease and optionally
# a lease time. If you have more than one network, you will need to
# repeat this for each network on which you want to supply DHCP
# service.
#dhcp-range=192.168.1.50,192.168.1.150,12h

# Set the NTP time server address to be the same machine as
# is running dnsmasq - option 42
#dhcp-option=42,192.168.1.1

# default router - 3
#dhcp-option=3,192.168.1.1

# DNS server - 6
#dhcp-option=6,192.168.1.1
...

# dnsmasq -v
Dnsmasq version 2.22  Copyright (C) 2000-2005 Simon Kelley
```

```
...

# netstat -atnp | grep 53
tcp        0      0 0.0.0.0:53    0.0.0.0:*  LISTEN     2006/dnsmasq
```

Referencias

[StevI] TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994. W. Richard Stevens.

[LX] The Linux Home Page: http://www.linux.org/.

[Siever] Linux in a Nutshell, Fourth Edition June, 2003. O'Reilly. Ellen Siever, Stephen Figgins, Aaron Weber.

[BIND] DNS and BIND, Fourth Edition By Paul Albitz, Cricket Liu. O'Reilly. La Third Edtion de 1998 esta disponible online: http://www.unix.com.ua/orelly/networking/dnsbind/index.htm.

[LNAG] Linux Network Administrators Guide. Olaf Kirch & Terry Dawson. 2nd Edition June 2000. http://oreilly.com/catalog/linag2/book/index.html.

[RFC-768] http://www.rfc-editor.org/rfc/rfc768.txt. User Datagram Protocol (Jon Postel 1980 USC-ISI IANA).

[RFC-793] http://www.rfc-editor.org/rfc/rfc793.txt. TCP Transmission Control Protocol (Jon Postel 1981 USC-ISI IANA).

[RFC-882] http://www.rfc-editor.org/rfc/rfc882.txt. DOMAIN NAMES - CONCEPTS and FACILITIES (P. Mockapetris 1983 ISI).

[RFC-883] http://www.rfc-editor.org/rfc/rfc883.txt DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION (P. Mockapetris 1983 ISI).

[RFC-1034] http://www.rfc-editor.org/rfc/rfc1034.txt. DOMAIN NAMES - CONCEPTS AND FACILITIES (P. Mockapetris 1987 ISI).

[RFC-1035] http://www.rfc-editor.org/rfc/rfc1035.txt. DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION (P. Mockapetris 1987 ISI).

[RFC-1912] http://www.rfc-editor.org/rfc/rfc1912.txt. Common DNS Operational and Configuration Errors (D. Barr 1996 The Pennsylvania State University).

[RFC-1834] http://www.faqs.org/rfcs/rfc1834.htm.
Whois and Network Information Lookup Service, Whois++ . (J. Gargano, K. Weiss 1995).

[RFC-2136] http://www.ietf.org/rfc/rfc2136.txt.
Dynamic Updates in the Domain Name System (DNS UPDATE). P. Vixie, Editor (ISC), S. Thomson (Bellcore), Y. Rekhter (Cisco), J. Bound (DEC). 1997.

[RFC-2317] http://www.ietf.org/rfc/rfc2317.txt.
Classless IN-ADDR.ARPA delegation. H. Eidnes (SINTEF RUNIT), G. de Groot (BSD), P. Vixie (ISC). 1998.

[ISC] Internet Software Corporation. Information about the Internet Software Consortium. http://www.isc.org/isc.

[DDNS] A DDNS Server Using BIND and Nsupdate. http://www.oceanwave.com/technical-resources/unix-admin/nsupdate.html.

[Dnsmasq] http://www.thekelleys.org.uk/dnsmasq/doc.html.

[gTLD-ICANN] ICANN gTLDs: http://www.icann.org/en/registries/listing.html.h.

[ccTLD-ICANN] ICANN ccTLDs: http://www.iana.org/domains/root/cctld/.

[NgTLD] ICANN New Generic Top Level Domains: http://newgtlds.icann.org/.

[ICANN] Internet Corporation for Assigned Names and Numbers. http://www.icann.org/en/faq/. http://data.iana.org/TLD/tlds-alpha-by-domain.txt.

[RS] Root Servers. http://www.root-servers.org/.

[Wiki] Wikipedia: http://en.wikipedia.org/wiki/Root_nameserver.

[COM05] Ethereal, Wireshark. Autor original Gerald Combs, 2005. http://www.ethereal.com/. http://www.wireshark.org/.

Índice

Índice de figuras

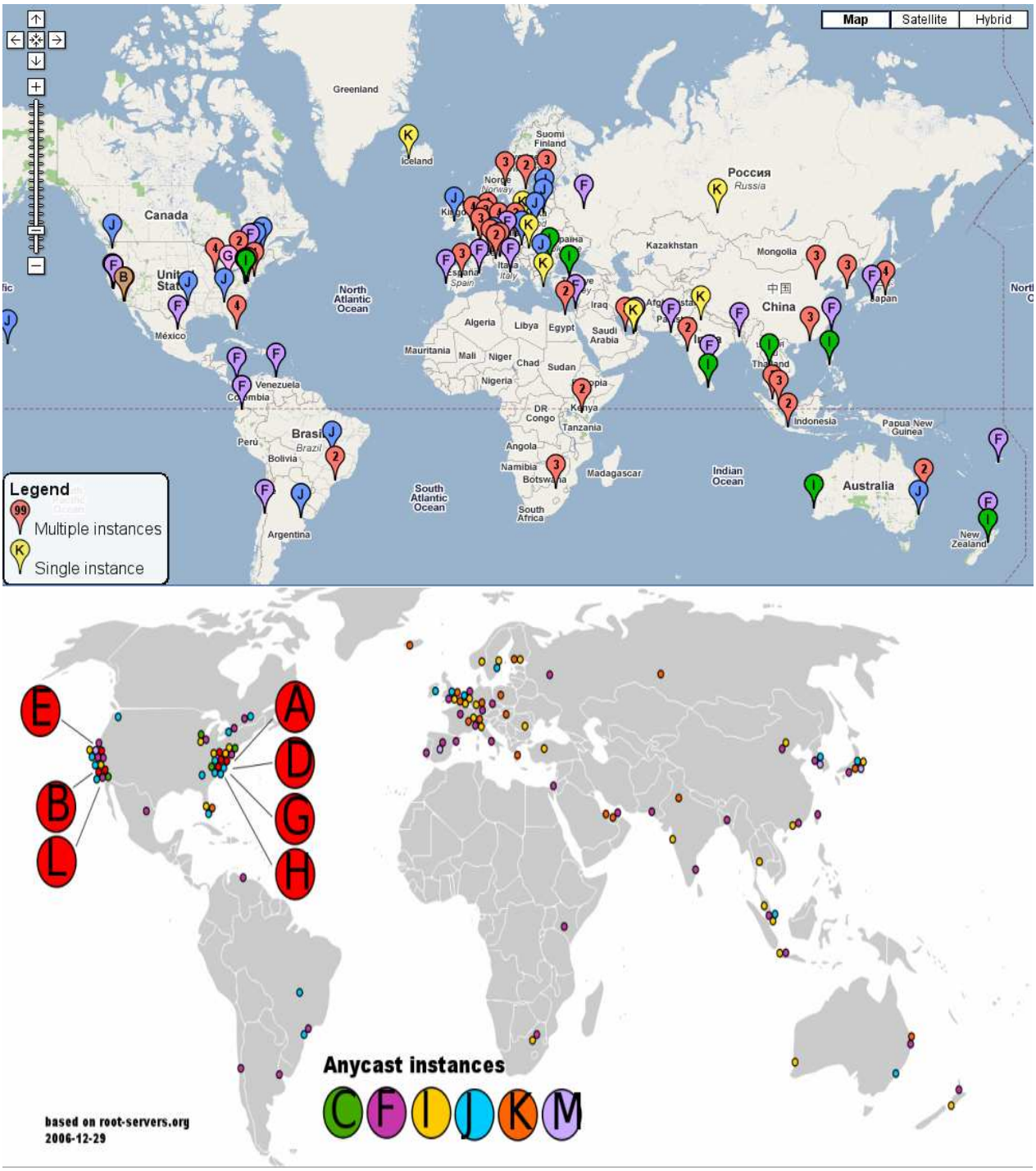

Figura 5: Distribución de los ROOT Servers

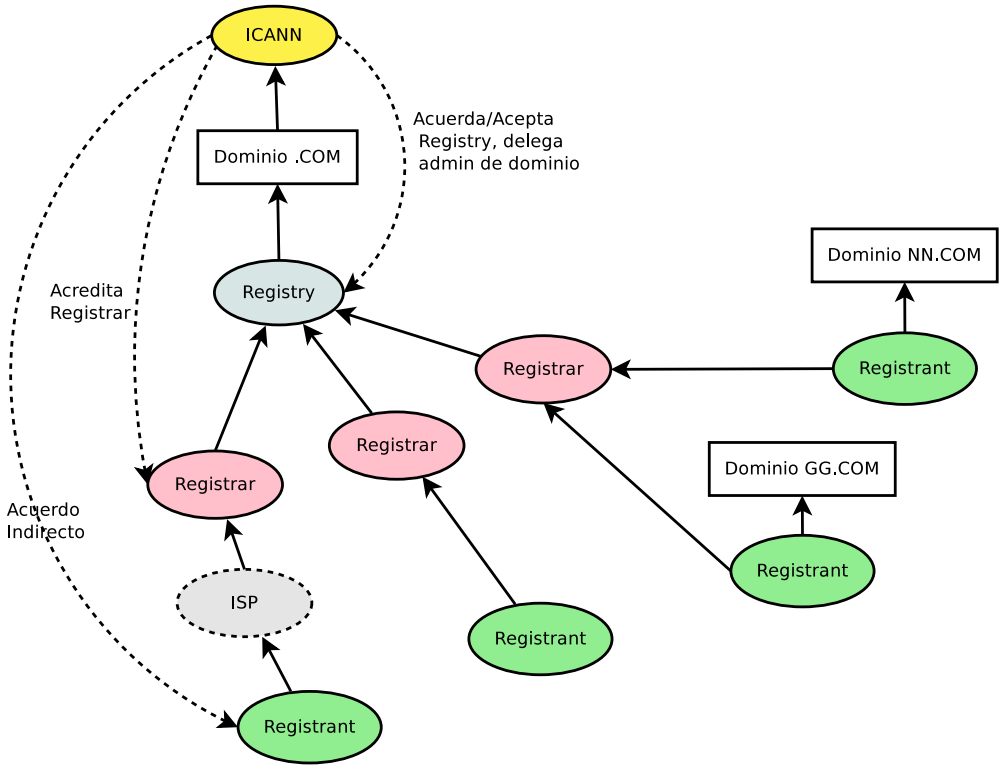

Figura 6: Flujo para el registro de un dominio

Figura 7: Pantalla de Registro de dominio en NIC.ar, anterior y luego de 2013

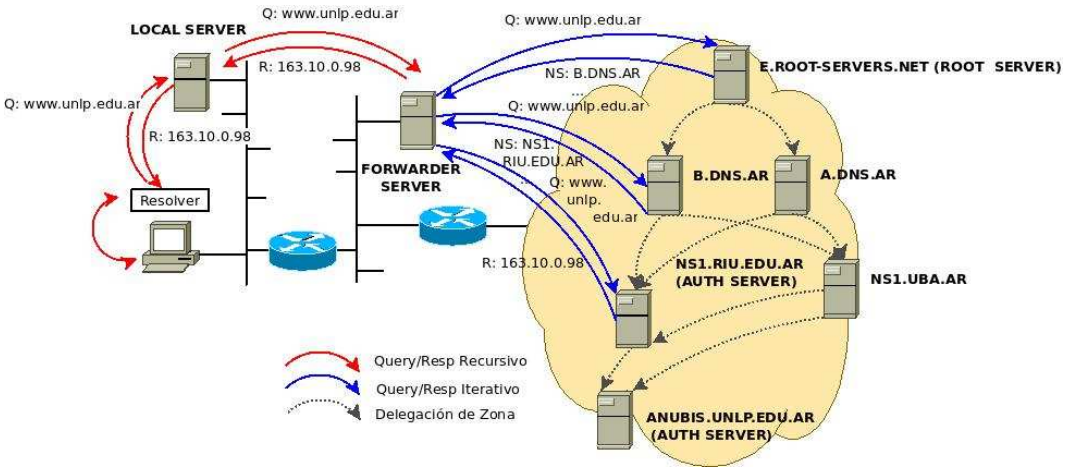

Figura 8: Ejemplos de Consultas Iterativas y Recursivas.

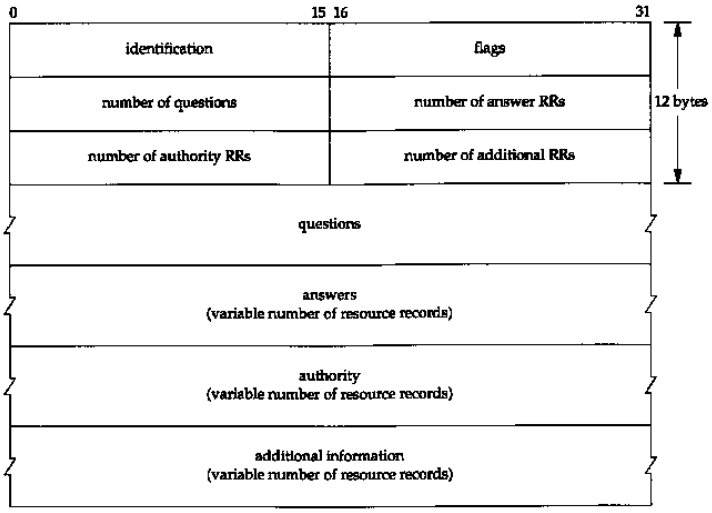

Figura 9: Formato de Mensaje de DNS.

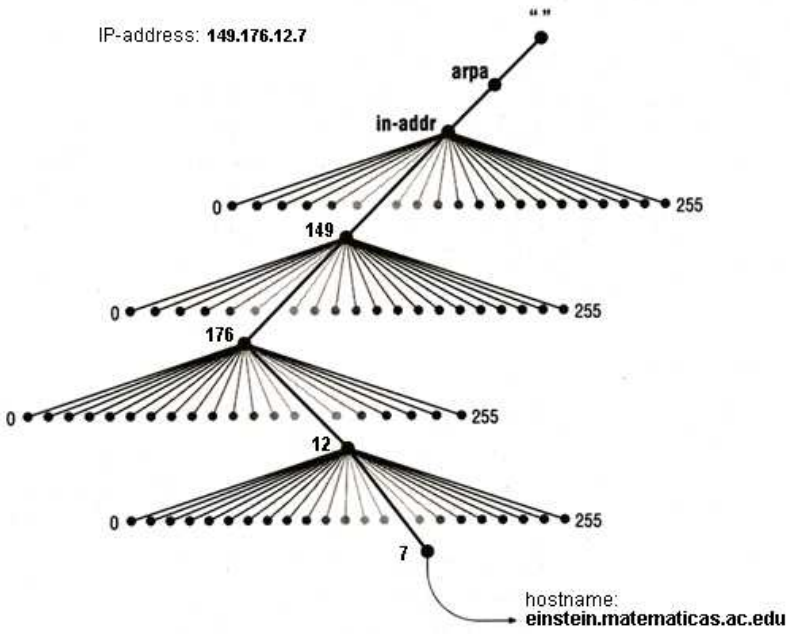

Figura 10: Sub-árbol de rerversos (in-addr.arpa).