Práctica 3.1: Implementación de un cliente DNS simple

Descripción:

En esta práctica el/la estudiante tendrá que implementar en Python un cliente (parcial) del servicio DNS cuyo único objetivo será obtener la dirección IP asociada al nombre DNS de una máquina cualquiera. Para ello, el cliente preparará la pregunta correspondiente siguiendo el protocolo DNS y se la enviará al servidor DNS configurado en la máquina donde se ejecute éste. Luego, el cliente quedará a la espera de una respuesta del servidor y, cuando la reciba, la analizará siguiendo el protocolo y, en el caso de que todo haya ido correctamente, devolverá la dirección IP obtenida.

Material:

Para hacer esta práctica se dispone de los siguientes recursos:

1. Comando *host* para conocer lo que responde el servidor DNS con un nombre DNS dado.
2. Servidor DNS en marcha al que el programa que ha de implementarse en esta práctica realizará las preguntas DNS deseadas.
3. Fichero con el código fuente del cliente parcialmente desarrollado (*dns_cli.py*). Éste es el fichero que habrá que modificar para desarrollar la práctica.

Protocolo de aplicación

Aunque el servicio DNS es sencillo, el protocolo que lo rige y el sistema que lo compone son relativamente complejos. De ahí el amplio conjunto de documentos RFC que lo describen. Uno de los documentos principales es el RFC 1034 *"DOMAIN NAMES - CONCEPTS AND FACILITIES"* (RFC1034-es). Sin embargo, para realizar esta práctica, dado que nos centraremos únicamente en el servicio de traducción de nombres DNS a direcciones IP y dado por conocido el funcionamiento básico general del servicio DNS, basta prácticamente con conocer cómo ha de ser el formato de un mensaje DNS para hacer la pregunta y lo mismo para interpretar la respuesta recibida.

El formato de los mensajes DNS se puede encontrar en el apartado "4. MESSAGES - 4.1. Format" del RFC 1035 "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION". Como verás parte de las respuestas DNS siguen exactamente el formato de los registros de recurso DNS (*Resource Record* (RR)). Algunos de los valores que te hará falta conocer para interpretar la respuesta están definidos en el apartado "3.2. RR definitions - 3.2.1. Format".

Por último resaltar que para interpretar algunas respuestas DNS (dependiendo del servidor DNS consultado), tendrás que ser capaz de interpretar un mecanismo de compresión de mensajes definido en el mismo RFC ("4.1.4. Message compression") que se puede utilizar para reducir el número de bytes que ocupan los nombres DNS en un mensaje.

Aunque el servicio DNS está definido para que pueda trabajar, tanto sobre el protocolo de transporte UDP, como sobre el protocolo TCP, en esta práctica trabajaremos sólo sobre el protocolo UDP.

Servidor

Esta práctica está pensada para ser realizada en el ordenador de un laboratorio docente de la facultad y, por tanto, el servidor que utilizaremos será el que tiene configurado el ordenador del laboratorio, es decir, el propio servidor DNS de la UPV/EHU. Para ver cuál es este servidor, así como el puerto asociado, se puede consultar el fichero "`/etc/resolv.conf`".

Si realizas la práctica en Linux, puede que la dirección IP que encuentres sea una dirección *localhost* (dentro del bloque 127.0.0.0/8). Esto querrá decir que tu ordenador tiene un servidor DNS local instalado. Normalmente estos son unos servidores DNS muy sencillos que poco hacen más allá de trabajos de caché y redireccionar las preguntas DNS al verdadero servidor DNS de la red.

¡ATENCIÓN! El servidor DNS que se propone utilizar es el servidor DNS real de la UPV/EHU y, por tanto, hemos de poner especial atención en lo que hacemos y realizar un uso adecuado de este servicio. Ten cuidado, por ejemplo, de no meter el envío de una pregunta DNS en un bucle infinito y hacer que el servidor se vea inundado de trabajo.

Cliente

El programa que tiene que realizar el/la estudiante será un sencillo cliente DNS que, tomando como parámetro un nombre DNS, construya un mensaje DNS siguiendo el formato definido (RFC 1035) con el objetivo de pedir al servidor DNS la dirección IP asociada a dicho nombre DNS.

Tras enviar la pregunta DNS a través del protocolo UDP, el programa quedará a la espera de la respuesta del servidor DNS. Una vez recibida ésta, el cliente debe verificar que todos los campos que se pueden ver afectados por una respuesta (ID, QR, RCODE, QDCOUNT, ANCOUNT, RDLENGTH) contienen el valor esperado y, en el caso de que todo haya sido correcto, mostrará en pantalla la dirección IP obtenida. Si se ha producido cualquier situación de error, el cliente dará el aviso correspondiente.

Nota: Ten en cuenta que el identificador ID de la cabecera (*header*) del mensaje que contiene la pregunta DNS puede tomar cualquier valor. Te lo puedes inventar tú, lo puedes generar al azar… Por otro lado, el único bit que será necesario que pongas a 1 de todos los flags de la segunda palabra de 16 bits de la cabecera de la pregunta es el bit '*Recursive Desired*' (RD); el resto pueden ser todos 0. Con este bit RD a 1 se le solicita al servidor DNS que sea él el que se encargue de preguntar a otros servidores DNS, si es que no conoce la respuesta (pregunta recursiva).

Dinámica de trabajo

1. Usa el comando *host* para saber qué ha de responder el servidor DNS configurado a cada pregunta concreta que le hagas. La opción *-v* te dará muchos más detalles sobre lo realmente recibido.
2. Construir la pregunta DNS, enviarla y comprobar si se recibe alguna respuesta del servidor.
3. Una vez que recibe alguna respuesta, interpretar ésta verificando el valor de todos los campos que tienen que ver con una respuesta, sea ésta correcta o no. Mostrar dirección IP o mensaje de error.

Fichero *"dns_cli.py"*:

```
#!/usr/bin/env python3

import socket, sys

# Mirar servidor DNS en fichero "/etc/resolv.conf"
DNS_DIR =
DNS_PORT =

if len( sys.argv ) != 2:
      print( "Uso: python3 {} <Nombre DNS de máquina>".format( sys.argv[0] ) )
      exit( 1 )

nombre_dns = sys.argv[1]
```

```
serv_dns = (DNS_DIR, DNS_PORT)

s = socket.socket( socket.AF_INET, socket.SOCK_DGRAM )

"""A COMPLETAR POR EL/LA ESTUDIANTE:
Preparar pregunta DNS
"""
buf = b''
# Header section
# # ID
# # Flags
# # QDCOUNT
# # ANCOUNT
# # NSCOUNT
# # ARCOUNT
# Question section
# # QNAME
# # QTYPE. type = A
# # QCLASS. class = IN
# -- Pregunta DNS completa --
print( "Pregunta DNS a enviar:\r\n", buf )
# Enviar pregunta DNS
s.sendto( buf, serv_dns )
# Recibir respuesta
buf = s.recv( 1024 )
print( "Respuesta recibida:\r\n", buf )
"""A COMPLETAR POR EL/LA ESTUDIANTE:
Intrepretar respuesta
"""
# Header section
# # ID
# # Flags: |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
# #        +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
# # QDCOUNT
# # ANCOUNT
# # NSCOUNT
# # ARCOUNT
# Question section
# # QNAME
# # QTYPE
# # QCLASS
# Answer section: 4.1.3. Resource record format
if not ancount:
      print( 'No se ha recibido ningún registro en la sección de respuestas!' )
else:
      # # NAME (Message compression?)
      # # TYPE
      # # CLASS
      # # TTL: a 32 bit unsigned integer
      # # RDLENGTH: an unsigned 16 bit integer
      # # RDATA
# Authority section: 4.1.3. Resource record format
# Additional section: 4.1.3. Resource record format
s.close()
```

Posibles mejoras

1. Cambia el bit RD de la pregunta DNS a 0 y comprueba el resultado en la respuesta.
2. Cambia el programa para que muestre los valores (a poder ser con su semántica también. Por ejemplo, el valor 1 del campo QTYPE quiere decir tipo A) de todos los campos de la respuesta DNS recibida. Puedes añadir al programa un parámetro estilo *verbose* (*-v*) que muestre toda esta información adicional si lo ha explicitado el usuario o sólo muestre la dirección IP o el mensaje de error, si es que no lo explicita.
3. ¿Qué pasaría si el nombre DNS que pasas como parámetro a tu programa es, en realidad, un *alias*? Pruébalo, por ejemplo, con `smtp.ehu.eus` y analiza el resultado. Cambia el programa para que muestre un resultado acorde a la respuesta recibida.
4. Cambia el programa para que muestre todas las preguntas (*Question*) y todas las respuestas de los 3 tipos (*Answer*, *Authority* y *Additional*) en base a los contadores *QDCOUNT*, *ANCOUNT*, *NSCOUNT* y *ARCOUNT*, respectivamente.
5. Añade un parámetro opcional al programa para que en función de éste el tipo de pregunta realizada pueda ser el indicado: A, AAAA, MX, NS… Ten en cuenta que el formato de la respuesta a mostrar puede cambiar en función de cuál es el tipo de la pregunta. Esta posibilidad también la acepta el comando *host* con la opción *-t* (consulta *man host* ó *host -h*).
6. Analizando en detalle la información que devuelve el comando *host* con la opción *-v* (*verbose*) ¿serías capaz de cambiar el programa para que simulara lo que hace éste?
7. Añade un temporizador (función *select()*) para que el programa, una vez enviada la pregunta DNS, se quede bloqueado un tiempo máximo (en segundos) a la espera de la respuesta del servidor. Este tiempo puede ser indicado también como parámetro opcional del programa. Una vez pasado ese tiempo, si no se ha recibido respuesta, simplemente muestra un mensaje de aviso indicándolo.