Departamento de Ciencias de la Computación
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
CC4303-2 Redes

INFORME ACTIVIDAD DNS

Construir un Resolver

Alumno: Benjamín San Martín
Profesora: Ivana Bachmann
Auxiliar: Vicente Videla

Fecha de entrega: 8 de Septiembre de 2023
Santiago, Chile

Para esta actividad, se solicitó la construcción de un resolver capaz de recibir peticiones de un cliente con un mensaje DNS y que este resolverá la IP del dominio solicitado para enviar la respuesta correspondiente.

1. Flujo de funcionamiento

El flujo de implementación es el siguiente: Se creó la función que parsea el mensaje DNS y lo transforma en un diccionario. Este diccionario contiene 2 tipos de valores, los primeros son 'Qname', 'ANCOUNT', 'NSCOUNT' y 'ARCOUNT'. La primera llave, 'Qname', contiene el dominio de la petición DNS. Las otras tres llaves, 'ANCOUNT', 'NSCOUNT' y 'ARCOUNT', contienen la cantidad de elementos en las secciones 'Answer', 'Authority' y 'Additional', respectivamente. Cada una de estas secciones también tenía su propio diccionario que contenía toda la información de los elementos de ese tipo.

Posteriormente, se creó la función 'resolver'. Esta función toma un mensaje DNS, lo envía al servidor raíz de DNS y, a la respuesta de este servidor raíz, se le aplica el parseo. Dependiendo de los valores obtenidos, se determina la acción a seguir. En primer lugar, se verifica si existen elementos de tipo 'Answer'. Si es así, se verifica dentro del diccionario de la sección 'Answer' si la llave 'answer type' tiene el valor 'A'. De ser así, la respuesta es precisamente esa y se retorna el mensaje. En caso de que esto no ocurra, se procede a verificar si existen elementos de tipo 'Authority'. Si es que hay, se verifica nuevamente si el tipo de estos elementos es 'NS', consultando el diccionario de esta sección. Si la condición se cumple, se procede a verificar si existen elementos de tipo 'Additional'. Si hay elementos de este tipo, existen dos opciones. Primero, se busca algún elemento en esta sección que sea de tipo 'A'. Si se encuentra uno, se toma la IP de este elemento y se envía la consulta original, y la respuesta, al recibirse, se retorna. El segundo caso ocurre cuando no se encuentra un elemento 'Additional' de tipo 'A'. En este caso, se busca un dominio en la sección 'Authority', se crea una 'query' con este dominio para encontrar su IP y se hace recursión para continuar el proceso.

Además del flujo de la función 'resolver', existe una opción de 'debug' que tiene la tarea de imprimir en la consola el dominio que se está resolviendo, el servidor al que se está consultando y su dirección IP.

Finalmente, se crea el socket del servidor, se conecta a la dirección local con el puerto 8000 y se inicia un bucle infinito para que el servidor espere la entrada de mensajes. Una vez que se recibe un mensaje, se llama a la función 'resolver' con ese mensaje y se reenvía lo que la función retorne al cliente.

Adicionalmente, no se logró completar la implementación del caché, pero se procederá a explicar la idea que se tenía en mente al momento de su ejecución. Se creó un diccionario vacío antes del bucle en el que se reciben los mensajes. Este diccionario guardaría el dominio solicitado por el cliente y la dirección IP que lo resuelve. Dentro del bucle, se pregunta si el dominio está dentro del diccionario. Si es así, se procede a tomar el dominio y la IP para crear una consulta DNS, enviarla al servidor que contiene la respuesta y luego reenviar la respuesta al cliente. Si el dominio no está dentro del diccionario, el código procedería de manera normal. Estos valores se actualizarían justo antes de que la función 'resolver' retorne, de esta manera actualizando el cache.

2. Ejecución del código

El código debe ser ejecutado a través de una terminal en un dispositivo que cuente con el sistema operativo Linux. El comando para ejecutarlo es el siguiente:

```
python3 resolver.py
```

Para realizar pruebas, se utilizaron los siguientes comandos, que están disponibles solo en Linux:

```
dig -p8000 @localhost example.com
dig -p8000 @localhost www.uchile.cl
dig -p8000 @localhost eol.uchile.cl
dig -p8000 @localhost www.webofscience.com
dig -p8000 @localhost cc4303.bachmann.cl
```

Cada uno de los comandos anteriores logró enviar las respuestas exitosamente al cliente.

3. Decisiones de diseño

A continuación, se comentarán las razones detrás de ciertas decisiones tomadas al momento de diseñar el código:

- **Uso de diccionarios:** La razón por la que se utilizó esta estructura para almacenar el mensaje DNS es porque cada elemento tiene un nombre específico y, dado que no tienen un orden aparente, se optó por asociar cada valor del mensaje con una llave para evitar tener que buscar los elementos en una lista.
- **Diccionarios dentro de diccionarios:** Dado que el mensaje DNS tiene tres secciones importantes que debíamos almacenar, se optó por guardar en cada sección un diccionario que, a su vez, contenía todos los valores relevantes. De esta manera, acceder a un valor era más sencillo, ya que se utilizaba la notación `diccionario['Answer']['answer type']` por ejemplo.

A continuación se van a responder las preguntas del enunciado:

- **¿Qué tipo de socket debe usar?:** Un socket no orientado a conexión, como el DNS, realizará numerosas conexiones y consultas a otros servidores. Es necesario que este proceso sea lo suficientemente rápido para que el cliente no lo note, incluso si eso implica la pérdida de información en algunos casos.
- **Intente resolver el siguiente dominio con su programa www.webofscience.com ¿Resuelve su programa este dominio? ¿Qué sucede? ¿Por qué? ¿Cómo arreglaría usted este problema?:** La respuesta es distinta a la que ofrece el DNS de google.
- **Ejecute el comando dig -p8000 @localhost www.cc4303.bachmann.cl ¿Qué ocurre? ¿Qué habría esperado que ocurriera?:** El resolver logra devolver una respuesta al cliente, pero no es la misma a la respuesta de google, debido a que en la sección de autorización hay elementos de tipo 'SOA', la implementación de este resolver ignora estos casos por lo que la manera de arreglar esto seria implementando dichos tipos.

- **¿Son siempre los mismos Name Servers? ¿Por qué cree usted que sucede esto?:** No siempre son los mismos Name Servers, yo creo que esto ocurre debido a que la conexión y los datos obtenidos no siempre están en un orden especifico por lo que simplemente se accede al que está primero o el más rápido.