PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC 2333 — Sistemas Operativos y Redes — 2/2016

Tarea 5

Fecha de Entrega: Domingo 04-Diciembre de 2016, 23:59
Composición: grupos de n personas, donde $n \leq 2$

Sobre la nota de Tareas

La nota de Tareas se calculará como:

$$NT = T_1 + T_2 + T_3 + T_4 + T_5 - \text{mín}(T_1, T_2, T_3, T_4, T_5)/4$$

Descripción

En esta tarea implementarán una versión **simplificada** del protocolo de aplicación DNS, **usando sockets en C ó C++**. Deberán implementar el programa servidor, y demostrar su funcionamiento con un programa cliente que haga consultas.

En este protocolo un cliente hace consultas (*query*) por un nombre de *host*. El servidor busca en su base de datos las entradas que coinciden con la consulta y las envía como respuesta *response* al cliente.

El cliente también puede mandar una *reverse query*, en la cual se consulta por una dirección IP, el servidor envía una *response* con la entrada correspondiente.

Formato del mensaje

Tanto la *query* como la *response* siguen el mismo formato:

- 16 bit que indican el *identificador* del mensaje. Este identificador sirve para relacionar *queries* con *responses*. Cada *response* lleva el mismo *identificador* que su correspondiente *query*. Puede ser determinada secuencial o aleatoriamente.
- *Flags*. 16 bit. De los cuales:
  - El primer bit es 0 cuando se trata de una *query*, y es 1 cuando se trata de una *response*.
  - Los siguientes 4 bit son $0b0000$ cuando se trata de un *query* normal (se consulta por un nombre), y $0b0100$ cuando se trata que una *query reversa* (se consulta por una IP).
  - El siguiente bit es 0 en todas las *queries* y es 1 en las *response* cuando el server es *authoritative* (posee la respuesta) para la correspondiente *query*.
  - Los siguientes 6 bit son 0 tanto para *query* como para *response*.
  - Los siguientes 4 bit contienen 0 si no hubo errores en la *response*, o algo distinto de 0 para indicar que la *query* no pudo ser resuelta.
- *NumQueries* 16 bit que indican la cantidad de consultas que vienen en un mensaje de tipo *query*. En este caso siempre será $0x0001$.
- *NumResponses* 16 bit que indican la cantidad de respuestas que vienen en un mensaje de tipo *response*. (0 en un mensaje de tipo *query*).
- 16 bit en 0

- 16 bit en 0

  A continuación se envía la consulta que incluye:

  - Nombre a resolver, o bien dirección IP en caso que se trate de una consulta reversa.

Servidor

El servidor se inicia con una base de datos que es un archivo `.dns` que contiene tuplas `(NAME, VALUE, TYPE, TTL)`

- `NAME`: Nombre consultado
- `VALUE`: IP u otro nombre
- `TYPE`, el tipo de entrada, que puede ser:
  - `A`. Signifca que `Value` es la IP para el `Name` consultado.
    Ejemplo: `(hercules.ing.puc.cl, 146.155.13.45, A, 36800)`
  - `CNAME`. Significa que `Value` es un *alias* para el valor consultado.
    Ejemplo: `(ing.puc.cl, www.ing.uc.cl, CNAME, 36800)`
- `TTL` es el tiempo de vida de la entrada, que puede ser ignorado en esta versión del protocolo.

Ejemplo de archivo `server.dns`

```
hercules.ing.puc.cl        146.155.13.45        A        46800
iic2333.ing.puc.cl        hercules.ing.puc.cl        CNAME        46800
hercules        hercules.ing.puc.cl        CNAME        46800
ing.puc.cl  www.ing.uc.cl   CNAME   46800
www.ing.puc.cl  www.ing.uc.cl   CNAME   46800
www.ing.uc.cl   146.155.4.12    A   46800
```

El servidor debe escuchar llamadas en el puerto 1029, vía UDP.

Al recibir una consulta, el servidor responde con una *response* de acuerdo al formato indicado, donde se adjunta la consulta, y además la respuesta correspondiente. En este caso los campos *NumQueries* y *NumResponses* deben ser $0x0001$.

Cliente

Debe construir un programa cliente que espere entradas del usuario por *stdin*. Por cada entrada del usuario, el cliente debe hacer una *query* DNS al servidor. Si la entrada tiene formata de dirección IPv4, entonces debe enviar una *reverse query*.

Luego de recibir la respuesta del servidor, el cliente debe desplegar el resultado en pantalla. Si la respuesta es del tipo `A`, entonces debe desplegar el mensaje:

`NAME has address VALUE`

Si la respuesta es del tipo `CNAME`, debe desplegar el mensaje:

`NAME is an alias for VALUE`

y luego debe repetir la búsqueda usando `VALUE` en el campo de la consulta.

Finalmente, si la respuesta no es válida, o el servidor no pudo encontrar la entrada, debe desplegar el mensaje:

`Host NAME not found`

El programa debe permance corriendo y esperar más consultas, o hasta que se le envíe un SIGINT (Ctrl+C).

La dirección del servidor debe ser ingresada por línea de comando al momento de iniciar el programa.

Entrega

Debe entregar dos carpetas en un archivo .zip.

- `server` con el código del servidor y las instrucciones para compilarlo (por ejemplo, un Makefile)
- `client` con el código del cliente y las instrucciones para compilarlo (por ejemplo, un Makefile)
- Un archivo `README` que contenga detalles a tener en cuenta para la ejecución de la solución. Todo lo que facilite la labor del ayudante será bien visto.

Evaluación

Se evaluará, con una escala de 1 a 7, los siguientes elementos. La nota final de la tarea será el promedio ponderado de ellas.

- 10 % Formato: Entrega correcta de los archivos incluyendo instrucciones de compilación.
- 30 % Construcción correcta de las *queries* y las *response*
- 10 % Construcción correcta de las *reverse queries*.
- 20 % Funcionamiento correcto del *server*. Capacidad de encontrar las respuestas.
- 15 % Funcionamiento correcto del *cliente*
- 15 % Interactividad con otro servidor. Si su cliente funciona correctamente, éste debe ser capaz de conversar usando el protocolo descrito, con un servidor correcto de otro usuario.
- 20 % (Bonus) Entrega una versión adicional del cliente (otra carpeta) que sea capaz de conversar con un servidor DNS (completo) real. Ejemplo: 8.8.8.8, 146.155.4.4. Para que esto funcione debe investigar características adicionales del protocolo DNS.