FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

ASIGNATURA: COMUNICACIONES Y REDES

PROFESOR: RAFAEL VICENTE PAÉZ MÉNDEZ

SERVIDOR DNS

AUTORES:

CARLOS ORLNADO BARÓN LEÓN

SANTIAGO CHAUSTRE PERDOMO

ANDRÉS FELIPE COCUNUBO QUINTERO

Hell Soft S . A

Descripción del proyecto

El Sistema de Nombres de Dominio (DNS- Domain Name System) es una base de datos que contiene los nombres de los websites vinculados a una dirección IP. Es un software simple basado en un servidor que permite de manera sencilla conectar direcciones web y direcciones IP registradas. DNS fue creado con la intención de que en vez de que los clientes digitaran la IP de una página web, se usaran nombres mucho más fáciles de reconocibles.

Como todo el tiempo existen millones de clientes que quieren acceder a páginas web existe un sistema de caché, que consiste en que una vez el dispositivo haya accedido a la página web guarde dentro de su memoria la dirección IP durante un periodo de tiempo.

Por otro lado, existe algo conocido como tipos de registros que son configuraciones implementadas internamente dentro del DNS. Las zonas del DNS que posean un determinado tipo de registro, permiten saber cómo resolver las peticiones que le haga el cliente, su formato es el siguiente:

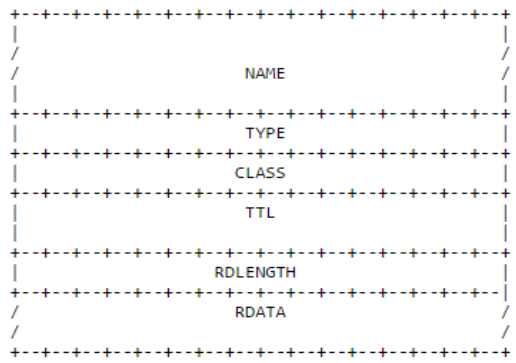


Imagen 1. Formato de Tipo de Registro

Se debe tener en cuenta un componente importante en las comunicaciones y es el mensaje, este contiene información muy importante, como por ejemplo si es una petición o una respuesta, una petición normal o una inversa, etc.

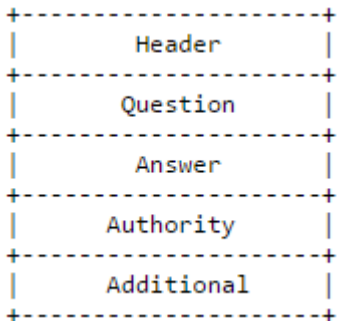


Imagen 2. Formato Mensaje

Está compuesto por cinco secciones, la cabecera, la pregunta al servidor de nombres, la respuesta del registro, el registro apuntando a un servidor mayor e información adicional que contiene registros relacionados a la petición, pero no son de mucha importancia.

Aplicación del DNS en el proyecto

Teniendo en cuenta la breve explicación de lo que era un DNS y sus principales elementos, se explicará el funcionamiento del mismo a través de un programa realizado en C/C++. Para ello se probará en un escenario de máquinas configuradas con la dirección DNS de uno de los computadores de la red , que hará las veces de servidor, donde se encontrarán las direcciones IP y los nombres de host , que se ubican en un master file.

- **Librerías utilizadas**

Librería	Descripción
<netinet/in.h>	Está librería contiene las definiciones para la familia de protocolos de internet, las cuales son necesarias para la transmisión de información entre máquinas.
<sys/socket.h>	Contiene las definiciones de los sockets, los cuales sirven para la entrega de paquetes de datos provenientes de la tarjeta de red, queda definido por de direcciones IP y un puerto.
<errno.h>	Define las constantes simbólicas que son retornadas por una variable externa.
<arpa/inet.h>	Contiene las definiciones de las operaciones de internet, por ejemplo, obtener una IP y convertirla a string o viceversa.
<netdb.h>	Define operaciones de la base de datos de la red.
<sys/types.h>	Posee una colección de tipos definidos de símbolos y estructuras utilizados para manipular algunas variables de la red como las IP.

- **Diagrama de clases**

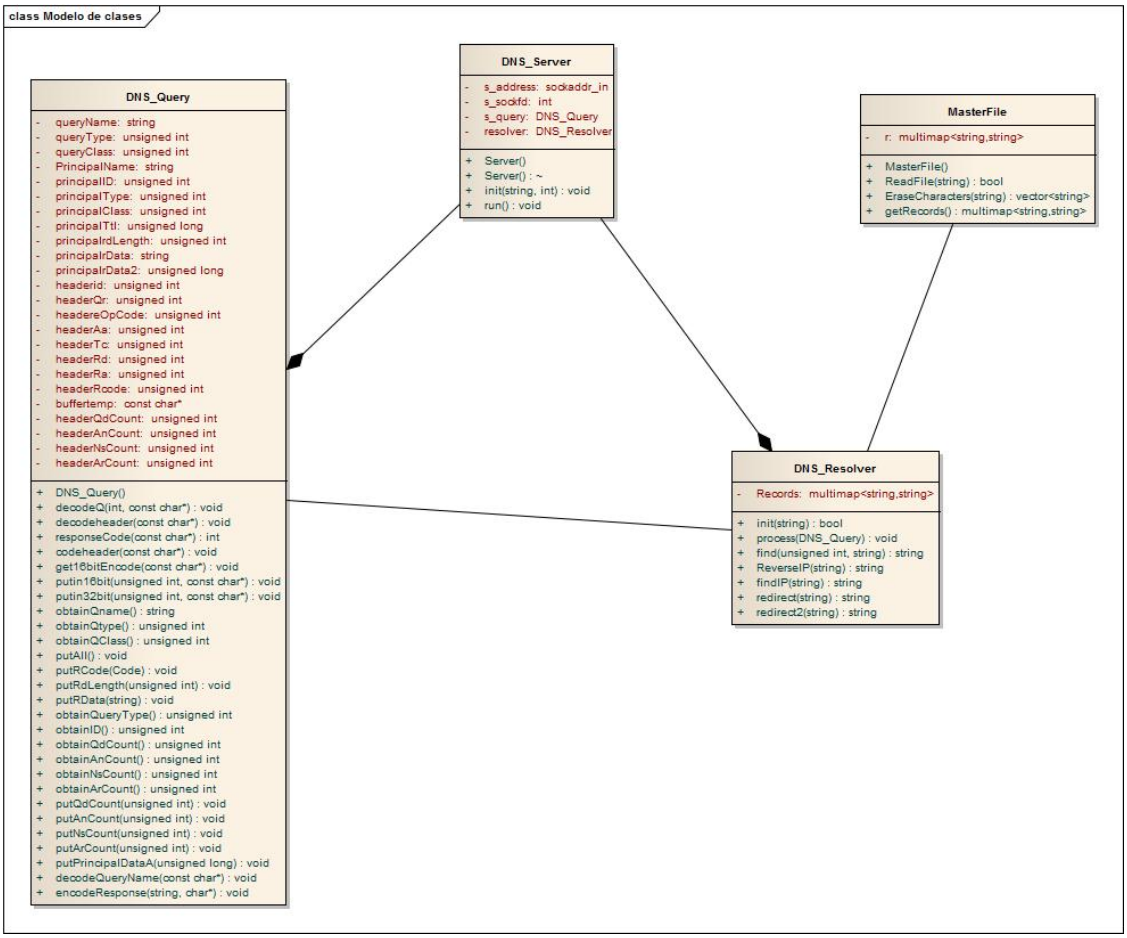
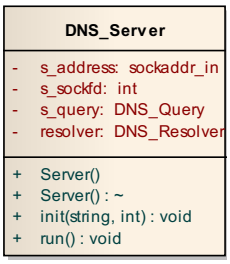
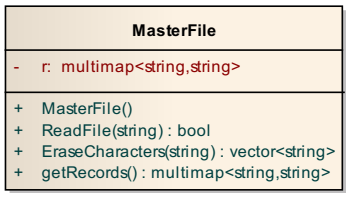
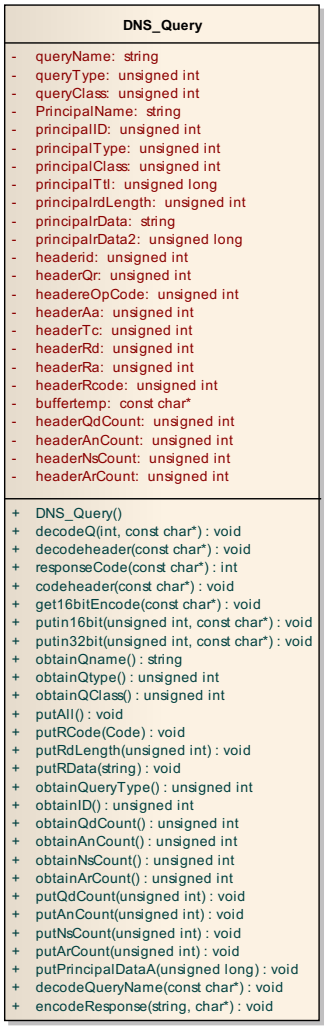
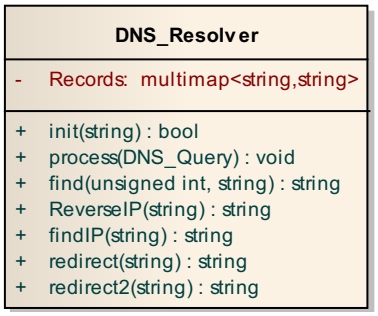


Imagen 3. Diagrama de clases

- **Descripción de principales funciones**

Clase	Función	Descripción
	Init(string,int)	Con el número del puerto y el nombre de masterfile donde se tiene guardado las relación de direcciones IP y hostname, se crea un socket a partir del cual se crea una conexión.
	Run()	Una vez se tenga la conexión el servidor se queda en espera de conexiones.
	ReadFile(string)	Lee el server Name que posee las direcciones IP con sus respectivos hostname y los guarda.
	EraseCharacters(string)	Obtiene cada hostname y dirección Ip.
	getRecords()	Obtiene la información de los hostname y sus respectivas direcciones.
	DecodeQ(int, const char*)	Descifra la dirección IP y asigna los respectivos datos a la query.
	decodeHeader(const char*)	Descifra la cabecera y guarda la información correspondiente en la query.
	responseCode(const char*)	Codifica el mensaje que se le va a enviar al cliente, y retorna el tamaño.
	Codeheader(const char*)	Función adicional que codifica los datos para la cabecera.
	Get16bitEncode(const char*)	Separa 16 bits de la cabecera para manipulación de la información.
	Putin16bit(unsigned int, const char*)	Encapsula 16 bits dentro del mensaje de retorno.
	Putin32bit(unsigned long, const char*)	Encapsula 32 bits dentro del mensaje de envío.
	encodeResponse(char* buffer,string response)	Codifica la respuesta que se le dará al cliente.

<table>
<tr>
<td rowspan="7"> 
</td>
<td>Init(string)</td>
<td>Guarda las direcciones IP y sus respectivos hostname.</td>
</tr>
<tr>
<td>Process(DNS_Query)</td>
<td>A partir de una petición que le haga el cliente se determina si es normal y se muestra la ip relacionada al hostname o inversa donde se muestra el hostname relacionado a la dirección IP.</td>
</tr>
<tr>
<td>Find(unsigned int,string)</td>
<td>A partir de un dominio se busca dentro de los records del resolver la dirección IP correspondiente.</td>
</tr>
<tr>
<td>ReverseIP(string)</td>
<td>Cuando el cliente envía una query inversa, se debe realizar una conversión ya que está se encuentra en sentido contrario, junto con ‘.in-addr.arpa’, lo que determina un query inversa.</td>
</tr>
<tr>
<td>findIP(string)</td>
<td>A partir de la dirección IP se busca en los records y se retorna el hostname correspondiente.</td>
</tr>
<tr>
<td>Redirect(string)</td>
<td>En consulta normal, cuando en el master file no se encuentra la dirección IP, retorna la dirección correspondiente al host desde otro server exterior.</td>
</tr>
<tr>
<td>Redirect2(string)</td>
<td>En consulta inversa, cuando no se encuentra la dirección IP, se retorna el nombre del host desde un servidor exterior.</td>
</tr>
</table>

- **Diagrama de secuencia**

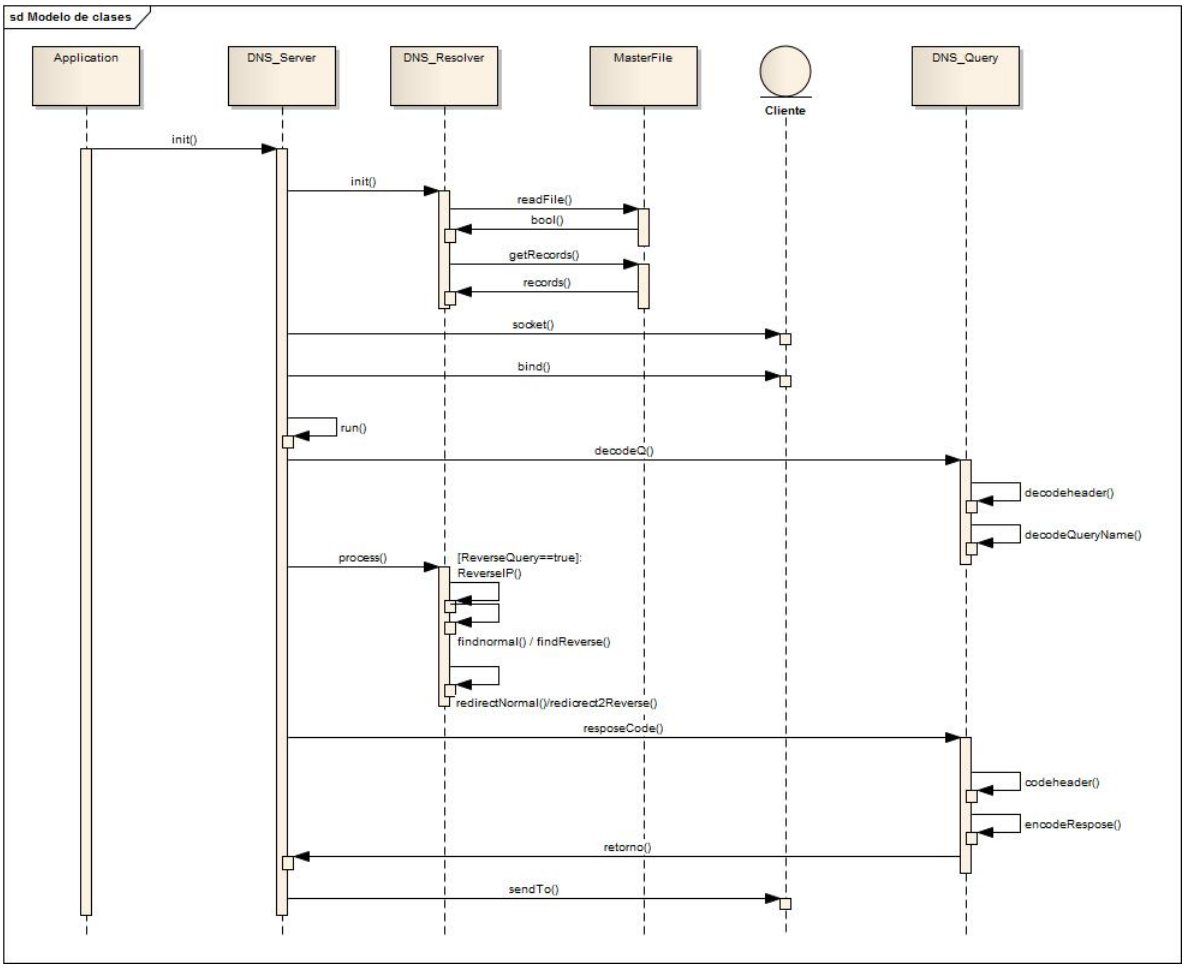


Imagen 4. Diagrama de secuencia

- **Explicación del funcionamiento del programa**

En el programa principal se inicializa el servidor, en este proceso el resolver obtiene la información del master file , el cual tiene como formato dos columnas , en la cual la primera se representa la direccion ip y en la segunda el nombre del host en la cual se guardan los registros en el sistema, después el servidor crea la conexión con el/los cliente(s), creando un socket y estableciendo un enlace a través del puerto 53 que se utiliza mediante UDP, el cual según la documentación del RFC 1035 “DOMAIN NAMES – IMPLEMENTATION AND SPECIFICATION”, el mensaje que se lleva es de estrictamente 512 bytes (sin incluir la IP ni la cabecera del UDP), como este protocolo es no orientado a conexión las peticiones que realice el usuario se pueden perder, por lo que se necesita una estrategia de retransmisión para no perder la información. Esta consiste en que la red reorganice la información o por procesamiento en los servidores de nombres, permitiendo así que el resolver no se preocupe por el orden de las peticiones.

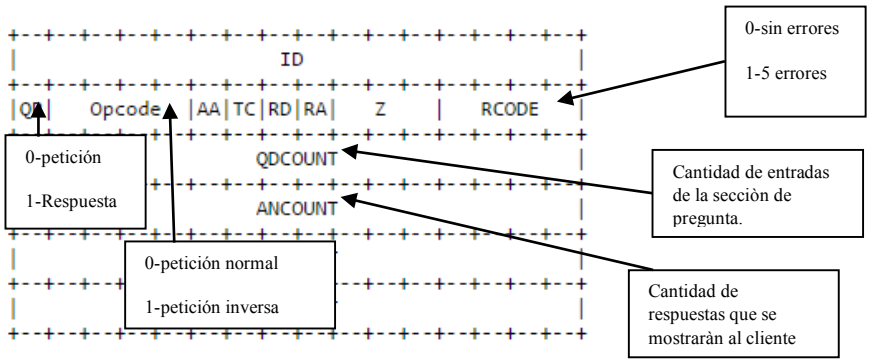


Imagen 5. Formato de cabecera

Con la conexión establecida el servidor, queda en estado de espera hasta que algún cliente le envíe una petición; Cada vez que recibe un mensaje del socket obtiene la cantidad de bytes y, además crea una petición nueva la cual va a llenar con la información suministrada por el mensaje.

Decodifica la cabecera del mensaje, partiéndola en cada uno de sus elementos como se muestra en la imagen 5, que está divida en 6 franjas y estas tienen un tamaño de 16 bits, se utiliza una función que obtiene los 16bits de cada campo, y a medida que se hace esto se guarda en los atributos de una petición.

Se decodifica los datos específicos de la query, a partir del formato de la pregunta, el nombre,

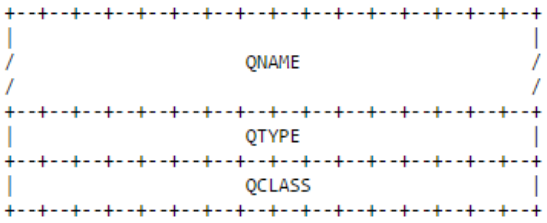


Imagen 5. Formato de Pregunta

el tipo y la clase.

El formato de la imagen 6, contiene información del nombre del dominio, el tipo de petición (normal o inversa), tipo de información, tiempo de vida de la petición la cual es la única que tiene un tamaño de 32 bits, el tamaño del Rdata, y el Rdata que el contiene la información de la máquina conectada (especificaciones y sistema operativo).

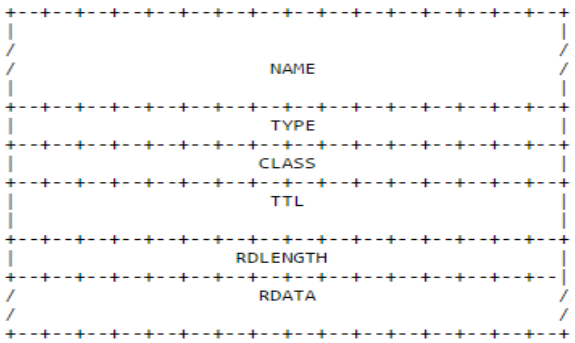


Imagen 6. Formato de Respuesta-
autoridad e info adicional

Una vez realizada la decodificación se procede con el procesamiento de la petición, este consiste en obtener el nombre de la query y su tipo, los cuales se encuentran guardados en el sistema, con estos datos se comprueba de si es una consulta normal o una inversa, en consulta normal se debe realizar una conversión para que al momento de armar el paquete tenga el formato establecido .

De no haber problemas con los datos se arma el paquete de respuesta, si es normal se busca la dirección IP de acuerdo al nombre del host dado por el cliente, ya que al enviar la ip al cliente es necesaria enviarla en formato de 4 bytes , cada uno con un octeto de la ip si es inversa se debe tener en cuenta que la ip se encuentra al revés junto con el prefijo ".in-addr.arpa", por lo cual se debe hacer un conversión, para obtener solo la dirección IP y con este buscar el nombre de host el cual se codifica en bloques de bytes para la respuesta.

Cuando no se encuentre el nombre del host o la dirección IP que solicita el usuario dentro del master file del servidor, el programa entra a la función redirect(query normal) o redirect2(query inversa), en la cual captura el error y busca en un servidor DNS más externo la dirección IP o nombre de host que se necesita.

Finalmente se codifica el paquete armado, codificando inicialmente la cabecera y después la respuesta, con sus correspondientes componentes para retornarle al cliente el nombre del host o la dirección IP.

Notas importantes:

- El cliente debe cambiar su dirección de DNS a la del servidor.
- El servidor debe ejecutarse en sistemas operativos basados en UNIX, tales como GNU/Linux, Mac OS X.
- Solo soporta IPv4.

REFERENCIAS

- https://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.v2r2.bpxbd00/toc.htm
- https://www.ietf.org/rfc/rfc1035.txt
- https://webhostinggeeks.com/guides/dns/