## Ejemplo Protocolo DNS

El presente ejemplo se basa en la manipulación y procesamiento de registros DNS (Domain Name System). El DNS es un componente crucial de la infraestructura de internet, ya que se encarga de resolver nombres de dominio en direcciones IP. Cuando un usuario intenta acceder a un sitio web, por ejemplo, "www.ejemplo.com", su dispositivo realiza una solicitud DNS para traducir este nombre de dominio en una dirección IP que las máquinas puedan entender y utilizar para establecer la conexión.

El proceso que realiza el código tiene como objetivo analizar y presentar información relacionada con los diferentes tipos de registros DNS. Los registros DNS son básicamente entradas en la base de datos DNS, y cada tipo de registro tiene una función específica. Por ejemplo, el tipo "A" es uno de los más comunes y asocia un nombre de dominio con una dirección IPv4. Otros tipos de registros incluyen "CNAME", que se utiliza para definir un alias para un dominio, y "MX", que se asocia con los servidores de correo electrónico de un dominio.

El código incluye una serie de funciones diseñadas para manejar estos diferentes tipos de registros. Una de las funciones principales, imprimir_tipo_registro, se encarga de identificar el tipo de registro DNS y mostrar su tipo en un formato comprensible para el usuario. Por ejemplo, si el registro es de tipo 1, la función imprimirá que se trata de un registro Host. Si es de tipo 2, indicará que es un registro (A) de servidor de nombres, y así sucesivamente. Este tipo de estructura es esencial en aplicaciones que requieren la interpretación de respuestas DNS, como los clientes DNS o las herramientas de diagnóstico de red.

Se tienen procedimientos relevantes para poder tener el flujo de procesamiento de las direcciones y segmentarlas como se muestra a continuación:

- **configurar_servidor() y configurar_cliente():** Configuran las direcciones IP y puertos para el servidor DNS y el cliente.
- **manejo_errores():** Maneja posibles errores en las operaciones de red.
- **configurar_encabezado():** Prepara el encabezado de la consulta DNS.
- **configurar_buffer():** Prepara el paquete completo para la solicitud DNS incluyendo el encabezado y la dirección web.
- **pedir_direccion():** Solicita al usuario ingresar la dirección web que quiere consultar.
- **respuesta_dns():** Procesa la respuesta recibida del servidor DNS y extrae los datos.

El flujo principal del código consiste en 5 pasos como se muestra a continuación:

1. Se abre un socket y se configuran las estructuras para cliente y servidor.
2. Se solicita la dirección web a consultar y se configura el paquete DNS.
3. Se envía la solicitud DNS y se espera una respuesta.
4. La respuesta se procesa y se imprime en la consola.
5. Se puede establecer la lógica del presente código como se muestra a continuación:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <ctype.h>

const unsigned char INDICADORES[2] = {0x01, 0x00};
const unsigned char ID_TRANS[2] = {0x02, 0x0a};   /* ID de la transaccion */
const unsigned char PETICIONES[2] = {0x00, 0x01}; /* Contador de Peticiones (1)
*/
const unsigned char CERO[1] = {0x00};

const unsigned char SOLICITUD[1] = {0x00};                          /* Solcitud
al DNS Host */
const unsigned char CODIGO_SOLICITUD[4] = {0x00, 0x00, 0x00, 0x00}; /* Codigo de
operacion de solicitud al DNS */
const unsigned char RECURSION[1] = {0x01};

const unsigned char REG_HOST[2] = {0x00, 0x01};  /* Registro Host */
const unsigned char CLASE_INT[2] = {0x00, 0x01}; /* Clase Internet */

struct sockaddr_in configurar_servidor(struct sockaddr_in);
struct sockaddr_in configurar_cliente(struct sockaddr_in);
void manejo_errores(int, int);
void configurar_encabezado(unsigned char[12]);
void pedir_direccion(unsigned char[50]);
int configurar_buffer(unsigned char[12], unsigned char[50], unsigned char[512]);
void respuesta_dns(unsigned char[512]);
void imprimir_tipo_registro(int);
void imprimir_clase();
void imprimir_tiempo(unsigned char[512], int);
int es_apuntador(unsigned char buffer[512], int p);
int imprimir_n_caracteres(unsigned char buffer[512], int num, int apuntador);
int ciclo_impresion(unsigned char buffer[512], int pointer);
int guardar_apuntador(unsigned char buffer[512], int point, int apuntador);
int imprimir_ip(unsigned char buffer[512], int pointer);
int recorrer_trama(unsigned char buffer[512], int apuntador);
int imprimir_ipv6(unsigned char buffer[512], int apuntador);


void main()
{
    int sockfd;                  /* Socket */
    int longFinal;               /* Almacenaremos la longitud final de la
solicitud */
    int tam;                     /* Variable que usaremos para guardar posibles
errores -1 */
    unsigned char buffer[512];   /* Buffer para almacenar paquetes recibidos */
    unsigned char encabezado[48]; /* Encabezado DNS */
    unsigned char direccion[50];  /* Arreglo donde almacenaremos la direccion Web
*/
    unsigned char direc[50]="www.youtube.com";
```

```
unsigned char di[50]="youtube.com";
    struct sockaddr_in cliente;   /* Estructura que usaremos para configurar el
cliente */
    struct sockaddr_in servidor;  /* Estructura que usaremos para configurar el
servidor */

    memset(buffer, 0, 512);   /* Limpiamos la cadena para evitar posibles errores
*/
    memset(direccion, 0, 50); /* Limpiamos la cadena para evitar posibles errores
*/

    /* Abrimos el socket */
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    manejo_errores(0, sockfd); /* Verificamos que no haya errores al abrir el
socket */

    /* Configuramos el cliente, familia, puerto, etc. */
    cliente = configurar_cliente(cliente);

    /* Asignamos direccion al socket */
    tam = bind(sockfd, (struct sockaddr *)&cliente, sizeof(cliente));
    manejo_errores(1, tam); /* Verificamos que no haya errores en el bind */

    /* Configuramos el servidor, familia, puerto, IP, etc. */
    servidor = configurar_servidor(servidor);

    /* Le pedimos al usuario que nos de la direccion Web */
    pedir_direccion(direccion);

if(strcmp(direccion,direc)==0){
printf("BIENVENIDO A TU DNS");

direc[strlen(direc) - 1] = '\0';

 /* Configuramos el encabezado con id de transaccion, identificadores,
peticiones, etc */
    configurar_encabezado(encabezado);

    /* Configuramos el buffer a enviar, colocamos el encabezado, entradas de
solicitud, etc.
    recibimos la longitd final de la solicitud */
    longFinal = configurar_buffer(encabezado,di, buffer);

    /* Enviamos la solicitud al DNS */
    tam = sendto(sockfd, buffer, longFinal, 0, (struct sockaddr *)&servidor,
sizeof(servidor));
    manejo_errores(3, tam); /* Verificamos que no hubo errores al enviar */
    perror("\nExito al enviar la solicitud");

    int len = sizeof(servidor); /* Es un apuntador a la estructura servidor */

    /* Recibimos la respuesta del DNS */
    tam = recvfrom(sockfd, buffer, 512, 0, (struct sockaddr *)&servidor, &len);
    manejo_errores(2, tam); /* Verificamos que no hubo errores al recibir */
    perror("\nExito al recibir la respuesta");

    /* Identificamos los datos recibidos en la respuesta del DNS */
    respuesta_dns(buffer);
```

```
    /* Cerramos el socket */
    close(sockfd);

    /* Metemos un salto de lines por pura estetica */
    printf("\n");


}



    /* Configuramos el encabezado con id de transaccion, identificadores,
peticiones, etc */
    configurar_encabezado(encabezado);

    /* Configuramos el buffer a enviar, colocamos el encabezado, entradas de
solicitud, etc.
    recibimos la longitd final de la solicitud */
    longFinal = configurar_buffer(encabezado, direccion, buffer);

    /* Enviamos la solicitud al DNS */
    tam = sendto(sockfd, buffer, longFinal, 0, (struct sockaddr *)&servidor,
sizeof(servidor));
    manejo_errores(3, tam); /* Verificamos que no hubo errores al enviar */
    perror("\nExito al enviar la solicitud");

    int len = sizeof(servidor); /* Es un apuntador a la estructura servidor */

    /* Recibimos la respuesta del DNS */
    tam = recvfrom(sockfd, buffer, 512, 0, (struct sockaddr *)&servidor, &len);
    manejo_errores(2, tam); /* Verificamos que no hubo errores al recibir */
    perror("\nExito al recibir la respuesta");

    /* Identificamos los datos recibidos en la respuesta del DNS */
    respuesta_dns(buffer);

    /* Cerramos el socket */
    close(sockfd);

    /* Metemos un salto de lines por pura estetica */
    printf("\n");
}

/* Configuramos el servidor */
struct sockaddr_in configurar_servidor(struct sockaddr_in servidor)
{
    servidor.sin_family = AF_INET;                          /* Familia: AF_INET */
    servidor.sin_port = htons(53);                          /* Puerto del DNS */
    servidor.sin_addr.s_addr = inet_addr("148.204.103.2"); /* IP del Poli */
    //servidor.sin_addr.s_addr = inet_addr("8.8.4.4"); //IP DE GOOGLE
    return servidor;
}

/* Configuramos el cliente */
struct sockaddr_in configurar_cliente(struct sockaddr_in cliente)
{
    cliente.sin_family = AF_INET; /* Familia: AF_INET */
```

```
    cliente.sin_port = htons(0);  /* Colocamos puerto efimero asi */
    cliente.sin_addr.s_addr = INADDR_ANY;
    return cliente;
}

/* Manejamos los errores -1 en tam */
void manejo_errores(int codigo, int num)
{
    switch (codigo)
    {
    case 0:
        if (num == -1)
        {
            perror("\nError al abrir el socket");
            exit(0);
        }
        perror("\nExito al abrir el socket");
        break;
    case 1:
        if (num == -1)
        {
            perror("\nError en el bind");
            exit(0);
        }
        perror("\nExito en el bind");
        break;
    case 2:
        if (num == -1)
        {
            perror("\nError al recibir");
            exit(0);
        }
        break;
    case 3:
        if (num == -1)
        {
            perror("\nError al enviar");
            exit(0);
        }
        break;
    }
}

/* Configuramos el encabezado */
void configurar_encabezado(unsigned char encabezado[12])
{
    memset(encabezado, 0, 12);              /* Colocamos todo en ceros */
    memcpy(encabezado + 0, ID_TRANS, 2);    /* Colocamos el ID de transaccion en
el encabezado */
    memcpy(encabezado + 2, INDICADORES, 2); /* Colocamos los indicadores para una
solicitud */
    memcpy(encabezado + 4, PETICIONES, 2);  /* Colocamos el contador de
peticiones en 1 */
}

int configurar_buffer(unsigned char encabezado[12], unsigned char direccion[50],
unsigned char buffer[512])
{
```

```
    int apuntador = 12;          /* Creamos un contador apuntado iniciado en 12
que es la posicon actual del buffer */
    unsigned char longCadena[2]; /* Almacenaremos el valor devuelto por strlen */

    memcpy(buffer + 0, encabezado, 12); /* Colocamos el encabezado en el buffer
*/

    /* Separamos la cadena por cada . en ella */
    unsigned char *token = strtok(direccion, ".");

    /* Si queremos que la función siga dividiendo nuestra cadena; en las
siguientes llamadas a la misma no debemos
    pasarle la cadena (sino NULL), y también los delimitadores. */
    if (token != NULL)
    {
        while (token != NULL)
        {
            *(short *)longCadena = htons(strlen(token));   /* Pasamos la longitud
de la cadena en hexadecimal */
            memcpy(buffer + apuntador, longCadena + 1, 1); /* Colocamos la
longitud de la cadena en el buffer */
            apuntador++;                                   /* Incrementamos en 1
el apuntador */

            memcpy(buffer + apuntador, token, strlen(token)); /* Colocamos la
cadena en el buffer */
            apuntador += strlen(token);                       /* Incrementamos el
apuntador con la longitud de la cadena */

            /* Sólo en la primera pasamos la cadena; en las siguientes pasamos
NULL */
            token = strtok(NULL, ".");
        }
    }

    memcpy(buffer + apuntador, CERO, 1); /* Colocamos cero en el buffer marcando
el final de la direccion web */
    apuntador++;                         /* Incrementamos en 1 el apuntador */

    memcpy(buffer + apuntador, REG_HOST, 2); /* Colocamos que es registro host en
el buffer */
    apuntador += 2;                          /* Incrementamos el apuntador en 2
*/

    memcpy(buffer + apuntador, CLASE_INT, 2); /* Colocamos la clase internet en
el buffer */
    apuntador += 2;                           /* Incrementamos el apuntador en 2
*/

    /* Devolvemos el apuntador */
    return apuntador;
}

/* Solicitamos al usuario la direccion web */
void pedir_direccion(unsigned char direccion[50])
{
printf("\nDireccion Web: ");
fgets(direccion, 50, stdin);
```

```
direccion[strlen(direccion) - 1] = '\0'; /* Eliminamos el salto de linea \n */
}

/* Comenzamos a idenificar y separar los datos que vienene en la trama recibida
*/
void respuesta_dns(unsigned char buffer[512])
{
    /* Iniciamos un apuntador en la posicon donde comienza la respuesta del DNS
*/
    int apuntador = 12 + strlen(buffer + 12) + 4 + 1 + 1;

    /* Mostramos los n registros de recursos */
    printf("\nRespuesta RRs: %d\n", buffer[7]);
    printf("----------------------------------------");
    /* Ciclamos por el numero de respuestas */
    for (int i = 0; i < (int)buffer[7]; i++)
    {
        apuntador = recorrer_trama(buffer, apuntador);
    }

    /* Mostramos los n registros de autoridad */
    printf("\nAutoridades RRs: %d\n", buffer[9]);
    printf("----------------------------------------");
    /* Ciclamos por el numero de respuestas */
    for (int i = 0; i < (int)buffer[9]; i++)
    {
        apuntador = recorrer_trama(buffer, apuntador);
    }

    /* Mostramos los n registros adicionales */
    printf("\nAdicionales RRs: %d\n", buffer[11]);
    printf("----------------------------------------");
    /* Ciclamos por el numero de respuestas */
    for (int i = 0; i < (int)buffer[11]; i++)
    {
        apuntador = recorrer_trama(buffer, apuntador);
    }
}

/* Imprimimos el tipo de registro */
void imprimir_tipo_registro(int tipo)
{
    switch (tipo)
    {
    case 1:
        /* Registro Host */
        printf("\nRegistro: Host");
        break;
    case 2:
        /* Registro (A) servidor de nombres */
        printf("\nRegistro: (A) servidor de nombres");
        break;
    case 5:
        /* Registro alias (CNAME) */
        printf("\nRegistro: alias (CNAME)");
        break;
    case 0:
        /* Registro de busqueda inversa */
```

```
        printf("\nRegistro: de busqueda inversa");
        break;
    case 28:
        /* Tiene IPv6 */
        printf("\nRegistro: AAAA (IPv6)");
        break;
    default:
        break;
    }
}

/* Imprimimos el tipo d clase (INTERNET XDXDXD) */
void imprimir_clase()
{
    printf("\nClase: Internet");
}

/* Imprimimos el tiempo de vida */
void imprimir_tiempo(unsigned char buffer[512], int p)
{
    // printf("\nTiempo de vida: %d s", buffer[p + 3]);
    printf("\nTiempo de vida: %u%u%u%u s", buffer[p], buffer[p + 1], buffer[p +
2], buffer[p + 3]);
    // printf("\nTiempo de vida: %d s", t);
}

/* Verificamos que sea apuntador */
int es_apuntador(unsigned char buffer[512], int point)
{
    if (buffer[point] == 192)
        return 1;
    else
        return 0;
}

/* Imprimimos n caracteres */
int imprimir_n_caracteres(unsigned char buffer[512], int num, int apuntador)
{
    for (int i = 0; i < num; i++)
    {
        apuntador++;
        printf("%c", buffer[apuntador]);
    }

    return apuntador + 1;
}

/* Hacemos un ciclo para imprimir los datos */
int ciclo_impresion(unsigned char buffer[512], int pointer)
{
    int p;

    for (int i = 0; i < strlen(buffer + 12); i++)
    {
        /* Mandamos como parametros el buffer, el tam a imprimir, y el apuntador
*/
        pointer = imprimir_n_caracteres(buffer, (int)buffer[pointer], pointer);
```

```
        if ((int)buffer[pointer] == 0)
        {
            break;
        }

        else if (es_apuntador(buffer, pointer) == 1)
        {
            printf(".");
            /* Incrementamos en un para quedar en el byte donde esta la posicion
*/
            pointer++;
            p = guardar_apuntador(buffer, p, pointer);
            p = ciclo_impresion(buffer, p);
            break;
        }

        else
        {
            /* Imprimimos el '.' */
            printf(".");
        }
    }

    return pointer;
}

/* Guardamos el apuntador en una variable */
int guardar_apuntador(unsigned char buffer[512], int point, int apuntador)
{
    point = (int)buffer[apuntador]; // Guardamos el apuntador
    return point;
}

/* Imprimimos la direccion IP */
int imprimir_ip(unsigned char buffer[512], int pointer)
{
    for (int i = 0; i < 4; i++)
    {
        printf("%d", buffer[pointer]);
        pointer++;

        if (es_apuntador(buffer, pointer) != 1)
        {
            // Ponemos el .
            printf(".");
        }
    }

    return pointer - 1;
}

/* Recorremos la trama recibida en busca de los datos */
int recorrer_trama(unsigned char buffer[512], int apuntador)
{
    int point; /* La usaremos para auxiliarnos con los punteros */

    /* Mostramos el nombre */
    printf("\nNombre: ");
```

```
    /* Guardamos la posicion que tiene el apuntador en la variable point */
    point = guardar_apuntador(buffer, point, apuntador);

    if ((int)buffer[12])
    {
        /* code */
    }

    point = ciclo_impresion(buffer, point);

    /* Incrementamos en +2 para quedar en el segundo byte del tipo de registro */
    apuntador += 2;
    imprimir_tipo_registro((int)buffer[apuntador]);

    /* Guardamos esta posicon para usarla mas adelante */
    int posRegistro = apuntador;

    /* Incrementemos en +2 para imprimir la clase internet
    + 1 para quedar en el primer byte del tiempo de vida */
    imprimir_clase();
    apuntador += 3;
    imprimir_tiempo(buffer, apuntador);

    /* Incrementamos en +4 para quedar en el primer byte del tam de los datos
    + 1 para quedar en el segundo byte */
    apuntador += 5;

    printf("\nTam de los datos = %d", buffer[apuntador]);

    /* Incrementamos en 1 para quedar en el primer byte, donde esta el tam de
cadenas  o la IP */
    apuntador++;

    /* Si es registro Host, entonces cambiamos el proceso de impresion un poco */
    if ((int)buffer[posRegistro] == 1)
    {
        printf("\nDireccion: ");
        apuntador = imprimir_ip(buffer, apuntador);
    }

    /* Si es IPv6, entonces cambiamos el proceso de impresion un poco */
    else if ((int)buffer[posRegistro] == 28)
    {
        printf("\nDireccion: ");
        apuntador = imprimir_ipv6(buffer, apuntador);
    }

    else
    {
        /* Si es registro Server, entonces cambiamos el proceso de impresion un
poco */
        if ((int)buffer[posRegistro] == 2)
            printf("\nServidor: ");
        else
            printf("\nCNAME: ");

        int bandera = 0;
```

```
        if (es_apuntador(buffer, apuntador) == 1)
        {
            apuntador++;
            point = guardar_apuntador(buffer, point, apuntador);
            point = ciclo_impresion(buffer, point);
            bandera = 1;
        }

        for (int i = 0; i < (int)buffer[apuntador - 1]; i++)
        {
            if (bandera == 1)
                break;

            /* Mandamos como parametros el buffer, el tam a imprimir, y el
apuntador */
            apuntador = imprimir_n_caracteres(buffer, (int)buffer[apuntador],
apuntador);

            if ((int)buffer[apuntador] == 0)
            {
                /* Si entramos aqui es porque es fin de la cadena */
                break;
            }

            else if (es_apuntador(buffer, apuntador) == 1)
            {
                printf(".");
                /* Incrementamos en un para quedar en el byte donde esta la
posicion */
                apuntador++;
                point = guardar_apuntador(buffer, point, apuntador);
                point = ciclo_impresion(buffer, point);
                break;
            }

            else
            {
                /* Imprimimos el '.' */
                printf(".");
            }
        }
    }

    /* Metemos un salto de linea para separar datos */
    printf("\n");

    /* Finalmente, incrementamos en +2 para quedar en el segundo byte, donde
estara el apuntador */
    apuntador += 2;

    return apuntador;
}

/* Imprimimos la IPv6 */
int imprimir_ipv6(unsigned char buffer[512], int apuntador)
{
    for (int i = 1; i <= 16; i++)
```

```
    {
        if (i > 6 && (int)buffer[apuntador] == 0)
        {
            apuntador++;
            continue;
        }

        printf("%x", buffer[apuntador]);
        apuntador++;

        if (i <= 6 && (int)buffer[apuntador] == 0)
            printf("0");

        if ((i) % 2 == 0)
        {
            if (i == 16)
                break;

            printf(":");
        }
    }

    return apuntador - 1;
}
```

Al realizar las pruebas de preferencia en un sistema operativo Linux se tiene la siguiente respuesta después de compilar con el siguiente comando:

```
gcc dns.c -o dns
```

La ejecución en la terminal de Ubuntu es como la que se muestra a continuación: