UCS 1511 - Network Lab

Exercise 5 - Domain Name Server using UDP

Name:	Mahesh Bharadwaj K
Reg No:	185001089
Semester:	V
Date:	September 22, 2020

Aim:

To simulate the concept of Domain Name Server using UDP..

Algorithm

1. Server

1. Initialise DNS table with URL and corresponding IP addresses.
2. Prompt admin to add new URLs and IP after verifying valid IP address and ensuring it is not duplicate.
3. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_DGRAM (for UDP protocol), default protocol and store as sockfd.
4. If sockfd is a negative number, socket creation failed, end program.
5. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to INADDR_ANY to accept connections from any client and port number required.
6. Bind newly created socket to addresss given in sockaddr_in.
7. If bind is non zero, bind failed, print error message and terminate.
8. Read URL requested into buffer using **recvfrom()** system call.
9. LOOP through DNS table:
   - IF domain name matches requested URL, copy all mapped IP addresses into buffer
   - ELSE write 'URL not Found Error!' onto buffer
10. Send response to client using **sendto()** system call, goto step 8.
11. Close connections on socket using **close()** and terminate program.

2. Client

1. Create a socket descriptor with **socket()** system call with AF_INET (IPV4 domain), SOCK_DGRAM, default protocol and store as sockfd.
2. If sockfd is a negative number, socket creation failed, end program.
3. Create sockaddr_in object to assign IP address and Port number for socket. Set family to AF_INET, IP address to localhost(127.0.0.1) to connect to server and port number required.
4. BEGIN LOOP
   - Read URL from use onto buffer.
   - IF input is 'END' terminate program.
   - Send the URL to server using **sendto()** system call.
   - Read the response from server onto buffer using **recvfrom()** system call.
   - IF response is 'URL not Found Error!', print error message to user
   - ELSE, display all received addresses to user.
5. Close the connections on socket using **close()** and terminate program.

Program

1. Server Side

```
#include <stdio.h>
#include <netdb.h>
#include <fcntl.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>

#include "DNS.h"

int main(int argc, char **argv)
{

        Entry table[MAX_DOMAIN], result;
        bzero(table, MAX_DOMAIN * sizeof(Entry));

        if (argc < 2)
        {
                fprintf(stderr, "Error: Enter port number for server as second
                ↪  argument!\n");
                exit(EXIT_FAILURE);
        }

        int PORT = atoi(argv[1]);
        int sockfd, len;
        struct sockaddr_in servaddr, cliadrr;
        char buff[30];
        int n;

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        if (sockfd == -1)
        {
                fprintf(stderr, "Error: Socket creation failed!\n");
```

```
        exit(EXIT_FAILURE);
}
else
        printf("Socket creation successfull!\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);

// Binding newly created socket to given IP and verification
if ((bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr))) != 0)
{
        fprintf(stderr, "Error: Socket bind failed!\n");
        exit(EXIT_FAILURE);
}
else
        printf("Socket bind successfull\n");

len = sizeof(cliadrr);

createEntry(table, "google.com", "192.168.1.1");
createEntry(table, "yahoo.com", "194.12.34.12");
createEntry(table, "google.com", "17.10.23.123");

printTable(table);

string domain, address, opt;

while (1)
{
        recvfrom(sockfd, buff, sizeof(buff), MSG_WAITALL, (struct sockaddr
        ↪  *)&cliadrr, &len);
        result = getAddress(table, buff);
        sendto(sockfd, &result, sizeof(Entry), MSG_CONFIRM, (struct sockaddr
        ↪  *)&cliadrr, len);

        int flag = 0;

        printf("Do you want to modify (yes/no): ");
        scanf("%s", opt);
        if (strcmp(opt, "yes") == 0)
        {
                printf("Enter domain: ");
                scanf("%s", domain);
                do
                {
                        printf("Enter IP address: ");
                        scanf("%s", address);
                        flag = createEntry(table, domain, address);
                        switch (flag)
                        {
                        case 1:
                                break; // Correct IP
                        case -1:
                                printf("Invalid IP address!\n");
                                break;
                        case -2:
                                printf("Duplicate IP address!\n");
```

```
                                        break;
                                default:
                                        printf("Error!\n");
                                }
                        } while (flag != 1);

                        printf("Updated table\n");
                        printTable(table);
                }
        }

        close(sockfd);
}
```

---

2. Client Side

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <sys/socket.h>

#define MAX 1024

#include "DNS.h"

#define SA struct sockaddr
int main(int argc, char **argv)
{
        if (argc < 2)
        {
                fprintf(stderr, "Please pass port number of server as second
                ↪  argument!\n");
                exit(EXIT_FAILURE);
        }
        int PORT = atoi(argv[1]);

        Entry query;
        int sockfd, connfd;
        struct sockaddr_in servaddr, cli;
        char buff[30] = {0};

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        if (sockfd == -1)
        {
                fprintf(stderr, "Error: Socket creation failed!\n");
                exit(EXIT_FAILURE);
        }
        else
                printf("Socket creation successfull!\n");

        bzero(&servaddr, sizeof(servaddr));
        // assign IP, PORT
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
        servaddr.sin_port = htons(PORT);
        int len = sizeof(Entry);
        while(1)
```

```
        {

                bzero(&query, sizeof(Entry));
                printf("Enter the domain name: ");
                scanf(" %[^\n]", query.domain);

                if (strcmp(query.domain, "END") == 0)
                        break;

                sendto(sockfd, query.domain, sizeof(query.domain), MSG_CONFIRM,
                ↪  (struct sockaddr *)&servaddr, sizeof(servaddr));
                recvfrom(sockfd, &query, sizeof(Entry), MSG_WAITALL, (struct sockaddr
                ↪  *)&servaddr, &len);

                if (!query.address[0][0])
                        printf("No entry in DNS!\n");
                else
                {
                        printf("The IP Address is: \n");
                        for (int i = 0; i < MAX_ADDR; i++)
                        {
                                if (query.address[i][0])
                                        printf("%s\n", query.address[i]);
                        }
                        printf("\n");
                }
        }

        close(sockfd);
}
```

---

3. DNS Specific Functions

```
#define MAX_ADDR 10
#define MAX_DOMAIN 20
typedef char string[30];

typedef struct Entry
{
    string domain;
    string address[MAX_ADDR];
} Entry;

void printTable(Entry table[MAX_DOMAIN])
{
    printf("+-----------------+----------------------+\n");
    printf("|   Domain Name   |        Address       |\n");
    printf("+-----------------+----------------------+\n");
    for (int i = 0; i < MAX_DOMAIN; i++)
    {
        if (table[i].domain[0])
        {
            printf("| %-15s | %-20s |\n", table[i].domain, table[i].address[0]);

            for (int j = 1; j < MAX_ADDR && table[i].address[j][0]; j++)
                printf("| %-15s | %-20s |\n", "", table[i].address[j]);
            printf("+-----------------+----------------------+\n");
        }
    }
```

```c
    printf("\n");
}

int checkAddress(Entry table[MAX_DOMAIN], char *const address)
{
    string addr_copy;
    strcpy(addr_copy, address);
    char *split;
    int val;
    split = strtok(addr_copy, ".");
    while (split)
    {
        val = atoi(split);
        if (val < 0 || val > 255)
            return -1;
        split = strtok(NULL, ".");
    }

    for (int i = 0; i < MAX_DOMAIN; i++)
    {
        if (!table[i].domain[0])
            continue;

        for (int j = 0; j < MAX_ADDR && table[i].address[j][0]; j++)
            if (strcmp(address, table[i].address[j]) == 0)
                return -2;
    }

    return 0;
}

int createEntry(Entry table[MAX_DOMAIN], char *domain, char *address)
{
    // Search if entry exists already
    int index = -1;
    int flag = 0;

    int addr_invalid = checkAddress(table, address);

    if (addr_invalid)
        return addr_invalid;

    for (int i = 0; i < MAX_DOMAIN; i++)
    {
        if (strcmp(table[i].domain, domain) == 0)
        {
            for (int j = 0; j < MAX_ADDR; j++)
                if (!table[i].address[j][0])
                {
                    strcpy(table[i].address[j], address);
                    flag = 1;
                    break;
                }
            break;
        }
        if (!table[i].domain[0] && index == -1)
            index = i;
    }

    // IF entry has to be created
    if (!flag)
```

```
    {
        strcpy(table[index].domain, domain);
        strcpy(table[index].address[0], address);
        flag = 1;
    }

    return flag;
}

Entry getAddress(Entry *table, char *const domain)
{
    Entry result;
    bzero(&result, sizeof(Entry));
    strcpy(result.domain, domain);

    for (int i = 0; i < MAX_DOMAIN; i++)
    {

        if (strcmp(table[i].domain, domain) == 0)
        {
            for (int j = 0; j < MAX_ADDR; j++)
            {
                strcpy(result.address[j], table[i].address[j]);
            }
            break;
        }
    }
    return result;
}
```

Output

Figure 1: **Client Program Output**

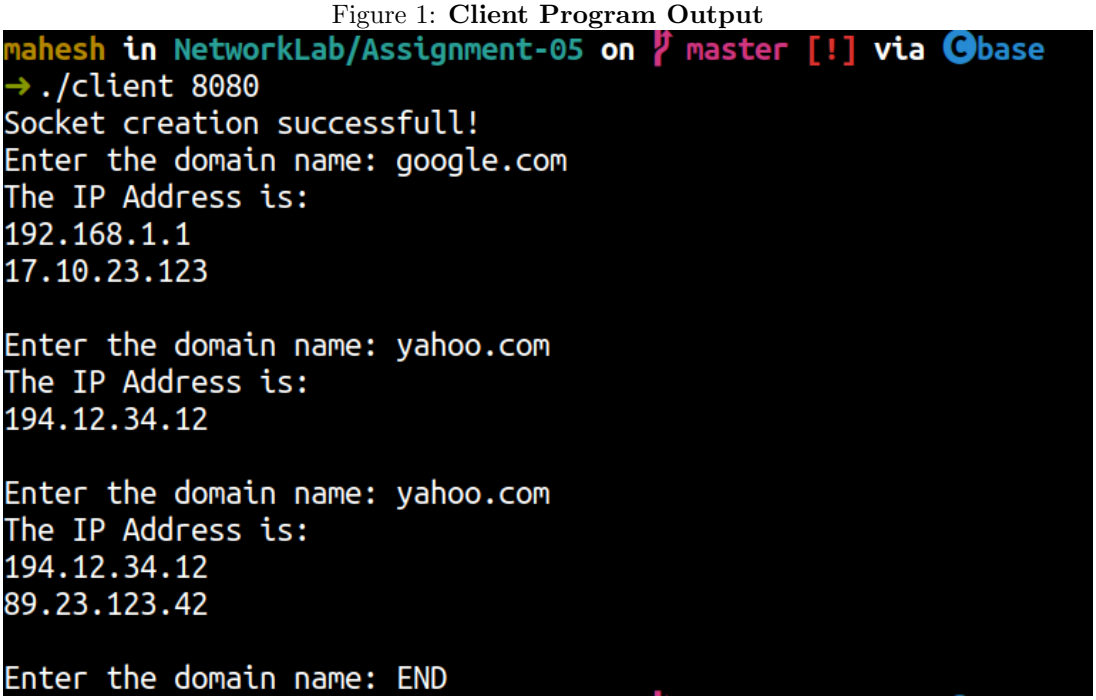

Figure 2: **Server Program Output**

```
mahesh in NetworkLab/Assignment-05 on master [!] via base
➜ ./server 8080
Socket creation successfull!
Socket bind successfull
+-----------------+----------------------+
|   Domain Name   |        Address       |
+-----------------+----------------------+
| google.com      | 192.168.1.1          |
|                 | 17.10.23.123         |
+-----------------+----------------------+
| yahoo.com       | 194.12.34.12         |
+-----------------+----------------------+

Do you want to modify (yes/no): no
Do you want to modify (yes/no): no
Do you want to modify (yes/no): yes
Enter domain: yahoo.com
Enter IP address: 190.266.23.51
Invalid IP address!
Enter IP address: 192.168.1.1
Duplicate IP address!
Enter IP address: 89.23.123.42
Updated table
+-----------------+----------------------+
|   Domain Name   |        Address       |
+-----------------+----------------------+
| google.com      | 192.168.1.1          |
|                 | 17.10.23.123         |
+-----------------+----------------------+
| yahoo.com       | 194.12.34.12         |
|                 | 89.23.123.42         |
+-----------------+----------------------+

Do you want to modify (yes/no): ^[
```

Learning Outcomes:

- We learn how to create a UDP client server connection.
- We learn to maintain a table for performing DNS functions.
- We learn to validate IP addresses.
- We learn to work with connection-less protocol.