SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NETWORKS LAB
EXERCISE 5

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

DOMAIN NAME SERVER USING UDP

Aim:

To simulate the concept of Domain Name Server using UDP

Algorithm:

SERVER

1. Maintain a DNS in the form of table. The table contains IP address and the corresponding Server name and displays the table.
2. When a request is for an IP address (Given a server name), from a client is received, verify the table and send the corresponding IP address to the client.
3. Make server to accept multiple client request simultaneously.
4. Also modify the server.

CLIENT

1. Request for an IP address is given to the server by the domain name.
2. Receive the corresponding IP address and display it.

Code:
Server

```c
#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>
```

```c
#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");
        }
    }
    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;
}


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);
}
```

Client

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

#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");
        }
```

```
    }
    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;
}

#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);
    }
```

```c
    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);
}
```

Output:

Server :

```
root@spl13:~/Desktop/Jayannthan/Assignment-05# ./s 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): yes
Enter domain: yahoo.com
Enter IP address: 194.12.35.65
Updated table
+-----------------+-----------------------+
|   Domain Name   |        Address        |
+-----------------+-----------------------+
| google.com      | 192.168.1.1           |
|                 | 17.10.23.123          |
|                 | 192.168.2.1           |
+-----------------+-----------------------+
| yahoo.com       | 194.12.34.12          |
|                 | 194.12.35.65          |
+-----------------+-----------------------+
```

```
Do you want to modify (yes/no): yes
Enter domain: google.com
Enter IP address: 192.168.2.1
Updated table
+-----------------+-----------------------+
|   Domain Name   |        Address        |
+-----------------+-----------------------+
| google.com      | 192.168.1.1           |
|                 | 17.10.23.123          |
|                 | 192.168.2.1           |
+-----------------+-----------------------+
| yahoo.com       | 194.12.34.12          |
+-----------------+-----------------------+
```

Client:

```
Enter the domain name: yahoo.com
The IP Address is:
194.12.34.12
```

```
root@spl13:~/Desktop/Jayannthan/Assignment-05# ./c 8080
Socket creation successfull!
Enter the domain name: google.com
The IP Address is:
192.168.1.1
17.10.23.123
```

Learning outcome:

Learnt the working of domain name server

Learnt to simulate the working of domain name server using UDP