Pradeep KM
3122225001092

Exercise 6: Domain Name Server using UDP

Server algorithm

1. Create a UDP socket.
2. Bind the socket to the DNS port (5353).
3. Enter an infinite loop to continuously listen for incoming requests.
4. Receive the domain name from the client via UDP.
5. Search the domain name in the local DNS table.
6. If found, prepare the corresponding IP address.
7. If not found, prepare a default IP ("0.0.0.0").
8. Print the received domain and the corresponding IP address.
9. Send the IP address back to the client via UDP.
10. Repeat from step 4.

Client algorithm

1. Create a UDP socket.
2. Prepare the domain name to be queried (e.g., "www.google.com").
3. Send the domain name to the server via UDP.
4. Wait to receive the IP address from the server.
5. Display the received IP address.
6. Close the socket.

server.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define DNS_PORT 5353
#define BUFFER_SIZE 1024

typedef struct {
    char domain[256];
    char ip[16];
} Dns_Table;

Dns_Table dns_table[] = {
    {"www.google.com", "142.250.190.46"},
    {"www.example.com", "93.184.216.34"},
    {"www.facebook.com", "157.240.7.35"},
    {"www.github.com", "140.82.112.3"}
};

int find_ip_for_domain(const char *domain) {
    for (int i = 0; i < 4; i++) {
```

```
        if (strcmp(domain, dns_table[i].domain) == 0) {
            return i;
        }
    }
    return -1;
}

int main() {
    int udp_sock;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];
    char domain[256];
    char ip[16];

    udp_sock = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(DNS_PORT);

    bind(udp_sock, (struct sockaddr*)&server_addr, sizeof(server_addr));

    while (1) {
        recvfrom(udp_sock, buffer, sizeof(buffer), MSG_WAITALL, (struct sockaddr*)&client_addr,
&addr_len);
        sscanf(buffer, "%s", domain);

        int index = find_ip_for_domain(domain);
        if (index != -1) {
            strcpy(ip, dns_table[index].ip);
        } else {
            strcpy(ip, "0.0.0.0");
        }

        printf("Received request for domain: %s, responding with IP: %s\n", domain, ip);
        sendto(udp_sock, ip, strlen(ip) + 1, 0, (struct sockaddr*)&client_addr, addr_len);
    }

    close(udp_sock);
    return 0;
}
```

client.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
```

Pradeep KM
3122225001092

```
#define DNS_PORT 5353
#define BUFFER_SIZE 1024

int main() {
    int udp_sock;
    struct sockaddr_in server_addr;
    char domain[256] = "www.google.com";
    char ip[16];
    socklen_t addr_len = sizeof(server_addr);

    udp_sock = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&server_addr, 0, addr_len);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(DNS_PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    sendto(udp_sock, domain, strlen(domain) + 1, MSG_CONFIRM, (struct sockaddr*)&server_addr,
addr_len);

    recvfrom(udp_sock, ip, sizeof(ip), MSG_WAITALL, (struct sockaddr*)&server_addr, &addr_len);

    printf("IP Address for %s: %s\n", domain, ip);

    close(udp_sock);
    return 0;
}
```

Output