# UDP DNS Server Program

## 19Z510 – COMPUTER NETWORKS LABORATORY

Anandkumar NS (22Z209)

**BACHELOR OF ENGINEERING**

Date: 19/08/2024

**DEPARTMENT OF COMPUTER SCIENCE**

**ENGINEERING PSG COLLEGE OF TECHNOLOGY**

(Autonomous Institution)

**COIMBATORE – 641 004**

Client:

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

#define MAX 80
#define PORT 3000
#define SA struct sockaddr

void send_request(int sockfd, struct sockaddr_in *servaddr) {
while(1){
  char buffer[MAX];
  char response[MAX];
  socklen_t len = sizeof(*servaddr);

  // Get URL from user
  printf("Enter the URL: ");
  fgets(buffer, sizeof(buffer), stdin);
  buffer[strcspn(buffer, "\n")] = '\0';  // Remove newline character

  if (strcmp(buffer,"exit")==0){
break;
}
else{      // Send URL to server
  sendto(sockfd, buffer, strlen(buffer), 0, (SA*)servaddr, len);
  printf("URL Sent\n");
  // Receive response from server
  recvfrom(sockfd, response, sizeof(response), 0, NULL, NULL);
  printf("Server response: %s\n", response);
}
}
}

int main() {
  int sockfd;
  struct sockaddr_in servaddr;

  // Create UDP socket
  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  if (sockfd < 0) {
    perror("Socket creation failed");
    exit(EXIT_FAILURE);
  }
```

```
        printf("UDP socket created\n");

        // Initialize server address
        memset(&servaddr, 0, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
        servaddr.sin_port = htons(PORT);

        // Send request and receive response
        send_request(sockfd, &servaddr);

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

Server:

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

#define MAX 80
#define PORT 3000
#define SA struct sockaddr

// Function to look up the IP address for the given URL
void handle_request(int sockfd, struct sockaddr_in *client_addr, socklen_t client_len) {
  char buffer[MAX];
  char response[MAX];
  FILE *fp;
  char url[MAX];
  char ip[MAX];
  socklen_t len = sizeof(*client_addr);

  // Receive URL from client
  ssize_t recv_len = recvfrom(sockfd, buffer, sizeof(buffer) - 1, 0, (SA*)client_addr,
&client_len);
  if (recv_len < 0) {
    perror("Receive failed");
    return;
  }
  buffer[recv_len] = '\0';  // Null-terminate the received string

  // Debugging: Print the received URL
```

```
    printf("Received URL: %s\n", buffer);

    // Open the text file containing URL-IP pairs
    fp = fopen("data.txt", "r");
    if (fp == NULL) {
        perror("Unable to open file");
        snprintf(response, sizeof(response), "Server error");
        sendto(sockfd, response, strlen(response), 0, (SA*)client_addr, len);
        return;
    }

    // Initialize response with "URL not found" message
    snprintf(response, sizeof(response), "URL not found");

    // Read the file line by line
    while (fscanf(fp, "%s %s", url, ip) != EOF) {
        if (strcmp(buffer, url) == 0) {
            snprintf(response, sizeof(response), "%s", ip);
            break;
        }
    }

    fclose(fp);

    // Debugging: Print the response to be sent
    printf("Sending response: %s\n", response);

    // Send the response to the client
    sendto(sockfd, response, strlen(response), 0, (SA*)client_addr, len);
}

int main() {
    int sockfd;
    struct sockaddr_in servaddr, cli;
    socklen_t len = sizeof(cli);

    // Create UDP socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
    printf("UDP socket created\n");

    // Initialize server address
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);

    // Bind the socket
    if (bind(sockfd, (SA*)&servaddr, sizeof(servaddr)) < 0) {
        perror("Bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }
    printf("UDP server listening on port %d\n", PORT);

    // Handle incoming requests
    while (1) {
        handle_request(sockfd, &cli, len);
    }

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