

## IMPLEMENTING THE DNS ITERATIVE

### ROOT DNS SERVER

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

#define ROOTPORT 8041
extern int errno;

int main()
{
    int sockfd = 0, clientfd = 0, sentbytes, recvbytes;
    socklen_t length = sizeof(struct sockaddr_in);
    struct sockaddr_in host_addr, client_addr;
    char buffer[20];
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        fprintf(stderr, "Error in socket creation.\n");
        return -1;
    }

    host_addr.sin_family = AF_INET;
    host_addr.sin_port = htons(ROOTPORT);
    inet_pton(AF_INET, "127.0.0.1", &host_addr.sin_addr);
    if (bind(sockfd, (struct sockaddr *)&host_addr, sizeof(host_addr)) < 0)
    {
        fprintf(stderr, "Error in binding port to socket.\n");
        return -1;
    }
    printf("ROOT DNS RESOLVER STARTED AT PORT : %d\n", ROOTPORT);

    while (1)
    {
        recvbytes = recvfrom(sockfd, buffer, sizeof(buffer), 0,
            (struct sockaddr *)&client_addr, &length);
        fprintf(stdout, "REQUEST FROM : %s\n", buffer);
        FILE *fd = fopen("rootdns.txt", "r");
        if (!fd)
        {
            fprintf(stderr, "Could not access DNS records.\n");
            sendto(sockfd, "ERROR", strlen("ERROR") + 1, 0,
                (struct sockaddr *)&client_addr, length);
            continue;
        }

        char linebuff[40], filebuff[400], ip[20], tempbuff[40], lastbuff[40];
        char *temp, *iptemp;
        int flag = 0, i;
        linebuff[0] = '\0';
        lastbuff[0] = '\0';
        filebuff[0] = '\0';
        ip[0] = '\0';
```

```

while (fgets(linebuff, sizeof(linebuff), fd))
{
    strcpy(tempbuff, linebuff);
    temp = strtok(tempbuff, " ");
    if (flag == 0 && strncmp(temp, buffer, strlen(temp)) == 0)
    {
        flag = 1;
        strcpy(lastbuff, linebuff);
        iptemp = strtok(NULL, "\n");
        for (i = 0; *iptemp != '\0'; i++, iptemp++)
            ip[i] = *iptemp;
        ip[i] = '\0';
    }
    else
    {
        strcat(filebuff, linebuff);
    }
}

fclose(fd);
if (flag == 0)
{
    sentbytes = sendto(socketfd, "404", strlen("404") + 1, 0, (struct sockaddr*)
    &client_addr, length);
}
else
{
    int fdes = open("rootdns.txt", O_WRONLY);
    strcat(filebuff, lastbuff);
    write(fdes, filebuff, strlen(filebuff));
    close(fdes);
    fprintf(stdout, "TOP LEVEL DOMAIN IP : %s\n\n", ip);
    sentbytes = sendto(socketfd, ip, strlen(ip) + 1, 0, (struct sockaddr*)
    &client_addr, length);
}
}
close(socketfd);
return 0;
}

```

## TOP LEVEL DOMAIN

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>

#define TLDPORT 8042
extern int errno;

int main()
{
    int socketfd = 0, clientfd = 0, sentbytes, recvbytes;
    socklen_t length = sizeof(struct sockaddr_in);
    struct sockaddr_in host_addr, client_addr;
    char buffer[20];

    socketfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (socketfd < 0)
    {
        fprintf(stderr, "Error in socket creation.\n");
        return -1;
    }
}

```

```

host_addr.sin_family = AF_INET;
host_addr.sin_port = htons(TLDPOR);
inet_pton(AF_INET, "127.0.0.1", &host_addr.sin_addr);
if (bind(socketfd, (struct sockaddr *)&host_addr, sizeof(host_addr)) < 0)
{
    fprintf(stderr, "Error in binding port to socket.\n");
    return -1;
}
printf("TOP LEVEL DOMAIN SERVER %d\n", TLDPOR);

while (1)
{
    recvbytes = recvfrom(socketfd, buffer, sizeof(buffer), 0, (struct sockaddr*)
    &client_addr, &length);
    fprintf(stdout, "REQUEST FROM CLIENT : %s\n", buffer);
    FILE *fd = fopen("tlddns.txt", "r");
    if (!fd)
    {
        fprintf(stderr, "Could not access DNS records.\n");
        sendto(socketfd, "ERROR", strlen("ERROR") + 1, 0, (struct sockaddr
        *)&client_addr, length);
        continue;
    }

    char linebuff[40], filebuff[400], ip[20], tempbuff[40], lastbuff[40];
    char *temp, *iptemp;
    int flag = 0, i;
    linebuff[0] = '\0';
    lastbuff[0] = '\0';
    filebuff[0] = '\0';
    ip[0] = '\0';

    while (fgets(linebuff, sizeof(linebuff), fd))
    {
        strcpy(tempbuff, linebuff);
        temp = strtok(tempbuff, " ");
        if (flag == 0 && strcmp(temp, buffer, strlen(temp)) == 0)
        {
            flag = 1;
            strcpy(lastbuff, linebuff);
            iptemp = strtok(NULL, "\n");
            for (i = 0; *iptemp != '\0'; i++, iptemp++)
                ip[i] = *iptemp;
            ip[i] = '\0';
        }
        else
        {
            strcat(filebuff, linebuff);
        }
    }
    fclose(fd);
    if (flag == 0)
    {
        sentbytes = sendto(socketfd, "404", strlen("404") + 1, 0, (struct sockaddr*)
        &client_addr, length);
    }
    else
    {
        int fdes = open("tlddns.txt", O_WRONLY);
        strcat(filebuff, lastbuff);
        write(fdes, filebuff, strlen(filebuff));
        close(fdes);
        fprintf(stdout, "AUTHORITATIVE SERVER IP : %s\n\n",
        ip);
        sentbytes = sendto(socketfd, ip, strlen(ip) + 1, 0, (struct sockaddr*)
        &client_addr, length);
    }
}

close(socketfd);
return 0;
}

```

## AUTHORIZATION SERVER CODE

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

#define IPLOOKUP_TABLE_COUNT 4
#define IP_FOR_EACH_DNS_RECORDS 3
#define AUTHPORT 8043
extern int errno;

typedef struct
{
    char *key;
    int value;
} keyValuePairs;

keyValuePairs ip_lookuptable[] = {
{"www.cricbuzz.com", 0},
{"mail.google.com", 0},
{"cric.cricbuzz.com", 0}};

int rotate_dns_ip(char *domain_name)
{
    for (int i = 0; i < IPLOOKUP_TABLE_COUNT; i++)
    {
        if (strcmp(domain_name, ip_lookuptable[i].key) == 0)
        {
            int value = ip_lookuptable[i].value;
            ip_lookuptable[i].value++;
            return value;
        }
    }
    return -1;
}

int main()
{
    int sockfd = 0, clientfd = 0, sentbytes, recvbytes;
    socklen_t length = sizeof(struct sockaddr_in);
    struct sockaddr_in host_addr, client_addr;
    char buffer[20];
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        fprintf(stderr, "Error in socket creation.\n");
        return -1;
    }

    host_addr.sin_family = AF_INET;
    host_addr.sin_port = htons(AUTHPORT);
    inet_pton(AF_INET, "127.0.0.1", &host_addr.sin_addr);
    if (bind(sockfd, (struct sockaddr *)&host_addr, sizeof(host_addr)) < 0)
    {
        fprintf(stderr, "Error in binding port to socket.\n");
        return -1;
    }
    printf("AUTHORITATIVE DNS SERVER PORT : %d\n", AUTHPORT);

    while(1) {
        recvbytes = recvfrom(sockfd, buffer, sizeof(buffer), 0,
            (struct sockaddr *)&client_addr, &length);
        fprintf(stdout, "DNS QUERY : %s\n", buffer);
        FILE *fd = fopen("authdns.txt", "r");
```

```

if (!fd)
{
    fprintf(stderr, "Could not access DNS records.\n");
    sendto(socketfd, "ERROR", strlen("ERROR") + 1, 0, (struct sockaddr*)
    &client_addr, length);
    continue;
}
char linebuff[80], filebuff[400], ip[40], tempbuff[80],
lastbuff[80];
char *temp, *iptemp;
int flag = 0, i;
linebuff[0] = '\0';
lastbuff[0] = '\0';
filebuff[0] = '\0';
ip[0] = '\0';

while (fgets(linebuff, sizeof(linebuff), fd))
{
    strcpy(tempbuff, linebuff);
    temp = strtok(tempbuff, " ");
    if (flag == 0 && strcmp(temp, buffer, strlen(temp)) == 0)
    {
        flag = 1;
        strcpy(lastbuff, linebuff);
        iptemp = strtok(NULL, " ");
        int counter = 0;
        int curr_pointer =
        rotate_dns_ip(buffer) % IP_FOR_EACH_DNS_RECORDS;
        int i = 0;

        while (1)
        {
            for (i = 0; *iptemp != ' ' && *iptemp != '\0'; i++, iptemp++)
                ip[i] = *iptemp;
            if (*iptemp == '\n' || counter == curr_pointer)
                break;
            counter++;
            iptemp = strtok(NULL, " ");
        }
        ip[i] = '\0';
    }
    else
    {
        strcat(filebuff, linebuff);
    }
}

fclose(fd);
if (flag == 0)
{
    sentbytes = sendto(socketfd, "404", strlen("404") + 1, 0, (struct sockaddr*)
    &client_addr, length);
}
else
{
    int fdes = open("authdns.txt", O_WRONLY);
    strcat(filebuff, lastbuff);
    write(fdes, filebuff, strlen(filebuff));
    close(fdes);
    fprintf(stdout, "AUTHORITATIVE SERVER IP : %s\n\n", ip);
    sentbytes = sendto(socketfd, ip, strlen(ip) + 1, 0, (struct sockaddr*)
    &client_addr, length);
}
}

close(socketfd);
return 0;
}

```

## LOCAL DNS

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

#define ROOTPORT 8041
#define TLDPORT 8042
#define AUTHPORT 8043
#define PORT 8044

int main()
{
    int sockfd = 0, localfd = 0;
    int rootfd = 0, tldfd = 0, authfd = 0;
    socklen_t length = sizeof(struct sockaddr_in);
    struct sockaddr_in host_addr, root_addr, tld_addr, auth_addr, client_addr;
    char buffer[512], root[20], tld[30], auth[100];
    char rootip[30], tldip[30], authip[30];
    int rcvbytes, sentbytes;
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        fprintf(stderr, "Error in socket creation.\n");
        return -1;
    }

    host_addr.sin_family = AF_INET;
    host_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, "127.0.0.1", &host_addr.sin_addr);
    if (bind(sockfd, (struct sockaddr *)&host_addr, sizeof(host_addr)) < 0)
    {
        fprintf(stderr, "Error in binding port to socket.\n");
        return -1;
    }
    printf(" LOCAL DNS PORT : %d\n", PORT);

    while (1)
    {
        rcvbytes = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)&client_addr, &length);
        if (strncmp(buffer, "exit", sizeof("exit")) == 0)
        {
            fprintf(stdout, "exiting");
            break;
        }

        fprintf(stdout, "Request from client : %s\n\n", buffer);
        strcpy(auth, buffer);
        int i = 0, j = 0, k = 0;
        while (buffer[i++] != '.')
        ;
        while (buffer[i] != '.')
        {
            tld[j++] = buffer[i++];
        }
        tld[j++] = buffer[i++];
        while (buffer[i] != ' ' && buffer[i] != '\0')
        {
            tld[j++] = buffer[i];
            root[k++] = buffer[i];
            i++;
        }
        tld[j] = '\0';
        root[k] = '\0';
    }
}
```

```

//fprintf(stdout, "\t\t[RESOLVING DNS QUERY]\n\n");
rootfd = socket(AF_INET, SOCK_DGRAM, 0);
if (rootfd < 0)
{
    fprintf(stderr, "Error in socket creation.\n");
    return -1;
}

root_addr.sin_family = AF_INET;
root_addr.sin_port = htons(ROOTPORT);
inet_pton(AF_INET, "127.0.0.1", &root_addr.sin_addr);
sentbytes = sendto(rootfd, root, strlen(root) + 1, 0,
(struct sockaddr *)&root_addr, length);
recvbytes = recvfrom(rootfd, rootip, sizeof(rootip), 0, NULL, NULL);

//fprintf(stdout, "[ROOT DNS SERVER]\n\n");
fprintf(stdout, " [#] TLD server IP for %s:%s\n\n", root, rootip);
close(rootfd);
tldfd = socket(AF_INET, SOCK_DGRAM, 0);
if (tldfd < 0)
{
    fprintf(stderr, "Error in socket creation.\n");
    return -1;
}

tld_addr.sin_family = AF_INET;
tld_addr.sin_port = htons(TLDPORT);
inet_pton(AF_INET, "127.0.0.1", &tld_addr.sin_addr);
sentbytes = sendto(tldfd, tld, strlen(tld) + 1, 0, (struct sockaddr *)&tld_addr, length);
recvbytes = recvfrom(tldfd, tldip, sizeof(tldip), 0, NULL, NULL);

//fprintf(stdout, "[TLD SERVER]\n\n");
fprintf(stdout, " [#] Auth server IP for %s:%s\n\n", tld, tldip);
close(tldfd);
authfd = socket(AF_INET, SOCK_DGRAM, 0);
if (authfd < 0)
{
    fprintf(stderr, "Error in socket creation.\n");
    return -1;
}

auth_addr.sin_family = AF_INET;
auth_addr.sin_port = htons(AUTHPORT);
inet_pton(AF_INET, "127.0.0.1", &auth_addr.sin_addr);
sentbytes = sendto(authfd, auth, strlen(auth) + 1, 0,
(struct sockaddr *)&auth_addr, length);
recvbytes = recvfrom(authfd, authip, sizeof(authip), 0, NULL, NULL);

// fprintf(stdout, "[AUTHORITATIVE SERVER]\n\n");
if (strcmp(authip, "404") == 0)
    fprintf(stdout, "DNS RECORDS NOT FOUND : %s \n", auth);
else
    fprintf(stdout, " [#] Server IP for %s: %s\n\n", auth, authip);
close(authfd);
sentbytes = sendto(socketfd, authip, strlen(authip) + 1, 0, (struct sockaddr *)
&client_addr, length);
}

close(socketfd);
return 0;
}

```

## CLIENT

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define LOCALDNS 8044

int main()
{
    int sockfd = 0, sentbytes, recvbytes;
    struct sockaddr_in host_addr;
    char input[20], buffer[20];
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        fprintf(stderr, "Error in socket creation.\n");
        return -1;
    }
    host_addr.sin_family = AF_INET;
    host_addr.sin_port = htons(LOCALDNS);
    inet_pton(AF_INET, "127.0.0.1", &host_addr.sin_addr);

    while (1)
    {
        fprintf(stdout, "\n [-] Enter the HostName: ");
        scanf("%s", input);
        sentbytes = sendto(sockfd, input, strlen(input) + 1, 0,
            (struct sockaddr *)&host_addr, sizeof(host_addr));
        if (strncmp(input, "exit", sizeof("exit")) == 0)
            break;
        recvbytes = recvfrom(sockfd, buffer, sizeof(buffer), 0, NULL, NULL);
        if (strcmp("404", buffer) == 0)
            printf("DNS RECORDS NOT FOUND FOR %s\n", input);
        else
            printf(" SERVER IP OF %s : %s\n", input, buffer);
    }
    close(sockfd);
    return 0;
}
```

## OUTPUT :-

### CLIENT

```
[#] Enter the HostName : www.cricbuzz.com
SERVER IP OF www.cricbuzz.com : 83.78.55.120

[#] Enter the HostName : www.cricbuzz.com
SERVER IP OF www.cricbuzz.com : 83.78.55.135

[#] Enter the HostName : www.cricbuzz.com
SERVER IP OF www.cricbuzz.com : 83.78.55.170

[#] Enter the HostName : cric.cricbuzz.com
SERVER IP OF cric.cricbuzz.com : 37.68.23.143

[#] Enter the HostName : cric.cricbuzz.com
SERVER IP OF cric.cricbuzz.com : 37.69.23.143
```



## LOCAL DNS

Request from client : www.cricbuzz.com

[#] TLD server IP for com : 10.2.3.45

[#] Auth server IPL for cricbuzz.com : 10.12.86.142

[#] Server IP for www.cricbuzz.com : 83.78.55.120

Request from client : www.cricbuzz.com

[#] TLD server IP for com : 10.2.3.45

[#] Auth server IPL for cricbuzz.com : 10.12.86.142

[#] Server IP for www.cricbuzz.com : 83.78.55.135

Request from client : www.cricbuzz.com

[#] TLD server IP for com : 10.2.3.45

[#] Auth server IPL for cricbuzz.com : 10.12.86.142

[#] Server IP for www.cricbuzz.com : 83.78.55.170

Request from client : cric.cricbuzz.com

[#] TLD server IP for com : 10.2.3.45

[#] Auth server IPL for cricbuzz.com : 10.12.86.142

[#] Server IP for cric.cricbuzz.com : 37.68.23.143

Request from client : cric.cricbuzz.com

[#] TLD server IP for com : 10.2.3.45

[#] Auth server IPL for cricbuzz.com : 10.12.86.142

[#] Server IP for cric.cricbuzz.com : 37.69.23.143

## AUTHORIZATION

```
AUTHORITATIVE DNS SERVER PORT : 8043

DNS QUERY : www.cricbuzz.com
AUTHORITATIVE SERVER IP : 83.78.55.120

DNS QUERY : www.cricbuzz.com
AUTHORITATIVE SERVER IP : 83.78.55.135

DNS QUERY : www.cricbuzz.com
AUTHORITATIVE SERVER IP : 83.78.55.170

DNS QUERY : cric.cricbuzz.com
AUTHORITATIVE SERVER IP : 37.68.23.143

DNS QUERY : cric.cricbuzz.com
AUTHORITATIVE SERVER IP : 37.69.23.143
```

## TOP LEVEL DOMAIN

```
TOP LEVEL DOMAIN SERVER 8042

REQUEST FROM CLIENT : cricbuzz.com
AUTHORITATIVE SERVER IP : 10.12.86.142

REQUEST FROM CLIENT : cricbuzz.com
AUTHORITATIVE SERVER IP : 10.12.86.142

REQUEST FROM CLIENT : cricbuzz.com
AUTHORITATIVE SERVER IP : 10.12.86.142

REQUEST FROM CLIENT : cricbuzz.com
AUTHORITATIVE SERVER IP : 10.12.86.142

REQUEST FROM CLIENT : cricbuzz.com
AUTHORITATIVE SERVER IP : 10.12.86.142
```

## ROOT

```
ROOT DNS RESOLVER STARTED AT PORT : 8041

REQUEST FROM : com
TOP LEVEL DOMAIN IP : 10.2.3.45

REQUEST FROM : com
TOP LEVEL DOMAIN IP : 10.2.3.45

REQUEST FROM : com
TOP LEVEL DOMAIN IP : 10.2.3.45

REQUEST FROM : com
TOP LEVEL DOMAIN IP : 10.2.3.45

REQUEST FROM : com
TOP LEVEL DOMAIN IP : 10.2.3.45
```

## TEXT FILES

### ROOT - rootDNS.txt

```
[s2019103573@centos8-linux Tue Oct 19 07:00 PM spot]$ cat rootDNS.txt

edu 24.245.36.36
org 4.44.24.1
com 10.2.3.45
```

### TLP - tldDNS.txt

```
[s2019103573@centos8-linux Tue Oct 19 07:13 PM spot]$ cat tldDNS.txt

amazon.com 75.16.143.461
google.com 95.67.34.14
cricbuzz.com 10.12.86.142
```

### AUTHORIZATION - authDNS.txt

```
[s2019103573@centos8-linux Tue Oct 19 07:32 PM spot]$ cat authDNS.txt

mail.google.com 65.15.75.42 65.15.75.46 65.15.75.74 65.85.75.42
www.cricbuzz.com 83.78.55.120 83.78.55.135 83.78.55.170 83.78.55.199
cric.cricbuzz.com 37.68.23.143 37.69.23.143 37.68.23.276 37.68.23.893
```