

Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester V

16 September 2020

UCS1511 - Networks Laboratory

Exercise 6: Domain Name Server using UDP

Objective:

Simulate the concept of **Domain Name Server** using UDP.

Code:

DNS Table structure:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<sys/types.h>
4 #include<sys/socket.h>
5 #include<netinet/in.h>
6 #include<string.h>
```

```

7 #include<unistd.h>
8 #include<arpa/inet.h>
9
10 #define ADDR_LIMIT 5
11 #define DOMAIN_LIMIT 10
12
13 //Record of each domain-address pair
14 struct record{
15     char *domain;
16     char *address[ADDR_LIMIT];
17 };
18
19 typedef struct record Record;
20
21 void init(Record *r){
22     r->domain = (char*)calloc(100, sizeof(char));
23     for(int i =0;i<ADDR_LIMIT;i++)
24         r->address[i] = (char*)calloc(100, sizeof(char));
25 }
26
27 //Print DNS Table
28 void DNSTable(Record table[DOMAIN_LIMIT]){
29     printf(" ----- \n");
30     printf("|__Domain Name__|_____Address_____|\n");
31
32     for (int i = 0; i < DOMAIN_LIMIT; i++){
33         if (table[i].domain[0]){
34             printf("| %-15s | %-20s |\n", table[i].domain,
35 table[i].address[0]);
36
37             for (int j = 1; j < ADDR_LIMIT && table[i].
38 address[j][0]; j++)
39                 printf("| %-15s | %-20s |\n", "", table[i].
40 address[j]);
41             printf("| _____|_____
42 |\n");
43         }
44     }
45     printf("\n");
46 }
47
48 //Check if newly specified address is a valid address
49 int checkAddress(Record *table, char *address){
50
51     char* addr_copy = (char*)calloc(100, sizeof(char));

```

```

48     strcpy(addr_copy, address);
49     char *split;
50     int val;
51     split = strtok(addr_copy, ".");
52     //Check if all octets lie within 0 and 255.
53     while (split){
54         val = atoi(split);
55         if (val < 0 || val > 255){
56             printf("\nError: Invalid Address.\n");
57             return 0;
58         }
59         split = strtok(NULL, ".");
60     }
61
62     //Check if new address already exists in the table
63     for (int i = 0; i < DOMAIN_LIMIT; i++){
64         if (!table[i].domain[0])
65             continue;
66
67         for (int j = 0; j < ADDR_LIMIT && table[i].address[j
68 ] [0]; j++){
69             if (strcmp(address, table[i].address[j]) == 0){
70                 printf("\nError: IP address already exists.\n
71 ");
72                 return 0;
73             }
74         }
75     }
76
77     return 1;
78 }
79
80 //Create DNS-address pair in the table
81 int createRecord(Record table[DOMAIN_LIMIT], char *domain,
82 char *address){
83
84     int ix = -1;
85     int flag = 0;
86     //Check if entry exists already
87     int addr_valid = checkAddress(table, address);
88     if (!addr_valid)
89         return flag;
90
91     for (int i = 0; i < DOMAIN_LIMIT; i++){
92         if (strcmp(table[i].domain, domain) == 0){
93             for (int j = 0; j < DOMAIN_LIMIT; j++){

```

```

90             if (!table[i].address[j][0]){
91                 strcpy(table[i].address[j], address);
92                 flag = 1;
93                 break;
94             }
95             break;
96         }
97         if (!table[i].domain[0] && ix == -1)
98             ix = i;
99     }
100
101     // If record can be created
102     if (!flag){
103         strcpy(table[ix].domain, domain);
104         strcpy(table[ix].address[0], address);
105         flag = 1;
106     }
107
108     return flag;
109 }
110
111 char *getAddress(Record *table, char *domain){
112
113     char* addresses = (char*)calloc(ADDR_LIMIT*20, sizeof(
114         char));
115
116     for (int i = 0; i < DOMAIN_LIMIT; i++){
117         if (strcmp(table[i].domain, domain) == 0){
118             for (int j = 0; j < ADDR_LIMIT; j++) {
119                 strcat(addresses, table[i].address[j]);
120                 strcat(addresses, " ");
121             }
122             break;
123         }
124     }
125     return addresses;
126 }

```

Server:

```

1 #include "DNSTable.h"
2

```

```

3 int main(int argc, char **argv){
4
5     Record table[DOMAIN_LIMIT];
6     char *addresses = (char*)calloc(ADDR_LIMIT*2, sizeof(char
7 ));
8     for(int i =0; i<DOMAIN_LIMIT;i++){
9         init(&table[i]);
10    }
11
12    if (argc > 1){
13        perror("\nError: No arguments needed for server.\n");
14        exit(0);
15    }
16
17    //Server and Client addresses
18    struct sockaddr_in server_address, client_address;
19    //Buffer to handle messages
20    char buffer[1024];
21
22    char *domain = (char*)calloc(100, sizeof(char));
23    char *address = (char*)calloc(100, sizeof(char));
24
25    //Server socket file descriptor
26    int sockfd = socket(AF_INET, SOCK_DGRAM, 0); //domain =
27    IPv4, type = UDP, protocol = ip
28    if (sockfd < 0)
29        perror("\nError: Socket creation failed!\n");
30
31    //Filling server_address with null bytes
32    bzero(&server_address, sizeof(server_address));
33
34    server_address.sin_family = AF_INET; // Uses Internet
35    address family
36    server_address.sin_addr.s_addr = htonl(INADDR_ANY); //Use
37    any of the available addresses
38    server_address.sin_port = htons(7894); //Use port 7894
39
40    //Bind socket to the specified port
41    if ((bind(sockfd, (struct sockaddr *)&server_address,
42    sizeof(server_address)))< 0)
43        perror("\nError: Socket bind failed!\n");
44
45    int len = sizeof(client_address);
46
47    //Create record in table

```

```

43     createRecord(table, "google.com", "192.168.1.1");
44     createRecord(table, "yahoo.com", "194.12.34.12");
45     createRecord(table, "google.com", "17.10.23.123");
46
47     //Allow modification of table
48     char opt='n';
49     do{
50         DNSTable(table);
51
52         printf("\nDo you want to update table? y/n: ");scanf(
53 " %c", &opt);
54
55         if(opt == 'y' || opt == 'Y'){
56             printf("\nEnter domain: ");scanf(" %[^\\n]",
57 domain);
58             printf("\nEnter address: ");scanf(" %[^\\n]",
59 address);
60
61             int rval = createRecord(table, domain, address);
62             if(rval)
63                 printf("\nSuccessfully added entry!!\\n");
64         }
65     }while(opt == 'y' || opt == 'Y');
66
67     printf("\nDNS Server set up\\n");
68
69     while(1){
70         bzero(&buffer, sizeof(buffer));
71         recvfrom(sockfd, buffer, sizeof(buffer), MSG_WAITALL,
72 (struct sockaddr *)&client_address, &len);
73
74         strcpy(addresses, getAddress(table, buffer));
75         sendto(sockfd, addresses, sizeof(buffer), MSG_CONFIRM
76 , (struct sockaddr *)&client_address, len);
77     }
78     close(sockfd);
79 }

```

Client:

```

1 #include "DNSTable.h"
2

```

```

3 int main(int argc, char **argv){
4
5     if (argc < 2){
6         perror("\nError: IP address is to be passed as
argument.\n");
7         exit(0);
8     }
9
10    //Server addresses
11    struct sockaddr_in server_address;
12    //Buffer to handle messages
13    char buffer[1024];
14    //Query
15    Record *query = (Record*)malloc(sizeof(Record));
16
17    //Server socket file descriptor
18    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
19    if (sockfd < 0){
20        perror("\nError: Socket creation failed!\n");
21    }
22    //Filling server_address with null bytes
23    bzero(&server_address, sizeof(server_address));
24
25    server_address.sin_family = AF_INET; // Uses Internet
address family
26    server_address.sin_addr.s_addr = INADDR_ANY; //Use any of
the available addresses
27    server_address.sin_port = htons(7894); //Use port 7894
28
29    int len = sizeof(Record);
30    while(1){
31        init(query);
32
33        printf("\nEnter the domain name: "); scanf(" %[^\n]",
query->domain);
34
35        if (strcmp(query->domain, "end") == 0)
36            break;
37        //Send requested domain name
38        sendto(sockfd, query->domain, sizeof(buffer),
MSG_CONFIRM, (struct sockaddr *)&server_address, sizeof(
server_address));
39
40        bzero(&buffer, sizeof(buffer));

```

```

41         //Recieve IP address(es) of requested domain if it
         exists.
42         recvfrom(sockfd, buffer, sizeof(buffer), MSG_WAITALL,
         (struct sockaddr *)&server_address, &len);
43
44         char* split = strtok(buffer, " ");
45         if(split){
46             printf("\nThe IP Address of the requested domain
is: ");
47             while(split){
48                 printf("\n%s", split);
49                 split = strtok(NULL, " ");
50             }
51             printf("\n");
52         }
53         else{
54             printf("\nNo address in DNS.\n");
55         }
56     }
57
58     close(sockfd);
59 }

```

Output:

Server:

```

1  -----
2  |__Domain Name__|_____Address_____|
3  | google.com   | 192.168.1.1 |
4  |              | 17.10.23.123 |
5  |-----|-----|
6  | yahoo.com    | 194.12.34.12 |
7  |-----|-----|
8
9
10 Do you want to update table? y/n: y
11
12 Enter domain: google.com
13
14 Enter address: 255.254.253.252

```



```

15
16 Successfully added entry!!
17 -----
18 |__Domain Name__| |__Address__|
19 | google.com    | | 192.168.1.1   |
20 |               | | 17.10.23.123  |
21 |               | | 255.254.253.252|
22 |-----| |-----|
23 | yahoo.com     | | 194.12.34.12  |
24 |-----| |-----|
25
26
27 Do you want to update table? y/n: y
28
29 Enter domain: youtube.com
30
31 Enter address: 111.234.15.1
32
33 Successfully added entry!!
34 -----
35 |__Domain Name__| |__Address__|
36 | google.com    | | 192.168.1.1   |
37 |               | | 17.10.23.123  |
38 |               | | 255.254.253.252|
39 |-----| |-----|
40 | yahoo.com     | | 194.12.34.12  |
41 |-----| |-----|
42 | youtube.com   | | 111.234.15.1  |
43 |-----| |-----|
44
45
46 Do you want to update table? y/n: n
47
48 DNS Server set up

```

Client 1:

```

1 Enter the domain name: google.com
2
3 The IP Address of the requested domain is:
4 192.168.1.1
5 17.10.23.123

```

```
6 255.254.253.252
7
8 Enter the domain name: youtube.com
9
10 The IP Address of the requested domain is:
11 111.234.15.1
12
13 Enter the domain name: end
```

Client 2:

```
1 Enter the domain name: youtube.com
2
3 The IP Address of the requested domain is:
4 111.234.15.1
5
6 Enter the domain name: yahoo.com
7
8 The IP Address of the requested domain is:
9 194.12.34.12
10
11 Enter the domain name: end
```

Client 3:

```
1 The IP Address of the requested domain is:
2 194.12.34.12
3
4 Enter the domain name: google.com
5
6 The IP Address of the requested domain is:
7 192.168.1.1
8 17.10.23.123
9 255.254.253.252
10
11 Enter the domain name: end
```

Objective:

Simulate the concept of **Recursive Domain Name Server** using UDP.

Code:

Server:

```
1 #include "DNSTable.h"
2
3 int main(int argc, char **argv){
4
5     if (argc > 1){
6         perror("\nError: No arguments needed for server.\n");
7         exit(0);
8     }
9     //IP address(es) retrieved
10    char *addresses = (char*)calloc(ADDR_LIMIT*2, sizeof(char
11    ));
12    //Table at authoritative level
13    Record auth_table[DOMAIN_LIMIT];
14    for(int i =0; i<DOMAIN_LIMIT;i++){
15        init(&auth_table[i]);
16    }
17    //Table at local DNS server
18    Record local_table[DOMAIN_LIMIT];
19    for(int i =0; i<DOMAIN_LIMIT;i++){
20        init(&local_table[i]);
21    }
22
23    //Server and Client addresses
24    struct sockaddr_in server_address, client_address;
25    //Buffer to handle messages
26    char buffer[1024];
27
28    char *domain = (char*)calloc(100, sizeof(char));
29    char *address = (char*)calloc(100, sizeof(char));
30
31    //Server socket file descriptor
```

```

31     int sockfd = socket(AF_INET, SOCK_DGRAM, 0); //domain =
IPv4, type = UDP, protocol = ip
32     if (sockfd < 0)
33         perror("\nError: Socket creation failed!\n");
34
35     //Filling server_address with null bytes
36     bzero(&server_address, sizeof(server_address));
37
38     server_address.sin_family = AF_INET; // Uses Internet
address family
39     server_address.sin_addr.s_addr = htonl(INADDR_ANY); //Use
any of the available addresses
40     server_address.sin_port = htons(7894); //Use port 7894
41
42     //Bind socket to the specified port
43     if ((bind(sockfd, (struct sockaddr *)&server_address,
sizeof(server_address))) < 0)
44         perror("\nError: Socket bind failed!\n");
45
46     int len = sizeof(client_address);
47
48     //Create record in table
49     createRecord(auth_table, "google.com", "192.168.1.1");
50     createRecord(auth_table, "yahoo.com", "194.12.34.12");
51     createRecord(auth_table, "google.com", "17.10.23.123");
52
53     //Allow modification of table
54     char opt='n';
55     do{
56         DNSTable(auth_table);
57
58         printf("\nDo you want to update table? y/n: "); scanf(
" %c", &opt);
59
60         if(opt == 'y' || opt == 'Y'){
61             printf("\nEnter domain: "); scanf(" %[^\n]",
domain);
62             printf("\nEnter address: "); scanf(" %[^\n]",
address);
63
64             int rval = createRecord(auth_table, domain,
address);
65             if(rval)
66                 printf("\nSuccessfully added entry!!\n");
67         }

```

```

68     }while(opt == 'y' || opt == 'Y');
69
70     printf("\nDNS Server set up\n");
71
72     while(1){
73         printf("\n%50s\n", "-");
74         bzero(&buffer, sizeof(buffer));
75         recvfrom(sockfd, buffer, sizeof(buffer), MSG_WAITALL,
76                 (struct sockaddr *)&client_address, &len);
77
78         strcpy(addresses, getAddress(local_table, buffer));
79
80         char *copy = (char*)calloc(ADDR_LIMIT*20, sizeof(char
81     ));
82         strcpy(copy, addresses);
83
84         printf("\nChecking local DNS server...");
85         char* split = strtok(copy, " ");
86         if(split){
87             printf("Available in local DNS server. \n");
88             sendto(sockfd, addresses, sizeof(buffer),
89                 MSG_CONFIRM, (struct sockaddr *)&client_address, len);
90         }
91         else{
92             printf("\nNot found in local DNS server.\n");
93
94             printf("\nChecking root DNS server...");
95             printf("\nNot found in root DNS server. \n");
96
97             printf("\nChecking top level DNS server...");
98             printf("\nNot found in top level DNS server. \n");
99
100         ;
101
102         printf("\nChecking authoritative DNS server...");
103         strcpy(addresses, getAddress(auth_table, buffer))
104
105         ;
106
107         strcpy(copy, addresses);
108         split = strtok(copy, " ");
109         if(split){
110             while(split){
111                 int val = createRecord(local_table,
112                     buffer, split);
113                 if(val)

```

```

106         printf("\nSuccessfully added entry in
    local DNS server.\n");
107         split = strtok(NULL, " ");
108     }
109     sendto(sockfd, addresses, sizeof(buffer),
MSG_CONFIRM, (struct sockaddr *)&client_address, len);
110 }
111     else{
112         sendto(sockfd, addresses, sizeof(buffer),
MSG_CONFIRM, (struct sockaddr *)&client_address, len);
113     }
114 }
115
116
117 }
118     close(sockfd);
119 }

```

Output:

Server:

```

1  -----
2  |__Domain Name__|_____Address_____|
3  | google.com    | 192.168.1.1 |
4  |               | 17.10.23.123 |
5  | -----|-----|
6  | yahoo.com     | 194.12.34.12 |
7  | -----|-----|
8
9
10 Do you want to update table? y/n: y
11
12 Enter domain: youtube.com
13
14 Enter address: 255.254.253.252
15
16 Successfully added entry!!
17 -----
18 |__Domain Name__|_____Address_____|
19 | google.com    | 192.168.1.1 |

```

```

20 |           | 17.10.23.123 |
21 |-----|-----|
22 | yahoo.com | 194.12.34.12  |
23 |-----|-----|
24 | youtube.com | 255.254.253.252 |
25 |-----|-----|
26
27
28 Do you want to update table? y/n: n
29
30 DNS Server set up
31
32 -----
33
34 Checking local DNS server...
35 Not found in local DNS server.
36
37 Checking root DNS server...
38 Not found in root DNS server.
39
40 Checking top level DNS server...
41 Not found in top level DNS server.
42
43 Checking authoritative DNS server...
44 Successfully added entry in local DNS server.
45
46 -----
47
48 Checking local DNS server...
49 Not found in local DNS server.
50
51 Checking root DNS server...
52 Not found in root DNS server.
53
54 Checking top level DNS server...
55 Not found in top level DNS server.
56
57 Checking authoritative DNS server...
58 Successfully added entry in local DNS server.
59
60 -----
61
62 Checking local DNS server...
63 Not found in local DNS server.
64

```

```

65 Checking root DNS server...
66 Not found in root DNS server.
67
68 Checking top level DNS server...
69 Not found in top level DNS server.
70
71 Checking authoritative DNS server...
72 Successfully added entry in local DNS server.
73
74 -----
75
76 Checking local DNS server...Available in local DNS server.
77
78 -----
79
80 Checking local DNS server...Available in local DNS server.
81
82 -----
83
84 Checking local DNS server...Available in local DNS server.

```

Client 1:

```

1 Enter the domain name: yahoo.com
2
3 The IP Address of the requested domain is:
4 194.12.34.12
5
6 Enter the domain name: google.com
7
8 The IP Address of the requested domain is:
9 192.168.1.1
10 17.10.23.123
11
12 Enter the domain name: end

```

Client 2:


```
1 Enter the domain name: google.com
2
3 The IP Address of the requested domain is:
4 192.168.1.1
5 17.10.23.123
6
7 Enter the domain name: youtube.com
8
9 The IP Address of the requested domain is:
10 255.254.253.252
11
12 Enter the domain name: end
```

Client 3:

```
1 Enter the domain name: youtube.com
2
3 The IP Address of the requested domain is:
4 255.254.253.252
5
6 Enter the domain name: yahoo.com
7
8 The IP Address of the requested domain is:
9 194.12.34.12
10
11 Enter the domain name: end
```
