EX7 – DOMAIN NAME SERVER USING UDP

- S. Vishakan CSE – C 18 5001 196

Local DNS Server Program:

```
#include "DNS.h"

dns_table *local;

int cli_fd, root_fd, tld_fd, auth_fd;
struct sockaddr_in client_addr, local_addr, root_addr, tld_addr, auth_addr;

char *iterativeQuery(char *req_server);

int main(int argc, char **argv){

	int n, addrlen, flag;
	char req_server[100], *ip, req_ip[50];

	local = initTable(local, "Local");
	printTable(local);

	printf("\n\nDo you wish to alter the allocation table? (1 - YES, 0 - NO) -> ");
	scanf("%d", &flag);

	if(flag == 1){
		updateTable(local);
		printTable(local);
	}

	cli_fd = setUpConnection(&local_addr, CLI_PORT, 1, "client"); //bind here since
								server
	root_fd = setUpConnection(&root_addr, ROOT_PORT, 0, "root server"); //do not bind
								here
	tld_fd = setUpConnection(&tld_addr, TLD_PORT, 0, "top-level domain server"); //do
								not bind here
	auth_fd = setUpConnection(&auth_addr, AUTH_PORT, 0, "authoritative server"); //do
								not bind here

	printf("\nLocal DNS Server awaiting clients on port %d...\n", CLI_PORT);

	addrlen = sizeof(client_addr);
```

```
	while(1){
		bzero(req_server, sizeof(req_server));
		recvfrom(cli_fd, &req_server, sizeof(req_server), 0, (struct
		sockaddr*)&client_addr, &addrlen);

		printf("\nReceived a request for IP Address of %s from a client.\n",
		req_server);


		ip = iterativeQuery(req_server);

		if(ip == NULL){ //IP address does not exist
			strcpy(req_ip, empty);
			sendto(cli_fd, &empty, sizeof(empty), 0, (struct
			sockaddr*)&client_addr, addrlen);
		}
		else{
			//pointer -> char_array conversion is necessary since pointer only
			sends 8 bits of data
			strcpy(req_ip, ip);
			sendto(cli_fd, &req_ip, sizeof(req_ip), 0, (struct
			sockaddr*)&client_addr, addrlen);
		}

		printf("\nReplied with IP Address %s\n", req_ip);
	}

	return 0;
}

char *iterativeQuery(char *req_server){
	//Queries the DNS table according to the iterative resolution method.

	char *address, reply[50], request[100];
	int addrlen;

	strcpy(request, req_server);

	printf("\nSearching the %s DNS table...", local->name);
	address = fetchAddress(local, request);
	addrlen = sizeof(root_addr);

	if(address == NULL){
		printf("\nLocal table does not have an entry for %s, requesting root...",
		req_server);

		bzero(&reply, sizeof(reply));
		sendto(root_fd, &request, sizeof(request), 0, (struct sockaddr*)&root_addr,
		addrlen);
		recvfrom(root_fd, &reply, sizeof(reply), 0, (struct sockaddr*)&root_addr,
		&addrlen);
```

```
            printf("\nRoot replied with address to ping \"%s\" TLD server.",
            toUppercase(reply));

            bzero(&reply, sizeof(reply));
            sendto(tld_fd, &request, sizeof(request), 0, (struct sockaddr*)&tld_addr,
            addrlen);
            recvfrom(tld_fd, &reply, sizeof(reply), 0, (struct sockaddr*)&tld_addr,
            &addrlen);

            printf("\nTLD replied with address to ping %s's authoritative server.",
            toUppercase(reply));

            bzero(&reply, sizeof(reply));
            sendto(auth_fd, &request, sizeof(request), 0, (struct sockaddr*)&auth_addr,
            addrlen);
            recvfrom(auth_fd, &reply, sizeof(reply), 0, (struct sockaddr*)&auth_addr,
            &addrlen);



            if(strcmp(reply, "NULL") == 0){
                    printf("\nAuthoritative server did not respond with an IP address.");
                    return NULL;
            }

            else{
                    printf("\nAuthoritative server replied with IP address: %s", reply);
                    addRecord(local, req_server, reply);
                    printf("\nAdded record for Server: %s with IP address: %s in local
                    table.",        req_server, reply);
                    address = reply;
            }

        }

        return address;
}
```

Output:

```
vishakan@Legion: ~/Desktop/Semester V/Practical/Computer Networks/Ex07

File  Edit  View  Search  Terminal  Help
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ gcc Local.c -o l -w
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ ./l

        -------------------------------------------------------
                        Local
        -------------------------------------------------------
        Server Name                     IP Address

        www.google.com                  142.89.78.66
        www.yahoo.com                   10.2.45.67
        www.annauniv.edu                197.34.53.122
        -------------------------------------------------------



Do you wish to alter the allocation table? (1 - YES, 0 - NO) -> 1

Enter Server Name:      lms.ssn.edu.in

Enter IP Address:       293.23.2.2

IP Address 293.23.2.2 is invalid.

Enter Server Name:      lms.ssn.edu.in

Enter IP Address:       10.2.45.67

IP Address 10.2.45.67 is already allocated.

Enter Server Name:      lms.ssn.edu.in

Enter IP Address:       34.22.123.11

Do you wish to continue modifying the table? (1 - YES, 0 - NO) -> 0

        -------------------------------------------------------
                        Local
        -------------------------------------------------------
        Server Name                     IP Address

        www.google.com                  142.89.78.66
        www.yahoo.com                   10.2.45.67
        www.annauniv.edu                197.34.53.122
        lms.ssn.edu.in                  34.22.123.11
        -------------------------------------------------------


Setting up connection to client through port 7100 ...
Connection to client successfully established.

Setting up connection to root server through port 7300 ...
Setting up connection to top-level domain server through port 7400 ...
Setting up connection to authoritative server through port 7500 ...
Local DNS Server awaiting clients on port 7100...
```

```
vishakan@Legion: ~/Desktop/Semester V/Practical/Computer Networks/Ex07
File  Edit  View  Search  Terminal  Help
        -------------------------------------------------------
                        Local
        -------------------------------------------------------
        Server Name                     IP Address

        www.google.com                  142.89.78.66
        www.yahoo.com                   10.2.45.67
        www.annauniv.edu                197.34.53.122
        lms.ssn.edu.in                  34.22.123.11
        -------------------------------------------------------


Setting up connection to client through port 7100 ...
Connection to client successfully established.

Setting up connection to root server through port 7300 ...
Setting up connection to top-level domain server through port 7400 ...
Setting up connection to authoritative server through port 7500 ...
Local DNS Server awaiting clients on port 7100...

Received a request for IP Address of www.google.com from a client.

Searching the Local DNS table...
Replied with IP Address 142.89.78.66

Received a request for IP Address of lms.ssn.edu.in from a client.

Searching the Local DNS table...
Replied with IP Address 34.22.123.11

Received a request for IP Address of www.amazon.com from a client.

Searching the Local DNS table...
Local table does not have an entry for www.amazon.com, requesting root...
Root replied with address to ping "COM" TLD server.
TLD replied with address to ping AMAZON's authoritative server.
Authoritative server replied with IP address: 13.33.148.207
Added record for Server: www.amazon.com with IP address: 13.33.148.207 in local table.
Replied with IP Address 13.33.148.207

Received a request for IP Address of www.doesnotexist1234.com from a client.

Searching the Local DNS table...
Local table does not have an entry for www.doesnotexist1234.com, requesting root...
Root replied with address to ping "COM" TLD server.
TLD replied with address to ping DOESNOTEXIST1234's authoritative server.
Authoritative server did not respond with an IP address.
Replied with IP Address NULL

Received a request for IP Address of www.amazon.com from a client.

Searching the Local DNS table...
Replied with IP Address 13.33.148.207
```

Root DNS Server Program:

```
#include "DNS.h"

char *findTLD(char *req_server);

int main(int argc, char **argv){
	struct sockaddr_in local;
	int sockfd, n, addrlen;
	char req_server[100], reply[50], *tld_extn;

	sockfd = setUpConnection(&local, ROOT_PORT, 1, "local DNS server");

	addrlen = sizeof(local);
	printf("\nRoot server awaiting requests from local DNS server in port %d...\n",
	ROOT_PORT);

	while(1){
		bzero(&req_server, sizeof(req_server));
		recvfrom(sockfd, &req_server, sizeof(req_server), 0, (struct sockaddr*)&local,
		&addrlen);
		printf("\nReceived request from local DNS server for %s", req_server);

		tld_extn = findTLD(req_server);
		strcpy(reply, tld_extn);

		printf("\nReplied back with address to \"%s\" TLD.\n", toUppercase(tld_extn));
		sendto(sockfd, &reply, sizeof(reply), 0, (struct sockaddr*)&local, addrlen);

	}

	close(sockfd);
	return 0;
}

char *findTLD(char *req_server){
	//finds the appropriate TLD server for the given domain name

	char *server_copy, *split;

	server_copy = (char *)calloc(100, sizeof(char));

	strcpy(server_copy, req_server);
	split = strtok(server_copy, ".");
	split = strtok(NULL, ".");
	split = strtok(NULL, ".");

	return split;
}
```

Output:

```
vishakan@Legion: ~/Desktop/Semester V/Practical/Computer Networks/Ex07
File  Edit  View  Search  Terminal  Help
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ gcc Root.c -o r -w
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ ./r

Setting up connection to local DNS server through port 7300 ...
Connection to local DNS server successfully established.

Root server awaiting requests from local DNS server in port 7300...

Received request from local DNS server for www.amazon.com
Replied back with address to "COM" TLD.

Received request from local DNS server for www.doesnotexist1234.com
Replied back with address to "COM" TLD.
```

Top-Level Domain DNS Server Program:

```
#include "DNS.h"

char *findAuth(char *req_server);

int main(int argc, char **argv){
	struct sockaddr_in local;
	int sockfd, n, addrlen;
	char req_server[100], reply[50], *auth_extn;

	sockfd = setUpConnection(&local, TLD_PORT, 1, "local DNS server");

	addrlen = sizeof(local);
	printf("\nTLD server awaiting requests from local DNS server in port %d...\n",
	TLD_PORT);

	while(1){
		bzero(&req_server, sizeof(req_server));
		recvfrom(sockfd, &req_server, sizeof(req_server), 0, (struct sockaddr*)&local,
		&addrlen);
		printf("\nReceived request from local DNS server for %s", req_server);

		auth_extn = findAuth(req_server);
		strcpy(reply, auth_extn);

		printf("\nReplied back with address to %s's authoritative server.\n",
		toUppercase(auth_extn));
		sendto(sockfd, &reply, sizeof(reply), 0, (struct sockaddr*)&local, addrlen);

	}

	close(sockfd);
	return 0;
}

char *findAuth(char *req_server){
	//finds the appropriate authoritative server for the given domain name

	char *server_copy, *split;

	server_copy = (char *)calloc(100, sizeof(char));

	strcpy(server_copy, req_server);
	split = strtok(server_copy, ".");
	split = strtok(NULL, ".");

	return split;

}
```

Output:

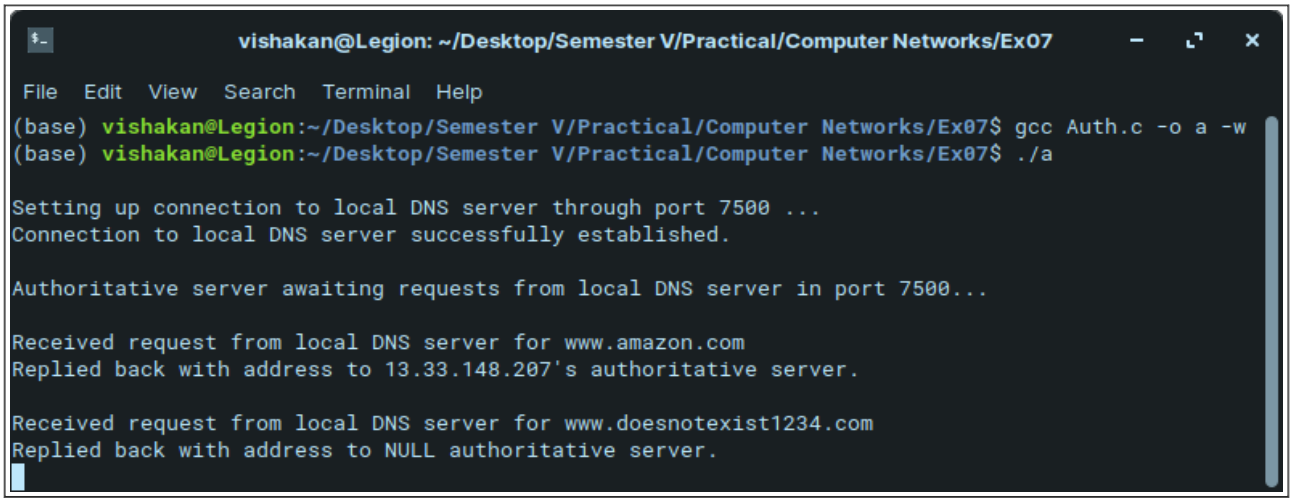
```
vishakan@Legion: ~/Desktop/Semester V/Practical/Computer Networks/Ex07
File  Edit  View  Search  Terminal  Help
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ gcc TLD.c -o t -w
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ ./t

Setting up connection to local DNS server through port 7400 ...
Connection to local DNS server successfully established.

TLD server awaiting requests from local DNS server in port 7400...

Received request from local DNS server for www.amazon.com
Replied back with address to AMAZON's authoritative server.

Received request from local DNS server for www.doesnotexist1234.com
Replied back with address to DOESNOTEXIST1234's authoritative server.
```

Authoritative DNS Server Program:

```
#include "DNS.h"

char *findIP(char *req_server);

int main(int argc, char **argv){
	struct sockaddr_in local;
	int sockfd, n, addrlen;
	char req_server[100], reply[50], *ip;

	sockfd = setUpConnection(&local, AUTH_PORT, 1, "local DNS server");

	addrlen = sizeof(local);
	printf("\nAuthoritative server awaiting requests from local DNS server in port %d...
	\n", AUTH_PORT);

	while(1){
		bzero(&req_server, sizeof(req_server));
		recvfrom(sockfd, &req_server, sizeof(req_server), 0, (struct sockaddr*)&local,
		&addrlen);
		printf("\nReceived request from local DNS server for %s", req_server);

		ip = findIP(req_server);

		if(ip == NULL){
			printf("\nReplied back with address to %s authoritative server.\n",
			empty);
			sendto(sockfd, &empty, sizeof(empty), 0, (struct sockaddr*)&local,
			addrlen);
		}

		else{
			strcpy(reply, ip);
			printf("\nReplied back with address to %s's authoritative server.\n", ip);
			sendto(sockfd, &reply, sizeof(reply), 0, (struct sockaddr*)&local,
			addrlen);
		}

	}

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

```c
char *findIP(char *req_server){
	//fetches the IP address for the given domain name
	//returns NULL if IP address does not exist

	struct hostent *he;
	char *ip;

	he = gethostbyname(req_server); //gets the host entry for the domain name

	if(he == NULL){ 		//if host entry does not exist for the domain
		return NULL;
	}

	ip = inet_ntoa(*((struct in_addr *)he->h_addr_list[0])); //get the IP address from host
								entry

	return ip;
}
```

Output:



Client Program:

```
#include "DNS.h"

int main(int argc, char **argv){
	struct sockaddr_in server;
	int sockfd, n, addrlen, flag, choice = 1;
	char req_server[100], req_ip[50];

	sockfd = setUpConnection(&server, CLI_PORT, 0, "local DNS server");
	addrlen = sizeof(server);

	while(choice){
		printf("\nEnter the Server's Name :\t");
		scanf("%s", req_server);
		sendto(sockfd, &req_server, sizeof(req_server), 0, (struct sockaddr*)&server,
		sizeof(server));

		recvfrom(sockfd, &req_ip, sizeof(req_ip), 0, (struct sockaddr*)&server,
		&addrlen);
		printf("The IP Address is :\t\t%s\n", req_ip);

		printf("\nContinue? (1 - YES, 0 - NO) -> ");
		scanf("%d", &choice);
	}
	close(sockfd);
	return 0;
}
```

Output:

```
vishakan@Legion: ~/Desktop/Semester V/Practical/Computer Networks/Ex07
File  Edit  View  Search  Terminal  Help
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ gcc Client.c -o c -w
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$ ./c

Setting up connection to local DNS server through port 7100 ...
Enter the Server's Name :	www.google.com
The IP Address is :		142.89.78.66

Continue? (1 - YES, 0 - NO) -> 1

Enter the Server's Name :	lms.ssn.edu.in
The IP Address is :		34.22.123.11

Continue? (1 - YES, 0 - NO) -> 1

Enter the Server's Name :	www.amazon.com
The IP Address is :		13.33.148.207

Continue? (1 - YES, 0 - NO) -> 1

Enter the Server's Name :	www.doesnotexist1234.com
The IP Address is :		NULL

Continue? (1 - YES, 0 - NO) -> 1

Enter the Server's Name :	www.amazon.com
The IP Address is :		13.33.148.207

Continue? (1 - YES, 0 - NO) -> 0
(base) vishakan@Legion:~/Desktop/Semester V/Practical/Computer Networks/Ex07$
```

Header File “DNS.h”:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define	CLI_PORT	7100
#define	LOCAL_PORT	7200
#define	ROOT_PORT	7300
#define	TLD_PORT	7400
#define	AUTH_PORT	7500

#define	SIZE		30

struct DNS_Table{ //struct for the DNS table
	char name[100];
	char server_list[SIZE][100];
	char ip_list[SIZE][50];
	int cur_size;
};

typedef struct DNS_Table dns_table;

char empty[5] = "NULL\0";

int setUpConnection(struct sockaddr_in *conn, int port, int is_bound, char *conn_name);
int checkIP(char *ip);
int addRecord(dns_table *table, char *server, char *ip);
void printTable(dns_table *table);
void updateTable(dns_table *table);
dns_table *initTable(dns_table *table, char *table_name);
char *fetchAddress(dns_table *table, char *req_server);
char *toUppercase(char *str);
```

```
int setUpConnection(struct sockaddr_in *conn, int port, int is_bound, char *conn_name){
	//sets up the basic socket connection and binds it to a port if specified, and returns
	socket file descriptor

	int sockfd;

	printf("\nSetting up connection to %s through port %d ...", conn_name, port);
	sockfd = socket(AF_INET, SOCK_DGRAM, 0);

	if(sockfd < 0){
		perror("Error in creating socket.\n");
	}

	bzero(conn, 16);
	conn->sin_family = AF_INET;
	conn->sin_addr.s_addr = INADDR_ANY;
	conn->sin_port = htons(port);

	if(is_bound){
		if(bind(sockfd, (struct sockaddr *)conn, 16) < 0){
			perror("Error in binding.\n");
		}
		else{
			printf("\nConnection to %s successfully established.\n", conn_name);
		}
	}

	return sockfd;
}

int checkIP(char *ip){
	//Checks for the validity of a given IP address

	int valid = 1, byte;
	char *ip_copy, *split;

	ip_copy = (char *)calloc(50, sizeof(char));
	strcpy(ip_copy, ip);
	split = strtok(ip_copy, ".");

	while(split){ //split pointer points to each "byte" iteratively
		byte = atoi(split);
		if(byte < 0 || byte > 255){
			return 0;
		}

		split = strtok(NULL, ".");
	}

	return 1;
}
```

```c
int addRecord(dns_table *table, char *server, char *ip){
	//Add a new record into a specific DNS table

	int valid;

	if(table->cur_size == SIZE - 1){ //if table is full, replace first record.
		strcpy(table->server_list[0], server);
		strcpy(table->ip_list[0], ip);

		return table->cur_size;
	}

	valid = checkIP(ip);

	if(valid){
		strcpy(table->server_list[table->cur_size], server);
		strcpy(table->ip_list[table->cur_size], ip);
		table->cur_size++;
	}

	else{
		printf("\tIP Address %s is invalid.\n", ip);

	}

	return table->cur_size;
}

void printTable(dns_table *table){
	//Print the current contents of a given table

	int i = 0;

	printf("\n\t-------------------------------------------------------");
	printf("\n\t\t\t%-30s", table->name);
	printf("\n\t-------------------------------------------------------");
	printf("\n\t%-25s\t%s\n", "Server Name", "IP Address");

	for(i = 0; i < table->cur_size; i++){
		printf("\n\t%-25s\t%s", table->server_list[i], table->ip_list[i]);
	}

	printf("\n\t-------------------------------------------------------\n\n");
}
```

```c
void updateTable(dns_table *table){
	//Update a given DNS table

	char serv[100], ip[50];
	int i = 0, exists = 0, choice = 1, valid;

	while(choice){
		printf("\nEnter Server Name:\t");
		scanf("%s", serv);
		printf("\nEnter IP Address:\t");
		scanf("%s", ip);
		valid = checkIP(ip);

		if(!valid){
			printf("\nIP Address %s is invalid.\n", ip);
			continue;
		}

		exists = 0;
		for(i = 0; i < table->cur_size; i++){
			if(strcmp(ip, table->ip_list[i]) == 0){
				exists = 1;
				printf("\nIP Address %s is already allocated.\n", ip);
				break;
			}
		}
		if(exists == 0){
			strcpy(table->ip_list[i], ip);
			strcpy(table->server_list[i], serv);
			table->cur_size++;

			printf("\nDo you wish to continue modifying the table? (1 - YES, 0 - NO)
			-> ");
			scanf("%d", &choice);
		}
	}
}

dns_table *initTable(dns_table *table, char *table_name){
	//Initialize the local, root and auth tables with some prefixed records

	table = (dns_table *)malloc(sizeof(dns_table));
	table->cur_size = 0;
	strcpy(table->name, table_name);

	addRecord(table, "www.google.com", "142.89.78.66");
	addRecord(table, "www.yahoo.com", "10.2.45.67");
	addRecord(table, "www.annauniv.edu", "197.34.53.122");

	return table;
}
```

```c
char *fetchAddress(dns_table *table, char *req_server){
	//Fetch the address of a given domain name from a given DNS table

	int i = 0;

	for(i = 0; i < table->cur_size; i++){
		if(strcmp(table->server_list[i], req_server) == 0){
			return table->ip_list[i];	//found
		}
	}

	return NULL;					//not found
}

char *toUppercase(char *str){
	//converts a given string to uppercase
	char *upper;
	int i = 0;

	upper = (char *)malloc(sizeof(str));

	for (i = 0; str[i] != '\0'; i++) {
		if(str[i] >= 'a' && str[i] <= 'z') {
			upper[i] = str[i] - 32;
		}

		else{
			upper[i] = str[i];
		}
	}

	upper[i] = '\0';

	return upper;
}
```