

Design and Implementation of A Simple DNS System

Course: Internet Applications
Name: LIU Toing (2017212783)
YAN Yangtian (2017212787)
Date: June 2, 2020

1. Overview

This project implements a DNS system with one DNS client and several DNS servers (includes one root server and one local server). The client can make query on the local server, and the local server can do iterative queries to the root server and the other servers upon the client's request. The whole system is designed to be run on Linux.

2. Requirements Analysis

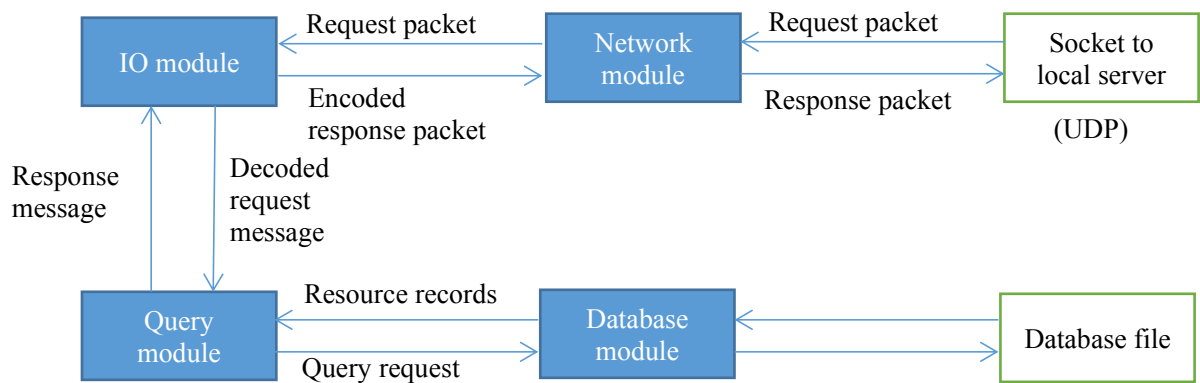
- The project is required to be implemented in C and run on Linux.
- A DNS client should be designed to do DNS query to the local DNS server over TCP.
- The local DNS server should be designed to accept the query from the client and do iterative queries to the other DNS servers over UDP if the requested name does not found in the local server's cache or the cache have expired.
- The root DNS server and the other four servers should search from their databases to find the queried names. The address of the next level DNS server will be included in the response.
- All the DNS traffic packets should be strictly in DNS format and can be decoded by Wireshark.
- The program should be able to handle errors during the execution.

3. Preliminary Design

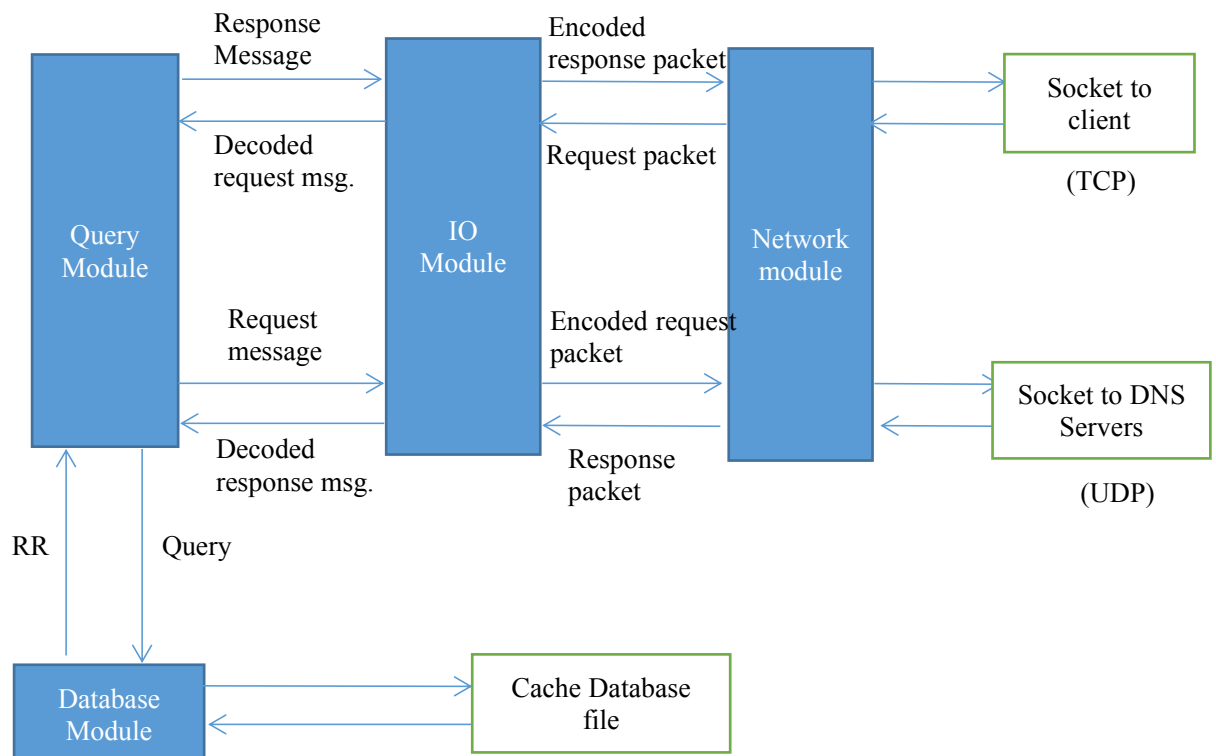
- The client can be broken into these function modules:
 - **Network module:** This module send the DNS query packet through TCP socket to the local DNS server and receive response from it.
 - **IO module:** This module read/write data from/to the buffer which is received from/sent to the DNS server. It can parse DNS packet data or serialize the data into the buffer.
 - **Query module:** This module constructs the request packet which is to be serialized by the IO module.
- The DNS server can be broken into these function modules:
 - **Network module:** This module handles the query requests and send response after querying the database and doing iterative queries (for local server).
 - **Query module:** This module handles the queries from the decoded request packet and do queries to the local database or other servers (for local server) and construct the response which is going to be serialized.
 - **IO module:** This module read/write data from/to the buffer which is received from/sent to the DNS server. It can parse DNS packet data or serialize the data into the buffer.
 - **Database module:** This module handles database query and read from/ write to local cache (for local server). The underlying database system is SQLite3.
- Project structure and development:

Since the project should be designed to be run on Linux and be written in C, we use the IDE CLion (by JetBrains) on Deepin OS (64bit Linux based operating system) to develop this project. CLion is a powerful IDE for C/C++ and it supports building and debugging C/C++ program on Windows, Mac OS and Linux. The projects in CLion is based on CMake, which is a build tool for C/C++. We can also use the terminal to build this project.

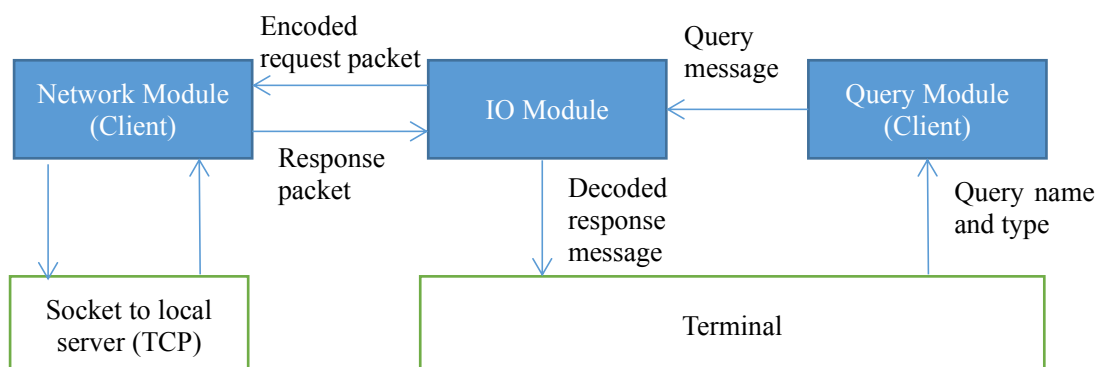
- The module relationship diagram of the non-local servers:



- The module relationship diagram of the local server:



- The module relationship diagram for the client:



- The DNS packet structure is shown as following code:

```
typedef struct {
    dns_header_t header;
    dns_query_t *queries;
    dns_rr_t *answers;
    dns_rr_t *authorities;
    dns_rr_t *additional;
} dns_packet_t;
```

The DNS packet header structure is shown in the following code:

```
typedef struct {
    uint16 id;

    uint8 rd:1;
    uint8 tc:1;
    uint8 aa:1;
    uint8 opcode:4;

    uint8 qr:1;
    uint8 rcode:4;
    uint8 z:3;
    uint8 ra:1;

    uint16 question_count;
    uint16 answer_count;
    uint16 authority_count;
    uint16 additional_count;
} dns_header_t;
```

The fields “rd” to the field “ra” are flags in the DNS headers. They are put in specific order to make sure that the flags can be resolved correctly.

The query type and the RR type is defined as follows, note that they are defined as nodes of linked lists:

```
typedef struct dns_query {
    ptr_t name;
    uint16 type;
    uint16 class;

    struct dns_query *next;
} dns_query_t;

typedef struct dns_rr {
    ptr_t name;
    uint16 type;
    uint16 class;
    uint32 ttl;
    uint16 length;
    ptr_t data;

    struct dns_rr *next;
} dns_rr_t;
```

In the codes above, there are definitions like “ptr_t”, “uint8”, these definitions are made for some basic types in order to shorten the code:

```
typedef unsigned char *ptr_t;
typedef unsigned char bool;
typedef unsigned char uint8;
typedef unsigned short uint16;
typedef unsigned int uint32;
```

Also, in order to make sure the IO functions work correctly, we defined a buffer type with limited capacity and a position. This can make sure that the read/write of the buffer will not exceed the boundary. The definition of the buffer type is:

```

struct dns_buffer {
    ptr_t ptr;
    uint32 capacity;
    uint32 pos;

    known_name_t *known_names;
};
typedef struct dns_buffer *buffer_t;

```

Inside the definition of the DNS buffer structure, there is a definition of field of type “known_name_t”, this is a structure used to store the known names and their locations, which is used to process the pointers of name in the DNS packet. When reading names from the buffer, if a name is read, it will be stored as a known name, and when a pointer of name (begin with 11 in binary) detected, the known names will be searched for the name at the specific position. This also works when writing to the buffer, when writing names to the buffer, the known names will be searched, if found the same name, its position will be written to the buffer instead of the raw string. The definition of the “known_name_t” is shown below (note that the structure is defined as a linked list node):

```

typedef struct known_name {
    ptr_t name;
    uint16 pos;
    struct known_name *next;
} known_name_t;

```

All the data used in the system, including RRs of the non-local DNS servers and the caches of the local server, is stored in database. The database is based on SQLite3. The source code of the SQLite3 library is added to the source code folder of the project. The file names of the source code of SQLite3 is “sqlite3.c” and “sqlite3.h”. The data structure in the database is like:

	id	name	ttl	class	type	data
1	1	www.baidu.com	60	1	5	www.a.shifen.com
2	2	www.a.shifen.com	60	1	1	14.215.177.30
3	3	www.a.shifen.com	60	1	1	14.215.177.39
4	4	tieba.baidu.com	60	1	5	post.n.shifen.com
5	5	post.n.shifen.com	60	1	1	14.215.177.221
6	6	code.org	60	1	1	99.84.57.215
7	7	studio.code.org	60	1	1	13.227.51.203

And the cache of local server is like:

	id	name	ttl	class	type	data	timestamp
1	1	www.baidu.com	60	1	5	www.a.shifen.com	1591600192
2	2	www.a.shifen.com	60	1	1	14.215.177.30	1591600192
3	3	www.a.shifen.com	60	1	1	14.215.177.39	1591600192
4	4	bupt.edu.cn	60	1	15	0,mx.bupt.edu.cn	1591600365
5	5	mx.bupt.edu.cn	60	1	1	183.3.235.87	1591600365
6	6	bupt.edu.cn	60	1	15	0,mx.bupt.edu.cn	1591600514
7	7	mx.bupt.edu.cn	60	1	1	183.3.235.87	1591600514
8	8	4.0.0.127.in-addr.arpa	60	1	12	s2.local	1591600587
9	9	www.baidu.com	60	1	5	www.a.shifen.com	1591601951

Note that the classes and types of the DNS records are stored as integer. And for caches, since all caches have a “time to live”, we should add a time stamp field indicate the time when the cache is added.

The IP addresses in the RR data are stored as human-readable strings, while in the DNS packets they are stored as 32-bit unsigned integers. This should be converted during the execution of the system. In the design of the system, this conversion is taken place in the IO module.

For MX records, there is a preference field, which should be the first byte of the RR data in the DNS packets. In the database, the preference and the name is sperated by comma (“,”). In the database, if only the name is presented, the preference will be set to 0.

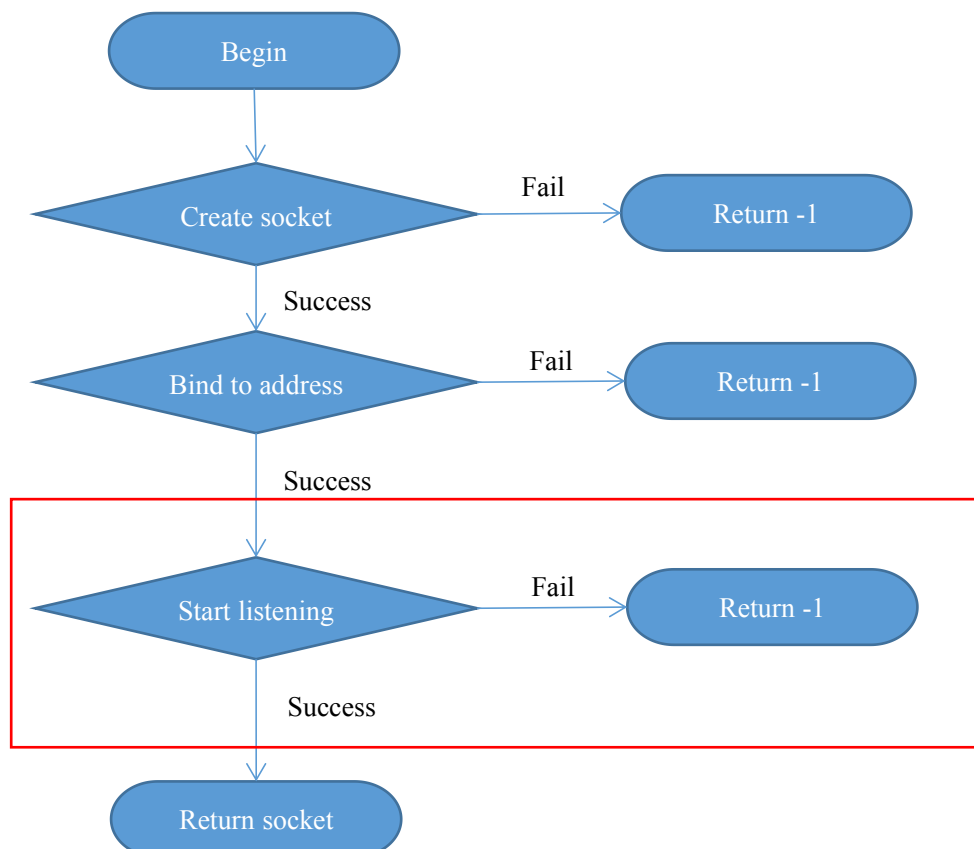
4. Detailed Design

4.1 The network module

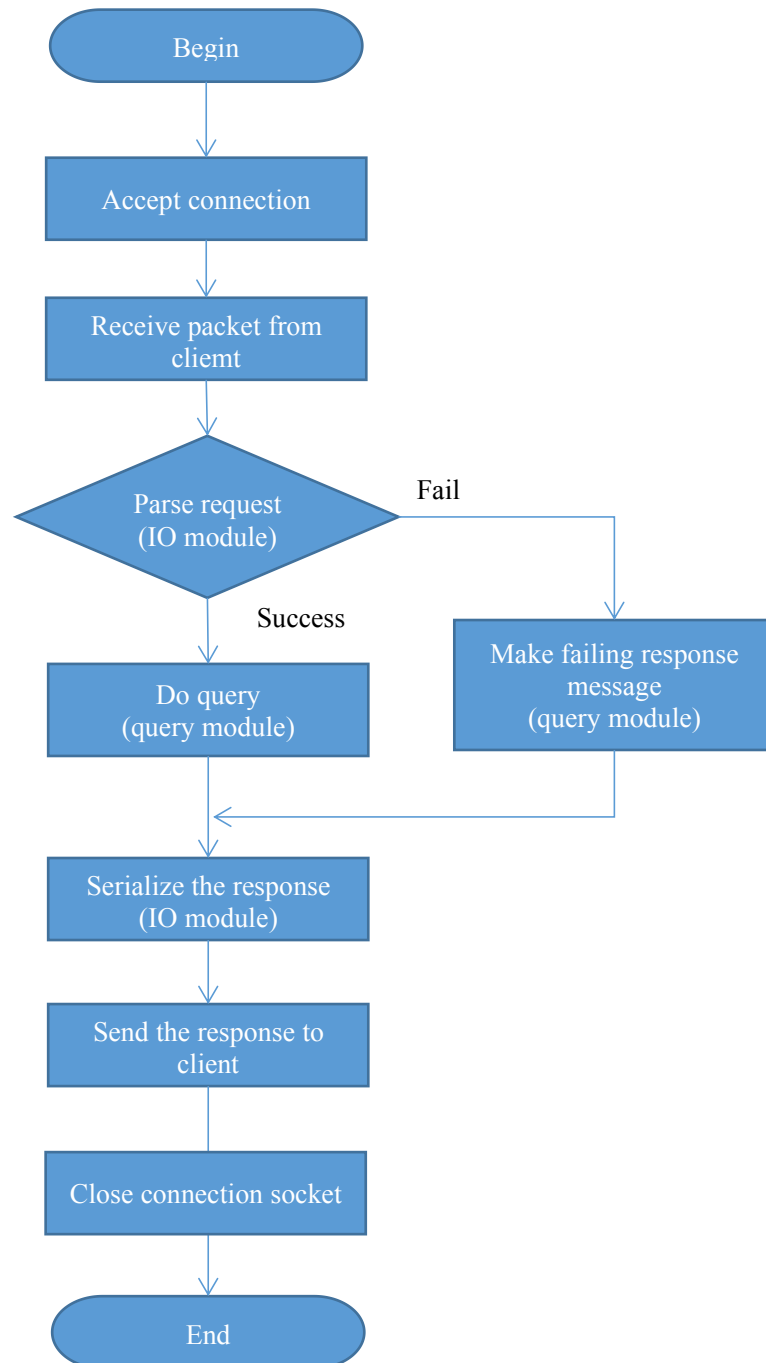
The network module includes the following functions.

```
int DNS_network_init_server_socket_udp(const char *address);  
int DNS_network_init_server_socket_tcp(const char *address);  
void DNS_network_handle_query_udp(int sock);  
void DNS_network_handle_query_tcp(int sock);  
dns_packet_t *DNS_network_send_query_udp(const char* address, char*  
name, int type);  
dns_packet_t *DNS_network_send_query_tcp(const char* address, char*  
name, int type);
```

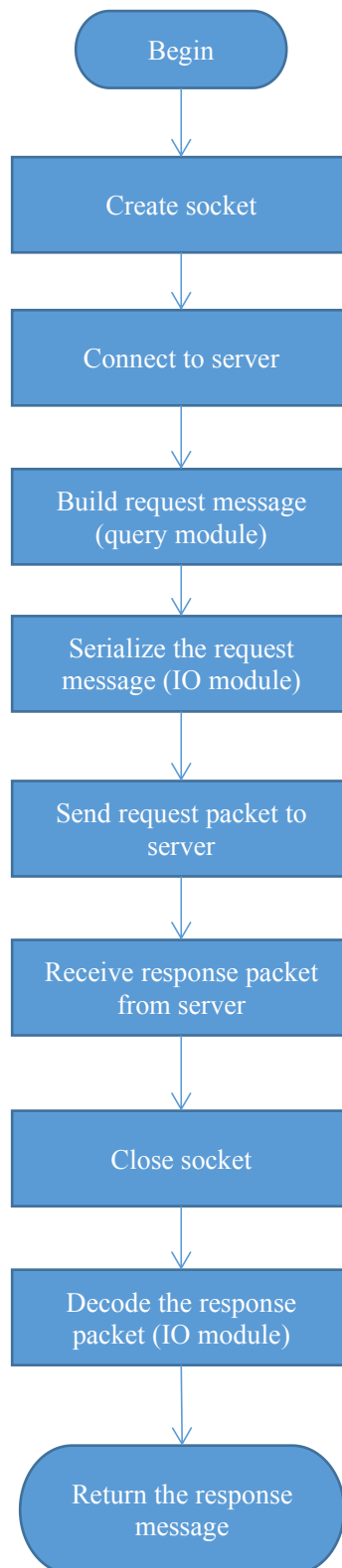
The “DNS_network_init_server_socket_udp” and “DNS_network_init_server_socket_tcp” functions are used for setting up sockets and listen on the specific address (the listen port is always 53, which is the port used for the DNS protocol) with UDP/TCP. And their overall flow chart is (The part within the red rectangle is only for TCP):



For “DNS_network_handle_query_udp” and “DNS_network_handle_query_tcp”, their functions are to accept requests from the client, do the query and then send the response back to the client. In the server code, after creating the socket, these functions will be called repeatedly. Their overall flowchart is as follows (all the procedure with possibility of failure is checked for success, if one procedure fails, the function will return immediately, the “accept connection” and “close connection socket” procedure is only used for TCP):



For the functions “DNS_network_send_query_udp” and “DNS_network_send_query_tcp”, they are used to send the DNS query to the specific server and receive response from them. Their flowcharts are shown as follows (note that the “connect to server” is only used for TCP, if any of these procedure fails, the function will return NULL):



4.2 The IO module

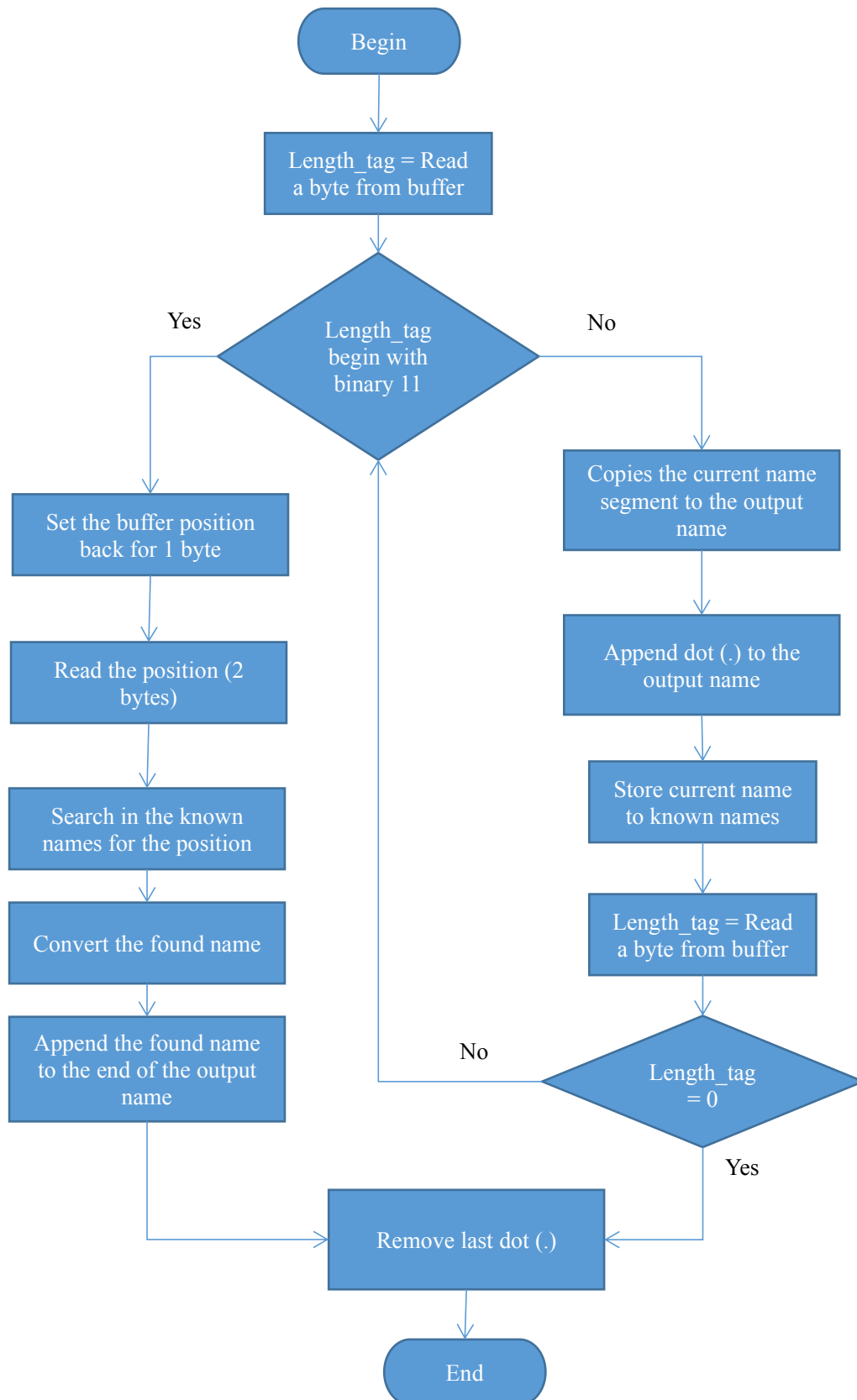
The design of the IO module is aiming to improve the flexibility and reliability of the I/O operations like parsing and serializing the DNS messages into DNS packets and parsing DNS packets to DNS messages. The module contains the following functions:

```
buffer_t DNS_buffer_create(int capacity);
buffer_t DNS_buffer_from_ptr(ptr_t ptr, int capacity);
void DNS_buffer_free(buffer_t buffer);
dns_query_t *DNS_query_create();
dns_rr_t *DNS_RR_create();
dns_query_t *DNS_query_copy(dns_query_t *other);
dns_rr_t *DNS_RR_copy(dns_rr_t *other);
void DNS_packet_append_query(dns_packet_t *packet, dns_query_t *query,
    bool increase_count);
void DNS_packet_append_answer(dns_packet_t *packet, dns_rr_t *rr,
    bool increase_count);
void DNS_packet_append_authority(dns_packet_t *packet,
    dns_rr_t *rr, bool increase_count);
void DNS_packet_append_additional(dns_packet_t *packet,
    dns_rr_t *rr, bool increase_count);
bool DNS_buffer_read_u8(buffer_t buffer, uint8 *v);
bool DNS_buffer_write_u8(buffer_t buffer, uint8 v);
bool DNS_buffer_read_u16(buffer_t buffer, uint16 *v);
bool DNS_buffer_write_u16(buffer_t buffer, uint16 v);
bool DNS_buffer_read_u32(buffer_t buffer, uint32 *v);
bool DNS_buffer_write_u32(buffer_t buffer, uint32 v);
bool DNS_buffer_read_DNS_header(buffer_t buffer,
    dns_header_t *v);
bool DNS_buffer_write_DNS_header(buffer_t buffer,
    dns_header_t v);
bool DNS_buffer_read_DNS_name(buffer_t buffer, ptr_t name);
bool DNS_buffer_write_DNS_name(buffer_t buffer, ptr_t name);
bool DNS_buffer_read_RR(buffer_t buffer, dns_rr_t *v);
bool DNS_buffer_write_RR(buffer_t buffer, dns_rr_t v);
bool DNS_buffer_read_query(buffer_t buffer, dns_query_t *v);
bool DNS_buffer_write_query(buffer_t buffer, dns_query_t v);
bool DNS_buffer_read_packet(buffer_t buffer, dns_packet_t *v);
bool DNS_buffer_write_packet(buffer_t buffer, dns_packet_t v);
```

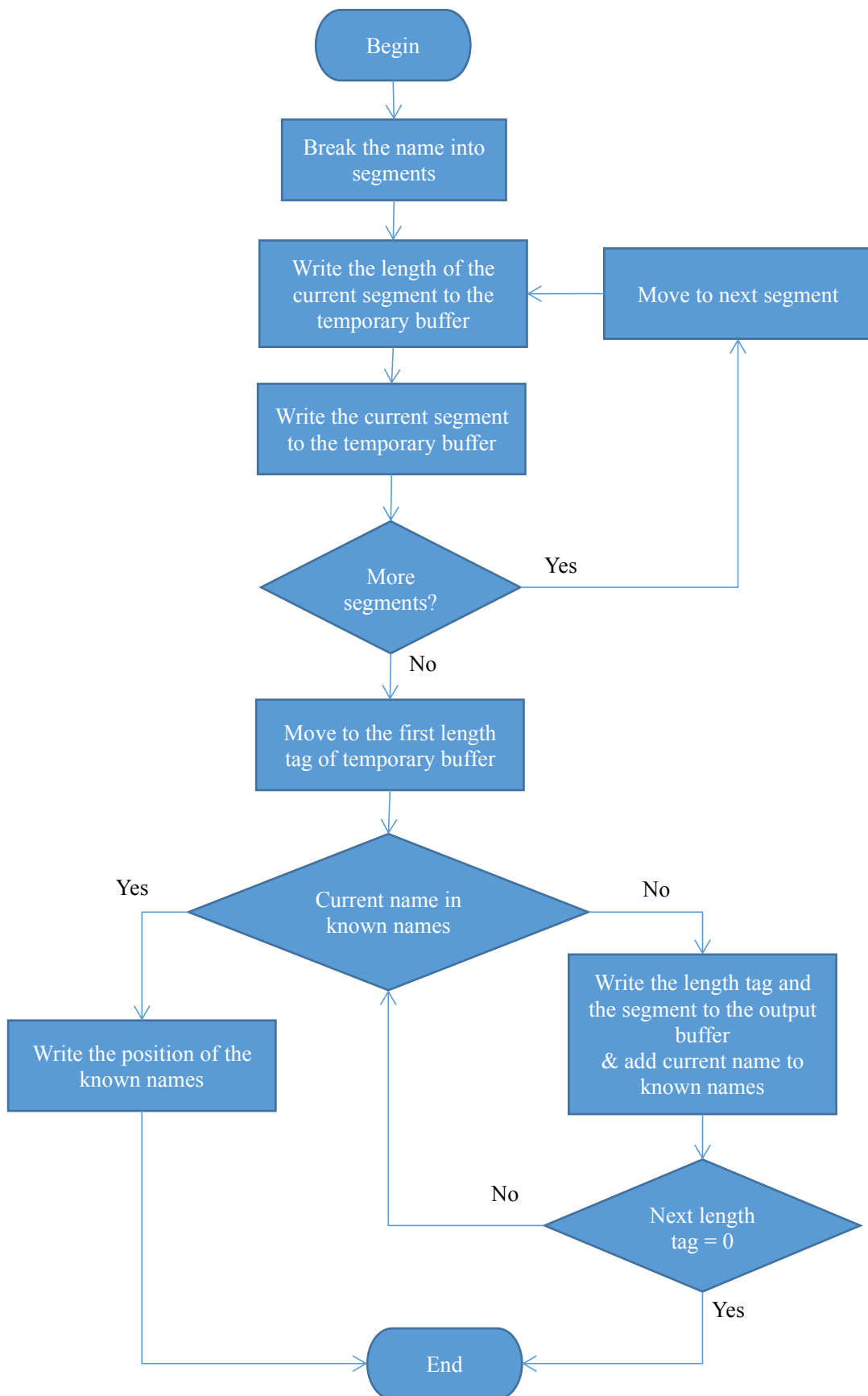
The design of the module is complex, but each of these functions below and including the “DNS_buffer_read_u8” are used to read or write a data unit (can be an unsigned integer, or some kinds of structures used for the DNS protocol). The last two functions is used to read/write the whole packet from/to the buffer. And the four functions “DNS_packet_append_query”, “DNS_packet_append_answer”, “DNS_packet_append_authority” and “DNS_packet_append_additional” are used to append queries or resource records to different sections of the existing DNS packet structure since in these structure, the RRs and queries are stored as linked lists. And we will only present the flowchart for reading/writing the DNS names.

Note that in the DNS packets, the IP addresses for records in type A is stored as unsigned integers instead of the text-like IP address, so we should convert the addresses when reading/writing the RRs. The conversion is done in the “DNS_buffer_read_RR” function and “DNS_buffer_write_RR” function with the “inet_addr” and “inet_ntoa” functions.

When writing or reading the DNS names, the name should be converted between the format used in the DNS packet (like \003www\004bupt\003edu\002\cn\000) and the common format (like www.bupt.edu.cn). And when a name is already existed in the current packet, the second occurrence will be a indicator of the position in the packet of the first occurrence of the name. The flowchart of the “DNS_buffer_read_DNS_name” function is:



The flowchart of the function “DNS_buffer_write_DNS_name” is:



4.3 The Query module

The query module is the module that handles the DNS queries, it takes the request message as input and will return the response message after the query is done. It can also build request messages. The functions defined in the query module is as follows:

```
dns_packet_t DNS_query_create_request(char* name, int type);
dns_packet_t DNS_query_create_fail_response(int rcode);
void DNS_query_set_table_name(const char* name);
dns_packet_t DNS_query_create_response(dns_packet_t request);
dns_packet_t DNS_query_create_response_local(dns_packet_t
request);
```

The “DNS_query_set_table_name” function is used to set the current database table name which will be queried by the “DNS_query_create_response” function. “DNS_query_create_fail_response” can create a empty DNS message which indicates an error. And “DNS_query_create_request” is used to generate the DNS request message according to the query name and type.

The algorithm of the “DNS_query_create_response” is too complex to be shown in flow charts, here goes the Pseudocode:

```
CNAME_pending = list of RR
ADD_pending = list of RR
data = DNS_database_get_record(name, [type, CNAME])
For item in data
    -- If we find RR of type CNAME but we are not looking for CNAME
    records, we should fetch the corresponding RR for that CNAME RR later
    If item.type = CNAME and type != CNAME
        CNAME_pending += copy of item
    -- The IP address of the MX server should be also presented
    If item.type = MX
        ADD_pending += copy of item
    Response.Answers += copy of item

-- Resolve the CNAME records
For item in CNAME_pending
    Data2 = DNS_database_get_record(item.data, [type, CNAME])
    For item2 in Data2
        -- Sometimes we found recursive CNAMEs, in this case we should
        add it to the list so it will be resolved later
        If item2.type = CNAME
            CNAME_pending += copy of item2
        If item2.type = MX
            ADD_pending += copy of item2
        Response.Answers += copy of item2

-- Find the addresses of the authoritative name servers
Name1 = name
While Name1 != null
    Data2 = DNS_database_get_record(Name1, NS)
    For item2 in Data2
        ADD_pending += copy of item2
        Response.Authorities += copy of item2
    Name1 = Name1 starting from next '.'

For item in ADD_pending
    Data2 = DNS_database_get_record(item.data, A)
    For item2 in Data2
        Response.Additionals += copy of item2
```

The Pseudocode of the function “DNS_query_create_response_local” is shown below, this function is used by the local DNS server to query the cache and making iterative queries.

```
-- Attempt to get cache from the database, the cache will also include
CNAME RRs
Cache = DNS_database_get_cache(name, type)
-- The cache is not empty
If not empty(Cache)
    CNAME_pending = list of RR
    MX_pending = list of RR
    For item in Cache
        If item.type = CNAME and type != CNAME
            CNAME_pending += copy of item
        If item.type = MX
            MX_pending += copy of item
        Response.Answers += copy of item

    For item in CNAME_pending
        Cache2 = DNS_database_get_cache(item.data, type)
        For item2 in Cache2
            If item2.type = CNAME
                CNAME_pending += copy of item2
            If item2.type = MX
                MX_pending += copy of item2
            Response.Answers += copy of item2

    For item in MX_pending
        Cache2 = DNS_database_get_cache(item.data, A)
        For item2 in Cache2
            Response.Additionals += copy of item2
-- Nothing found in cache, start iterative query
Else
    NS_pending = list of RR
    -- Add the root NS to the list of name servers
    NS_pending += (root server)
    For ns in NS_pending
        -- Query each NS in the list, recursive NSes will be added to
        the list
        NS_res = DNS_network_send_query_udp(ns.data, name, type)
        For item in NS_res.Answers
            Response.Answers += copy of item
            DNS_database_put_cache(item)
            If item.type = MX
                For item2 in NS_res.Additionals
                    If item2.name = item.value and item2.type = A
                        Response.Additionals += copy of item2
                        DNS_database_put_cache(item2)

        -- For each NS records, find the additional record containing
        its IP address
        For item in NS_res.Authorities
            For item2 in NS_res.Additionals
                If item2.name = item.value and item2.type = A
                    NS_pending += copy of item2
```

4.4 The Database module

The database module is designed to handle the database queries by both the non-local servers and the local server (for caches), the local server can also write cache data to the database.

The public functions defined in the database module are:

```
dns_rr_t *DNS_database_get_record(const char* table_name, char*
                                name, int type, int class, bool include_cname);
dns_rr_t * DNS_database_get_cache(char* name, int type,
                                int class);
bool DNS_database_put_cache(dns_rr_t rr);
```

Note that since the database is used by all the DNS servers at the same time, in each of the above functions, the database is opened when the functions are called and closed before the functions returns. And the returned collection of RRs by the first two functions are in linked list. In the function “DNS_database_get_record”, the parameter “include_cname” is used to determine whether to include CNAME records or just include the record of the type requested, while the function “DNS_database_put_cache” always include the CNAME records.

The underlying database engine is SQLite3. And the SQL statement used by each of the functions is shown below:

DNS_database_get_record:

```
SELECT * FROM %table_name WHERE name = '%name' and (type = %type)
and class = %class;
```

DNS_database_get_cache:

```
SELECT * FROM cache WHERE name = '%name' and (type = %type or type
= 5) and class = %class and timestamp + ttl > %time;
```

The last part “timestamp + ttl > %time” is used to make sure the result caches is not expired.

The “type = 5” part indicates that CNAME records will be included.

DNS_database_put_cache:

```
INSERT INTO cache VALUES (NULL, '%name', %ttl, %type, %class,
'%data', %time);
```

The first value is NULL, this is because the id field in the database is a integer primary key, its value will be automatically added.

When initializing the database, if the database file cannot be found, a new database will be created and some test data for the system will be write into it. The statements used to create the tables is as follows.

```
CREATE TABLE root (id INTEGER PRIMARY KEY, name TEXT, ttl INTEGER,
class INTEGER, type INTEGER, data TEXT);
CREATE TABLE s1 (id INTEGER PRIMARY KEY, name TEXT, ttl INTEGER,
class INTEGER, type INTEGER, data TEXT);
CREATE TABLE s2 (id INTEGER PRIMARY KEY, name TEXT, ttl INTEGER,
class INTEGER, type INTEGER, data TEXT);
CREATE TABLE s3 (id INTEGER PRIMARY KEY, name TEXT, ttl INTEGER,
class INTEGER, type INTEGER, data TEXT);
CREATE TABLE s4 (id INTEGER PRIMARY KEY, name TEXT, ttl INTEGER,
class INTEGER, type INTEGER, data TEXT);
CREATE TABLE cache (id INTEGER PRIMARY KEY, name TEXT, ttl INTEGER,
class INTEGER, type INTEGER, data TEXT,
timestamp INTEGER);
```

5. Compile & Execution

This project contains several different modules and each of them are implemented in different source code files, so it could be complicated to compile it directly with GCC. This project is managed by CMake. To compile the code, we should install CMake first, install CMake by the following command:

```
sudo apt-get install cmake
```

After this, run the following command one by one.

```
mkdir build
cd build
cmake ..
make
```

After this, the binaries will be generated in the “build” folder. Note that there is a possibility that the compiler could crash when compiling in the BUPTIA virtual machine, if this happens, run the “make” command again.

To start the server, run the “dns_server” executable in the terminal with one of the following commands, each of these commands starts up a DNS server in a specific mode. Run these commands respectively in different terminals to start the whole DNS system.

```
# "sudo" should be used since most linux distribution don't allow
# non-root to bind on port number lower than 1024
sudo ./dns_server root # starts the root name server
sudo ./dns_server s1   # starts the 1st name server
sudo ./dns_server s2   # starts the 2nd name server
sudo ./dns_server s3   # starts the 3rd name server
sudo ./dns_server s4   # starts the 4th name server
sudo ./dns_server local # starts the local name server
```

To execute the client, using the following command after starting all the servers:

```
./dns_client bupt.edu.cn MX # you can change the query name and type
```

The “nslookup” command can also be used as the DNS client. Note that when using the “nslookup” command we should always specify the query type.

```
# When querying from a non-local server
nslookup -query=A www.baidu.com 127.0.0.4
# When querying from the local server, "-vc" means using TCP
nslookup -vc -query=MX bupt.edu.cn 127.0.0.2
```

6. Results

The project has been tested on both Deepin v20 beta (Linux 64bit, installed on physical machine) and the BUPTIA virtual machine (Ubuntu Server 14.04 32bit). The following screen shots are taken in the Deepin OS.

6.1 Type-A query

After starting up all the servers, type the following command in the terminal, and its result will be:

```

liu@ThinkP liu@ThinkP liu@ThinkP liu@ThinkP liu@ThinkP liu@ThinkP liu@ThinkP
(base) liu@ThinkPad-P70: ~/Desktop/Project_DNS/cmake-build-debug$ ./dns_client www.bupt.edu.cn A
[ INFO ] Server: 127.0.0.2
[ INFO ] Address: 127.0.0.2#53

[ TRACE ] [ dns_network] Sending packet to 127.0.0.2:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x09f0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Server respond in 25.766001 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.2:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x09f0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 2
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Answers
[ TRACE ] www.bupt.edu.cn: type CNAME, class IN, cname vn64.bupt.edu.cn
[ TRACE ] vn64.bupt.edu.cn: type A, class IN, addr 211.68.69.240
[ TRACE ] [ dns_network] END of DNS packet.

[ INFO ] Answers:
[ INFO ] www.bupt.edu.cn canonical name = vn64.bupt.edu.cn
[ INFO ] vn64.bupt.edu.cn internet address = 211.68.69.240
[ INFO ]
(base) liu@ThinkPad-P70: ~/Desktop/Project_DNS/cmake-build-debug$

```

The client have received the IP address and the canonical name of the requested domain name. The content of the outgoing and incoming packets are printed in the Wireshark-like format. During this query, the output of the local DNS server program is:

```
(base) liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - - - x
[ INFO ] Listening on TCP port 53 on 127.0.0.2
[ TRACE ] [ dns_network ] Accepted connection from 127.0.0.1:40794
[ TRACE ] [ dns_network ] Received packet from 127.0.0.1:40794 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x09f0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network ] END of DNS packet.
[WARNING] [ dns_query ] Record not found in local cache, start iterative query...
[ TRACE ] [ dns_query ] Sending query request to root.local (127.0.0.7)
[ TRACE ] [ dns_network ] Sending packet to 127.0.0.7:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries:
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network ] END of DNS packet.
```



```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - □ ×
[ TRACE ] [ dns_network] Server respond in 1.270000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.7:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 1
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Authoritative nameservers
[ TRACE ] cn: type NS, class IN, ns ns1.local
[ TRACE ] Additional Records
[ TRACE ] ns1.local: type A, class IN, addr 127.0.0.3
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_query ] Sending query request to ns1.local (127.0.0.3)
[ TRACE ] [ dns_network] Sending packet to 127.0.0.3:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.
```

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - □ ×
[ TRACE ] [ dns_network] Server respond in 1.293000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.3:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 1
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Authoritative nameservers
[ TRACE ] edu.cn: type NS, class IN, ns ns3.local
[ TRACE ] Additional Records
[ TRACE ] ns3.local: type A, class IN, addr 127.0.0.5
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_query ] Sending query request to ns3.local (127.0.0.5)
[ TRACE ] [ dns_network] Sending packet to 127.0.0.5:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.
```

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - □ ×
[ TRACE ] [ dns_network] Server respond in 1.198000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.5:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 2
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] www.bupt.edu.cn: type CNAME, class IN, cname vn64.bupt.edu.cn
[ TRACE ] 它将指向vn64.bupt.edu.cn: type A, class IN, addr 211.68.69.240
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:40794 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x09f0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 2
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] www.bupt.edu.cn: type CNAME, class IN, cname vn64.bupt.edu.cn
[ TRACE ] vn64.bupt.edu.cn: type A, class IN, addr 211.68.69.240
[ TRACE ] [ dns_network] END of DNS packet.
```

From the screen shots above, we can see that the local DNS server first look up the cache, then query the name servers root, ns1 and ns3 in order. In the response of root, the name and IP address of ns1 is given, then in the response of ns1, it gives the address of ns3. Then ns3 respond with the desired records to the local DNS server. The output of the root server, s1 and s3 are:

```

liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ sudo ./dns_server root
[sudo] password for liu:
[ INFO ] Listening on UDP port 53 on 127.0.0.7
[ TRACE ] [ dns_network] Received packet from 127.0.0.1:41244 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:41244 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 1
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Authoritative nameservers
[ TRACE ] cn: type NS, class IN, ns ns1.local
[ TRACE ] Additional Records
[ TRACE ] ns1.local: type A, class IN, addr 127.0.0.3
[ TRACE ] [ dns_network] END of DNS packet.

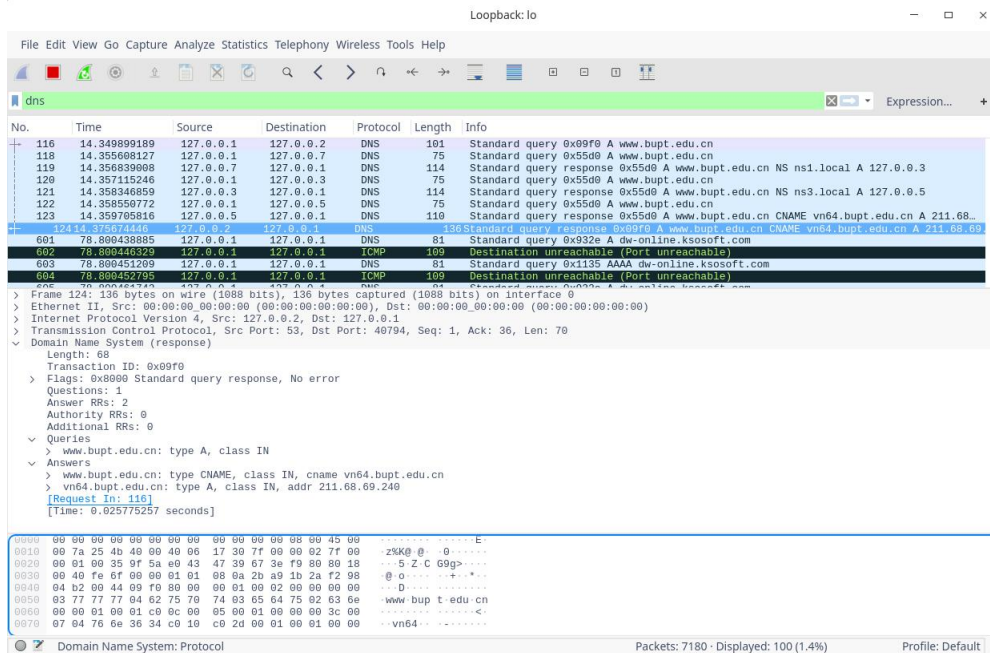
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Thi
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ sudo ./dns_server s1
[sudo] password for liu:
[ INFO ] Listening on UDP port 53 on 127.0.0.3
[ TRACE ] [ dns_network] Received packet from 127.0.0.1:49683 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:49683 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 1
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Authoritative nameservers
[ TRACE ] edu.cn: type NS, class IN, ns ns3.local
[ TRACE ] Additional Records
[ TRACE ] ns3.local: type A, class IN, addr 127.0.0.5
[ TRACE ] [ dns_network] END of DNS packet.

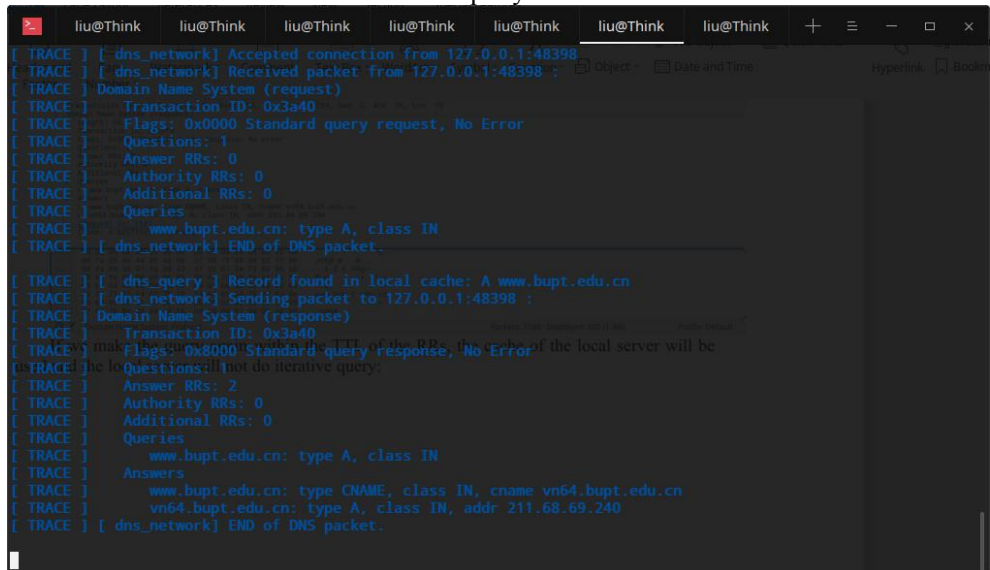
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ sudo ./dns_server s3
[sudo] password for liu:
[ INFO ] Listening on UDP port 53 on 127.0.0.5
[ TRACE ] [ dns_network] Received packet from 127.0.0.1:58947 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0 Standard query 0x09f0 A www.bupt.edu.cn
[ TRACE ] Flags: 0x0000 Standard query request, No Error Standard query 0x09f0 A www.bupt.edu.cn
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0 75 Standard query 0x55d0 A www.bupt.edu.cn
[ TRACE ] Authority RRs: 0 1 Standard query response 0x55d0 A www.bupt.edu.cn NS ns3.local A 127.0.0.5
[ TRACE ] Additional RRs: 0 75 Standard query 0x55d0 A www.bupt.edu.cn
[ TRACE ] Queries Standard query response 0x55d0 A www.bupt.edu.cn CNAME vn64.bupt.edu.cn A
[ TRACE ] www.bupt.edu.cn: type A, class IN Standard query response 0x09f0 A www.bupt.edu.cn CNAME vn64.bupt.edu.cn A
[ TRACE ] [ dns_network] END of DNS packet. Standard query 0x932e A dw-online.ksoofo.com
[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:58947 : Standard query 0x932e A dw-online.ksoofo.com
[ TRACE ] Domain Name System (response) Standard query 0x932e A dw-online.ksoofo.com
[ TRACE ] Transaction ID: 0x55d0 Standard query response, No Error (Port unreachable)
[ TRACE ] Flags: 0x8000 Standard query response, No Error Standard query 0x1135 AAAA dw-online.ksoofo.com
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 2(1s) on interface 0
[ TRACE ] Authority RRs: 0 1.00:00:00 (00:00:00:00:00:00)
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Answers
[ TRACE ] www.bupt.edu.cn: type CNAME, class IN, cname vn64.bupt.edu.cn
[ TRACE ] vn64.bupt.edu.cn: type A, class IN, addr 211.68.69.240
[ TRACE ] [ dns_network] END of DNS packet.

```

All the packets can be captured and parsed by Wireshark without any error (The packets below the selected packet is sent by other programs in my system):



If we make the query again within the TTL of the RRs, the cache of the local server will be used and the local server will not do iterative query:



The packets captured in Wireshark when the cache is used, no iterative query is made:

7824	1041.146812..	127.0.0.1	127.0.0.2	DNS	101	Standard query 0x4cf0 A www.bupt.edu.cn
7826	1041.147302..	127.0.0.1	127.0.0.7	DNS	75	Standard query 0x55d0 A www.bupt.edu.cn
7827	1041.148728..	127.0.0.1	127.0.0.1	DNS	114	Standard query response 0x55d0 A www.bupt.edu.cn NS ns1.local A 127.0.0.3
7828	1041.148951..	127.0.0.1	127.0.0.3	DNS	75	Standard query 0x55d0 A www.bupt.edu.cn
7829	1041.150261..	127.0.0.3	127.0.0.1	DNS	114	Standard query response 0x55d0 A www.bupt.edu.cn NS ns3.local A 127.0.0.5
7830	1041.150549..	127.0.0.1	127.0.0.5	DNS	75	Standard query 0x55d0 A www.bupt.edu.cn
7831	1041.151680..	127.0.0.1	127.0.0.1	DNS	110	Standard query response 0x55d0 A www.bupt.edu.cn CNAME vn64.bupt.edu.cn A 211.68.69.240
7832	1041.168918..	127.0.0.2	127.0.0.1	DNS	136	Standard query response 0x4cf0 A www.bupt.edu.cn CNAME vn64.bupt.edu.cn A 211.68.69.240
7848	1042.307847..	127.0.0.1	127.0.0.2	DNS	101	Standard query 0x3a40 A www.bupt.edu.cn
7850	1042.307891..	127.0.0.2	127.0.0.1	DNS	136	Standard query response 0x3a40 A www.bupt.edu.cn CNAME vn64.bupt.edu.cn A 211.68.69.240

The cache data produced in the database:

id	name	ttl	class	type	data	timestamp
12	12 www.bupt.edu.cn	60	1	5	vn64.bupt.edu.cn	1591787460
13	13 vn64.bupt.edu.cn	60	1	1	211.68.69.240	1591787460

6.2 Type-MX query

Make a MX typed request in the client:

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ ./dns_client bupt.edu.cn MX
[ INFO ] Server: 127.0.0.2
[ INFO ] Address: 127.0.0.2#53

[ TRACE ] [ dns_network] Sending packet to 127.0.0.2:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0xe730
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] bupt.edu.cn: type MX, class IN
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Server respond in 20.164000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.2:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0xe730
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 1
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ] bupt.edu.cn: type MX, class IN
[ TRACE ] Answers
[ TRACE ] bupt.edu.cn: type MX, class IN, preference 0, mx mx.bupt.edu.cn
[ TRACE ] Additional Records
[ TRACE ] mx.bupt.edu.cn: type A, class IN, addr 183.3.235.87
[ TRACE ] [ dns_network] END of DNS packet.
[ INFO ] Answers:
[ INFO ] bupt.edu.cn mail exchanger = 0,mx.bupt.edu.cn
[ INFO ] Flags: 0x8000 Standard query response, No error
[ INFO ] Additional records: Questions: 1
[ INFO ] mx.bupt.edu.cn internet address = 183.3.235.87
```

We can see that both the domain name of the mail exchange service and its IP address is received from the servers.

The output of the local server is:

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
[ TRACE ] [ dns_network] Accepted connection from 127.0.0.1:51092
[ TRACE ] [ dns_network] Received packet from 127.0.0.1:51092 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0xe730
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] bupt.edu.cn: type MX, class IN
[ TRACE ] [ dns_network] END of DNS packet.

[WARNING] [ dns_query ] Record not found in local cache, start iterative query...
[ TRACE ] [ dns_query ] Sending query request to root.local (127.0.0.7)
[ TRACE ] [ dns_network] Sending packet to 127.0.0.7:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] bupt.edu.cn: type MX, class IN
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Server respond in 1.153000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.7:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 1
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] bupt.edu.cn: type MX, class IN
[ TRACE ] Answers
[ TRACE ] mx.bupt.edu.cn: type A, class IN, addr 183.3.235.87
[ TRACE ] [ dns_network] END of DNS packet.
```

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - □ ×
[ TRACE ] [ dns_network] Server respond in 1.153000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.7:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 1
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ]     bupt.edu.cn: type MX, class IN
[ TRACE ] Authoritative nameservers
[ TRACE ]     cn: type NS, class IN, ns ns1.local
[ TRACE ] Additional Records
[ TRACE ]     ns1.local: type A, class IN, addr 127.0.0.3
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_query ] Sending query request to ns1.local (127.0.0.3)
[ TRACE ] [ dns_network] Sending packet to 127.0.0.3:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ]     bupt.edu.cn: type MX, class IN
[ TRACE ] [ dns_network] END of DNS packet.
```

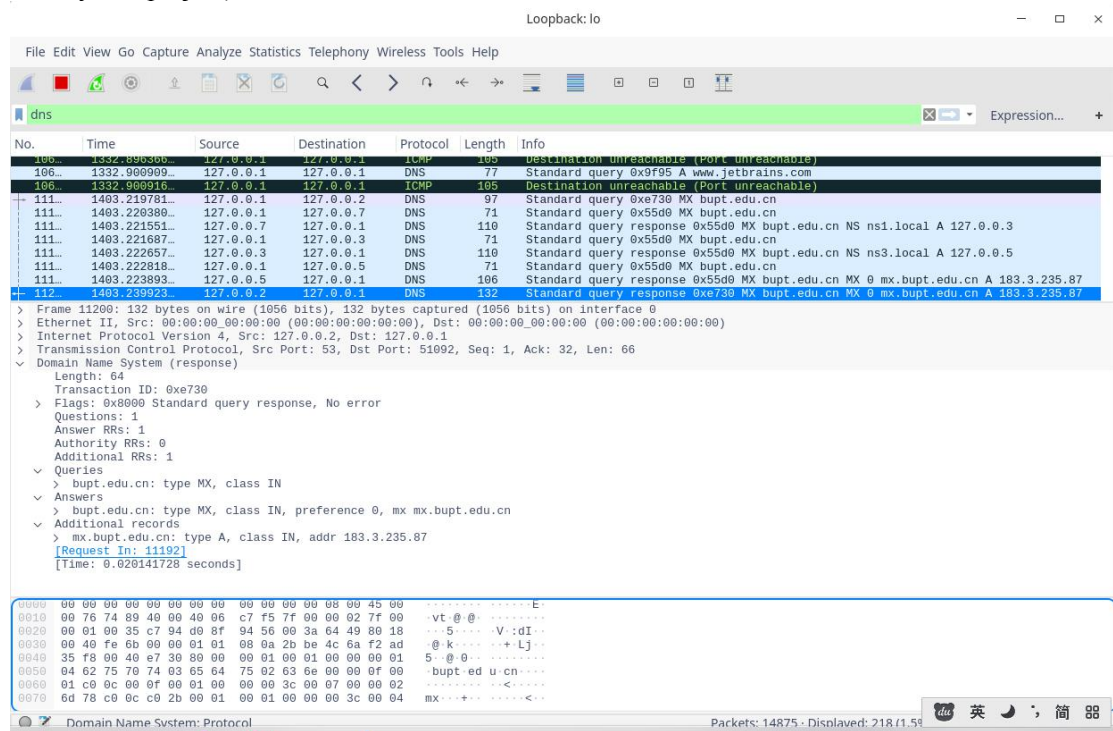
```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - □ ×
[ TRACE ] [ dns_network] Server respond in 0.978000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.3:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 1
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ]     bupt.edu.cn: type MX, class IN
[ TRACE ] Authoritative nameservers
[ TRACE ]     edu.cn: type NS, class IN, ns ns3.local
[ TRACE ] Additional Records
[ TRACE ]     ns3.local: type A, class IN, addr 127.0.0.5
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_query ] Sending query request to ns3.local (127.0.0.5)
[ TRACE ] [ dns_network] Sending packet to 127.0.0.5:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ]     bupt.edu.cn: type MX, class IN
[ TRACE ] [ dns_network] END of DNS packet.
```

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - □ ×
[ TRACE ] [ dns_network] Server respond in 1.062000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.5:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 1
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ]     bupt.edu.cn: type MX, class IN
[ TRACE ] Answers
[ TRACE ]     bupt.edu.cn: type MX, class IN, preference 0, mx mx.bupt.edu.cn
[ TRACE ] Additional Records
[ TRACE ]     mx.bupt.edu.cn: type A, class IN, addr 183.3.235.87
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:51092 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0xe730
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 1
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ]     bupt.edu.cn: type MX, class IN
[ TRACE ] Answers
[ TRACE ]     bupt.edu.cn: type MX, class IN, preference 0, mx mx.bupt.edu.cn
[ TRACE ] Additional Records
[ TRACE ]     mx.bupt.edu.cn: type A, class IN, addr 183.3.235.87
[ TRACE ] [ dns_network] END of DNS packet.
```


The packets captured by Wireshark are (The DNS request of name www.jetbrains.com is not produced by this project):

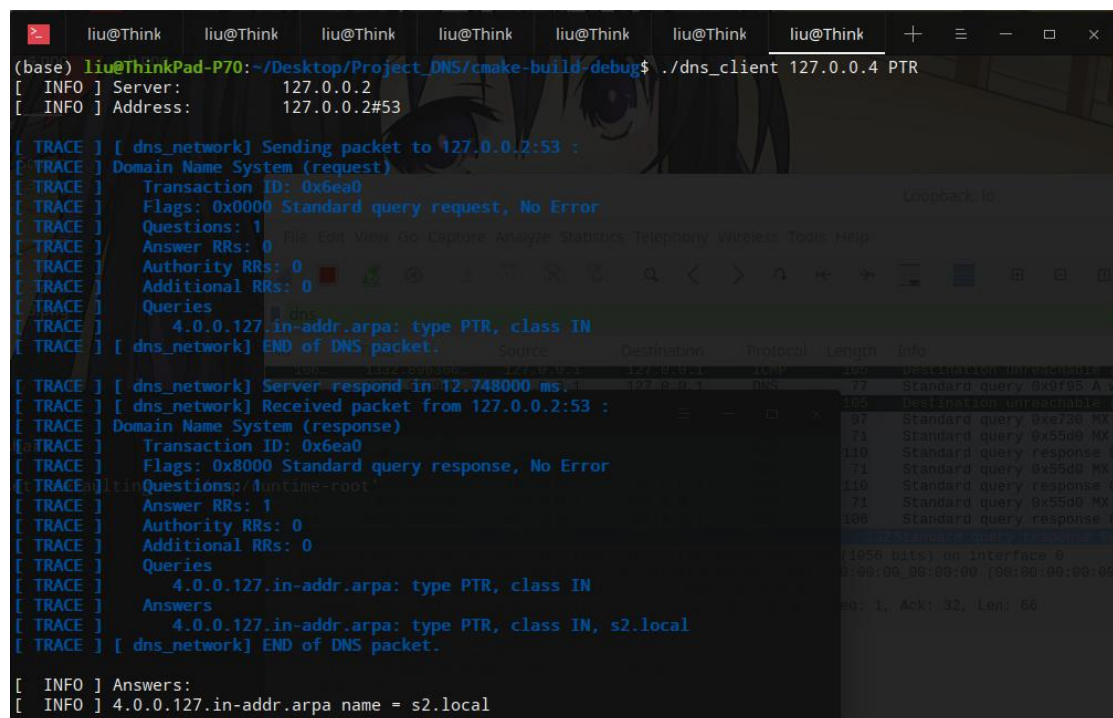


The cache also includes the IP address of the mail exchange service:

	id	name	tll	class	type	data	timestamp
14	14	bupt.edu.cn	60	1	15	0,mx.bupt.edu.cn	1591787823
15	15	mx.bupt.edu.cn	60	1	1	183.3.235.87	1591787823

6.3 Type-PTR query

Make a PTR typed query in the client, the queried IP address is converted to the format ends with “in-addr.arpa”:



The output of the local server is:

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + ≡ - □ ×
[ TRACE ] [ dns_network ] Accepted connection from 127.0.0.1:56474
[ TRACE ] [ dns_network ] Received packet from 127.0.0.1:56474 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x6ea0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN
[ TRACE ] [ dns_network ] END of DNS packet.

[WARNING] [ dns_query ] Record not found in local cache, start iterative query...
[ TRACE ] [ dns_query ] Sending query request to root.local (127.0.0.7)
[ TRACE ] [ dns_network ] Sending packet to 127.0.0.7:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN
[ TRACE ] [ dns_network ] END of DNS packet.

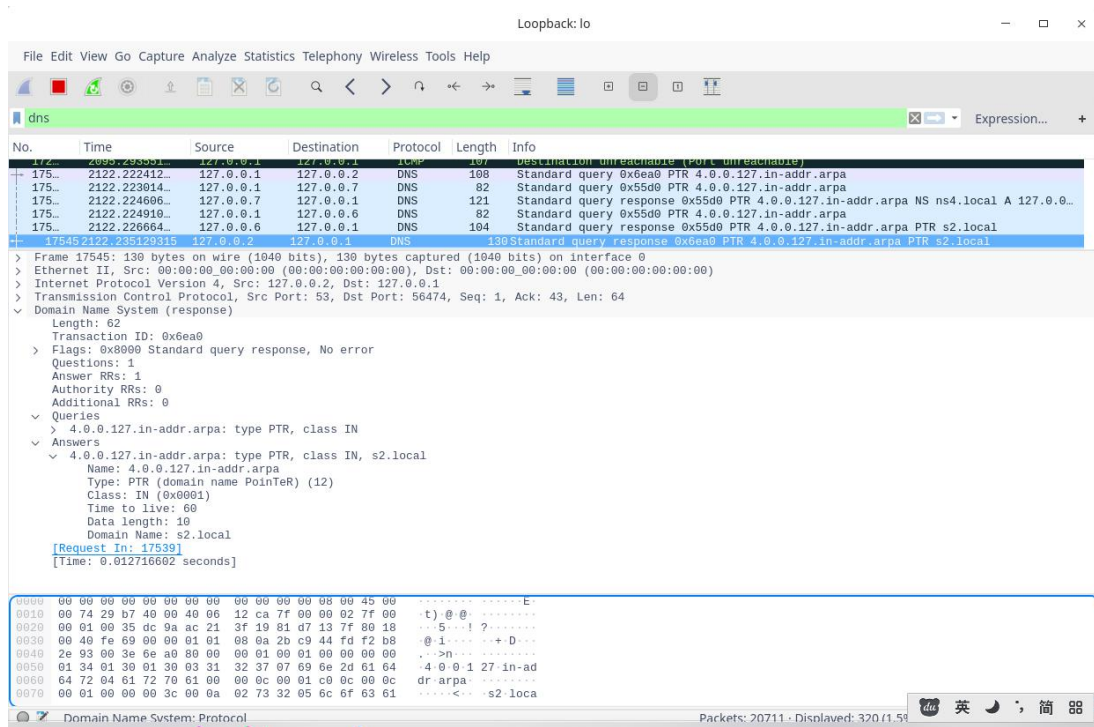
[ TRACE ] [ dns_network ] Server respond in 1.659000 ms.
[ TRACE ] [ dns_network ] Received packet from 127.0.0.7:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 1
[ TRACE ] Additional RRs: 1
[ TRACE ] Queries
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN
[ TRACE ] Authoritative nameservers
[ TRACE ] in-addr.arpa: type NS, class IN, ns ns4.local
[ TRACE ] Additional Records
[ TRACE ] ns4.local: type A, class IN, addr 127.0.0.6
[ TRACE ] [ dns_network ] END of DNS packet.

[ TRACE ] [ dns_network ] END of DNS packet.
[ TRACE ] [ dns_query ] Sending query request to ns4.local (127.0.0.6)
[ TRACE ] [ dns_network ] Sending packet to 127.0.0.6:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN
[ TRACE ] [ dns_network ] END of DNS packet.

[ TRACE ] [ dns_network ] Server respond in 1.837000 ms.
[ TRACE ] [ dns_network ] Received packet from 127.0.0.6:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 1
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN
[ TRACE ] Answers
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN, s2.local
[ TRACE ] [ dns_network ] END of DNS packet.

[ TRACE ] [ dns_network ] Sending packet to 127.0.0.1:56474 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x6ea0
[ TRACE ] Flags: 0x8000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 1
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN
[ TRACE ] Answers
[ TRACE ] 4.0.0.127.in-addr.arpa: type PTR, class IN, s2.local
[ TRACE ] [ dns_network ] END of DNS packet.
```

The packets captured by Wireshark are:



The cache produced during this query:

	id	name	tll	class	type	data	timestamp
16	16	4.0.0.127.in-addr.arpa	60	1	12	s2.local	1591788542

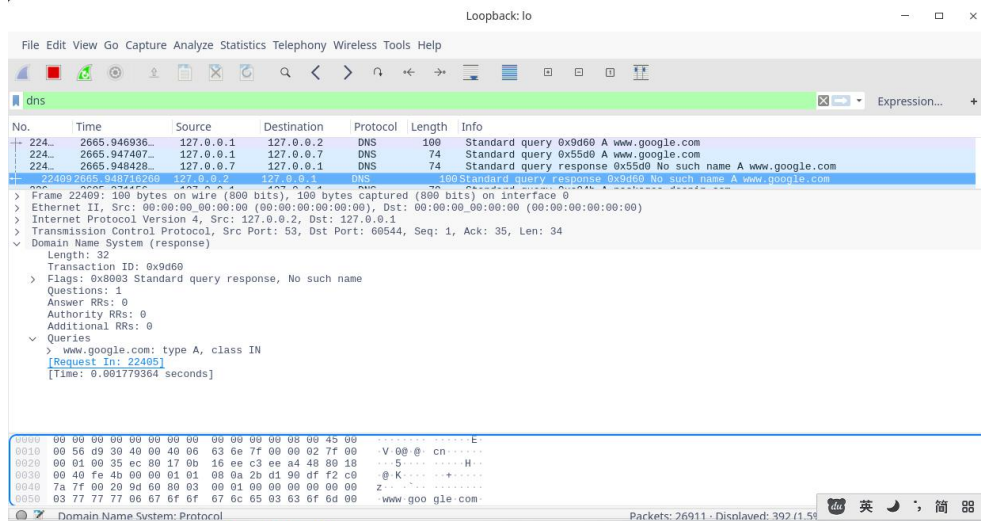
6.4 Error handling

When querying a name that is not existed in the records database:

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think + - □ ×
[ INFO ] liu@ThinkPad-P70: ~/Desktop/Project_DNS/cmake-build-debug$ ./dns_client www.google.com A
[ INFO ] Server: 127.0.0.2
[ INFO ] Address: 127.0.0.2#53
[ TRACE ] [ dns_network ] Sending packet to 127.0.0.2:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x9d60
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.google.com: type A, class IN
[ TRACE ] [ dns_network ] END of DNS packet.
[ TRACE ] [ dns_network ] Server respond in 1.829000 ms.
[ TRACE ] [ dns_network ] Received packet from 127.0.0.2:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x9d60
[ TRACE ] Flags: 0x8003 Standard query response, Name does not exists
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.google.com: type A, class IN
[ TRACE ] [ dns_network ] END of DNS packet.
[ ERROR ] [ dns_client ] Query failed: Name does not exists (3).
(base) liu@ThinkPad-P70: ~/Desktop/Project_DNS/cmake-build-debug$
```

From the screen shot, we can see that the DNS client receives a response code of 3, which indicates that the name does not exists.

The status code can also be seen in Wireshark:



When the user typed invalid query type in the client:

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ ./dns_client www.bupt.edu.cn AAAA
[ INFO ] Server: 127.0.0.2
[ INFO ] Address: 127.0.0.2#53

[ ERROR ] [ dns_common ] Unknown DNS type 'AAAA'
[ TRACE ] [ dns_network ] Sending packet to 127.0.0.2:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x0990
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ ERROR ] [ dns_common ] The DNS type 0 is currently not supported.
[ TRACE ] www.bupt.edu.cn: type [UNKNOWN], class IN
[ TRACE ] [ dns_network ] END of DNS packet.

[ TRACE ] [ dns_network ] Server respond in 0.170000 ms.
[ TRACE ] [ dns_network ] Received packet from 127.0.0.2:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x0990
[ TRACE ] Flags: 0x8004 Standard query response, Unsupported query type
[ TRACE ] Questions: 0
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] [ dns_network ] END of DNS packet.

[ ERROR ] [ dns_client ] Query failed: Unsupported query type (4).
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$
```

If invalid query type is found, the client will give a error message, and the query type will be set to 0, which is not used in the DNS standard. The server response also give the error code.

The output of the local server is:

```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
[ TRACE ] [ dns_network ] Accepted connection from 127.0.0.1:36770
[ TRACE ] [ dns_network ] Received packet from 127.0.0.1:36770 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x0990
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ ERROR ] [ dns_common ] The DNS type 0 is currently not supported.
[ TRACE ] www.bupt.edu.cn: type [UNKNOWN], class IN
[ TRACE ] [ dns_network ] END of DNS packet.

[ ERROR ] [ dns_common ] The DNS type 0 is currently not supported.
[ TRACE ] [ dns_network ] Sending packet to 127.0.0.1:36770 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x0990
[ TRACE ] Flags: 0x8004 Standard query response, Unsupported query type
[ TRACE ] Questions: 0
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] [ dns_network ] END of DNS packet.
```

To make the request packet corrupt, modify the code in the function “DNS_network_send_query_tcp” in “dns_network.c” to make the client send truncated packet:

```

327     uint16 len = (uint16) buffer->pos;
328     *((uint16 *) send_buf) = htons(len);
329     if (send(sock, send_buf, /*len + 2*/ n: 15, flags: 0) < 0) {
330         DNS_log_error( format: "[ dns_network] Failed to send TCP packet to DNS server: %s", strerror(errno));
331         close(sock);
332         return NULL;
333     }

```

Compile the modified client, then make a request with the modified client:

```

liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ ./dns_client www.bupt.edu.cn A
[ INFO ] Server: 127.0.0.2
[ INFO ] Address: 127.0.0.2#53

[ TRACE ] [ dns_network] Sending packet to 127.0.0.2:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x0210
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Server respond in 0.103000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.2:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8001 Standard query response, Format error
[ TRACE ] Questions: 0
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] [ dns_network] END of DNS packet.

[ ERROR ] [ dns_client ] Query failed: Format error (1).

```

As we can see, if the request packet is corrupted, the server will send a error code indicates there is format error in the request. And the output of the local server also indicates error:

```

liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:36770 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x0990
[ TRACE ] Flags: 0x8004 Standard query response, Unsupported query type
[ TRACE ] Questions: 0
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] [ dns_network] END of DNS packet.

[ TRACE ] [ dns_network] Accepted connection from 127.0.0.1:39690
[ ERROR ] [ dns_io ] DNS_buffer_read_DNS_name (Line 310) failed, Buffer boundary reached
[ ERROR ] [ dns_io ] DNS_buffer_read_query (Line 473) failed, Buffer boundary reached
[ ERROR ] [ dns_io ] DNS_buffer_read_packet (Line 492) failed, Buffer boundary reached
[ ERROR ] [ dns_network] Failed to decode incoming packet as DNS packet, the length is 15
[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:39690 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x8001 Standard query response, Format error
[ TRACE ] Questions: 0
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries
[ TRACE ] [ dns_network] END of DNS packet.

```

The local server gives error message that the reading operation of the buffer is failed due to the insufficient length of the packet. If there is no insurance that the read/write operation of the buffer does not exceed the boundary of the buffer, the program would crash for segmentation fault.

Undo the modification on the client, make sure it works again. Then modify one RR of type A in the database to make its data not an IP address:

	id	name	ttl	class	type	data
1	1	bupt.edu.cn	60	1	15	mx.bupt.edu.cn
2	2	mx.bupt.edu.cn	60	1	1	183.3.235.87
3	3	www.bupt.edu.cn	60	1	5	vn64.bupt.edu.cn
4	4	vn64.bupt.edu.cn	60	1	1	www.bilibili.com

Then make request of this domain name with the client, the IP address is set to 255.255.255.255:

```

liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ ./dns_client www.bupt.edu.cn A
[ INFO ] Server: 127.0.0.2
[ INFO ] Address: 127.0.0.2#53
[ TRACE ] [ dns_network] Sending packet to 127.0.0.2:53 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0xc8c0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries: main.s3 [dns_database.db] main.cache [dns_database.db] dns_database.c README.md dns
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.
[ TRACE ] [ dns_network] Server respond in 19.864000 ms.
[ TRACE ] [ dns_network] Received packet from 127.0.0.2:53 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0xc8c0
[ TRACE ] Flags: 0x0000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 2
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries:
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Answers
[ TRACE ] www.bupt.edu.cn: type CNAME, class IN, cname vn64.bupt.edu.cn
[ TRACE ] vn64.bupt.edu.cn: type A, class IN, addr 255.255.255.255
[ TRACE ] [ dns_network] END of DNS packet.
[ INFO ] Answers:
[ INFO ] www.bupt.edu.cn canonical name = vn64.bupt.edu.cn
[ INFO ] vn64.bupt.edu.cn internet address = 255.255.255.255
[ INFO ]
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$

```

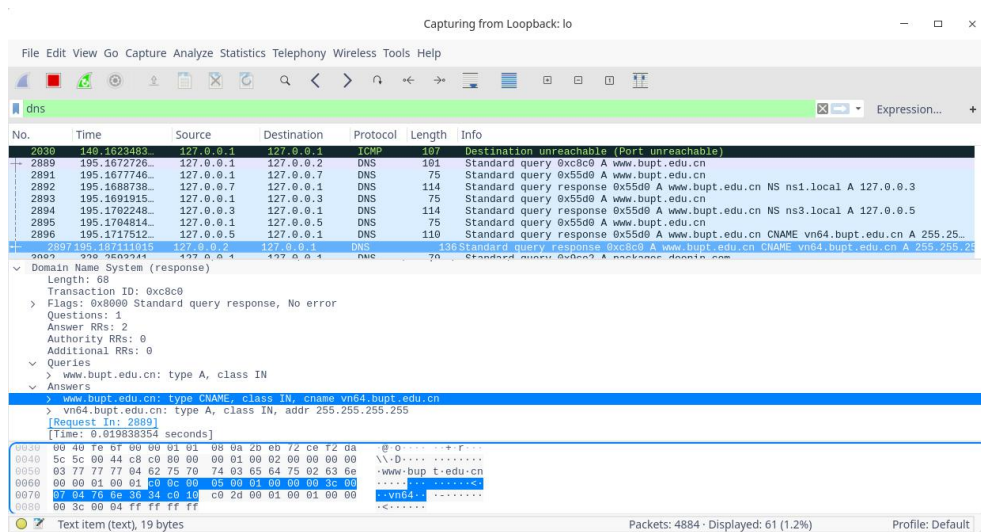
The server s2 (where the modified RR locates) shows a warning:

```

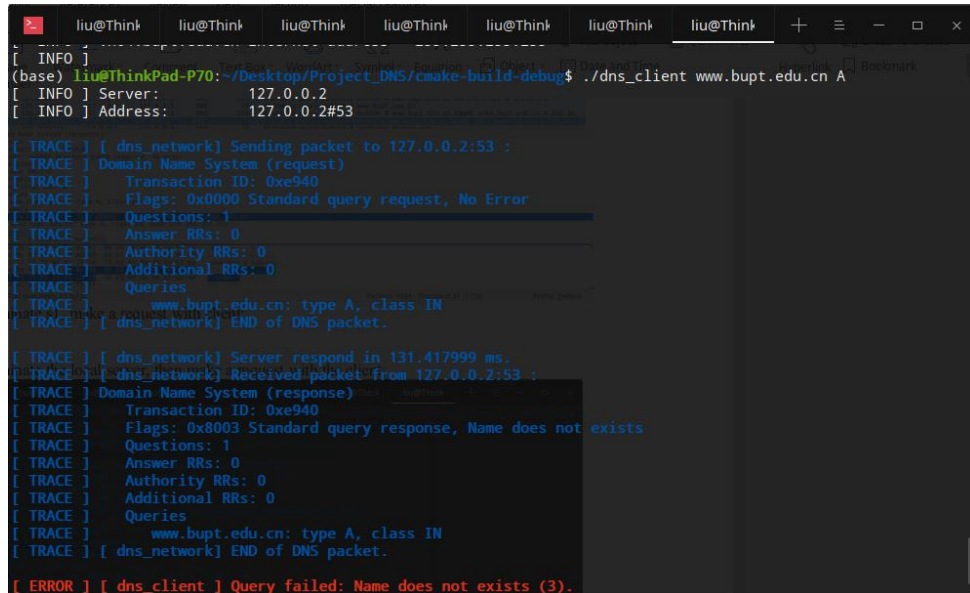
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
[ TRACE ] [ dns_network] Received packet from 127.0.0.1:48658 :
[ TRACE ] Domain Name System (request)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query request, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 0
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries:
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network] END of DNS packet.
[ TRACE ] [ dns_network] Sending packet to 127.0.0.1:48658 :
[ TRACE ] Domain Name System (response)
[ TRACE ] Transaction ID: 0x55d0
[ TRACE ] Flags: 0x0000 Standard query response, No Error
[ TRACE ] Questions: 1
[ TRACE ] Answer RRs: 2
[ TRACE ] Authority RRs: 0
[ TRACE ] Additional RRs: 0
[ TRACE ] Queries:
[ TRACE ] www.bupt.edu.cn: type A, class IN
[ TRACE ] Answers
[ TRACE ] www.bupt.edu.cn: type CNAME, class IN, cname vn64.bupt.edu.cn
[ TRACE ] vn64.bupt.edu.cn: type A, class IN, addr www.bilibili.com
[ TRACE ] [ dns_network] END of DNS packet.
[WARNING] [ dns_io ] Expected IP address in RR of type A, but got 'www.bilibili.com'.

```

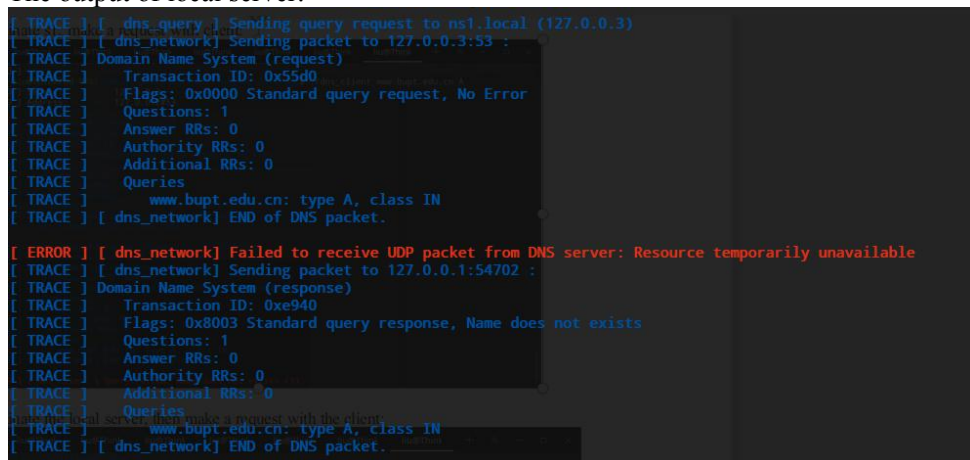

There is no error in the packets in Wireshark (except the IP address):



Terminate s1, make a request with client, after waiting for 10 seconds the client will give a error:

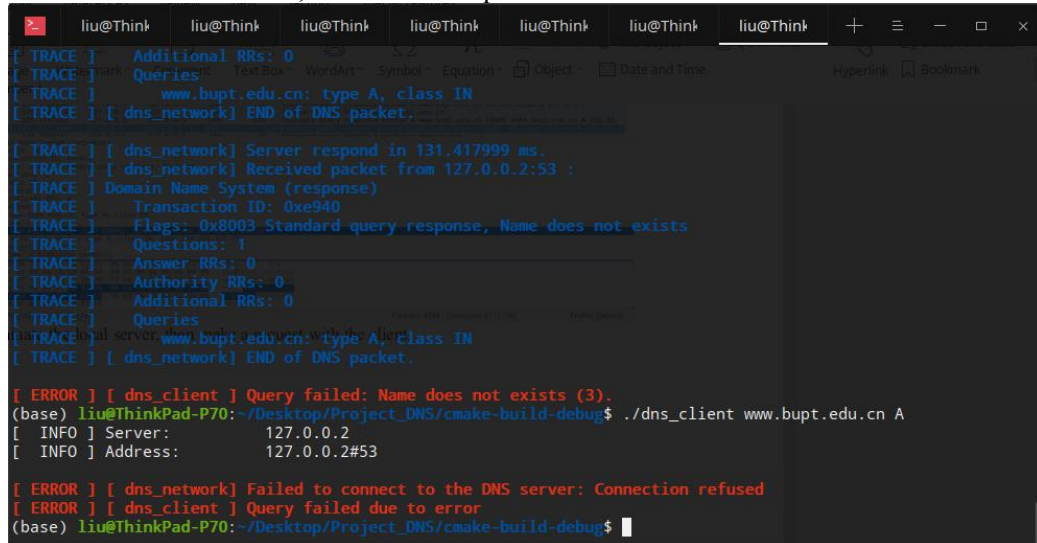


The output of local server:



When querying with TCP, there is a 10 second timeout set in the program, if the server do not respond in time, the local server will send a “no answer” response.

Terminate the local server, then make a request with the client:



```
liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think liu@Think
[ TRACE ] [ dns_network ] Additional RRs: 0
[ TRACE ] [ dns_network ] Queries
[ TRACE ] [ dns_network ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network ] END of DNS packet.
[ TRACE ] [ dns_network ] Server respond in 131.417999 ms.
[ TRACE ] [ dns_network ] Received packet from 127.0.0.2:53 :
[ TRACE ] [ dns_network ] Domain Name System (response)
[ TRACE ] [ dns_network ] Transaction ID: 0xe940
[ TRACE ] [ dns_network ] Flags: 0x8003 Standard query response, Name does not exists
[ TRACE ] [ dns_network ] Questions: 1
[ TRACE ] [ dns_network ] Answer RRs: 0
[ TRACE ] [ dns_network ] Authority RRs: 0
[ TRACE ] [ dns_network ] Additional RRs: 0
[ TRACE ] [ dns_network ] Queries
[ TRACE ] [ dns_network ] www.bupt.edu.cn: type A, class IN
[ TRACE ] [ dns_network ] END of DNS packet.
[ ERROR ] [ dns_client ] Query failed: Name does not exists (3).
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$ ./dns_client www.bupt.edu.cn A
[ INFO ] Server: 127.0.0.2
[ INFO ] Address: 127.0.0.2#53
[ ERROR ] [ dns_network ] Failed to connect to the DNS server: Connection refused
[ ERROR ] [ dns_client ] Query failed due to error
(base) liu@ThinkPad-P70:~/Desktop/Project_DNS/cmake-build-debug$
```

Since the TCP connection cannot be established, the query cannot be continued.

7. Summary and Conclusion

The individual contribution of the group members of the project are:

LIU Tong: Since LIU have a higher coding ability and nicer coding style, the database module, IO module, query module are designed and composed by him. Since he have read a lot of open source code, he can write the project in a more professional way, which makes this project more extensible and stable. He is also responsible for tidying the code from the other group member.

YAN Yangtian: YAN is responsible for writing the main entries of both the DNS server and the client. He have good understanding of computer network, so he also writes the network related functions in this project.

The project is designed as multiple separated modules, which make this project more flexible and extensible. The extensibility of the project makes more possibility on future improvement, like supporting more query types . The design of the project also aims at stability, ensuring the program will not crashed easily.