1. Overview

The requirement of the project is to achieve Chinese domain name resolution based on the DNS (Domain Name System) client/server and a Linux command-line terminal. The purpose of the project is to help us get in-depth understanding of the DNS protocol by combining with previous experimental programming practice.

2. Requirements Analysis

2.1. Development Environment:

Linux environment and C language.

2.2. Functional requirements:

1. Realize the resolution of Chinese domain names. The project should support at least 4 top-level domains, and achieve the analysis of third-level domain names.
2. Resource Record types should include A, MX and CNAME. For MX type, IP address is required to be carried in Additional Section.
3. Iterative Resolution method should be supported.
4. Local server would have a cache. The project should print trace records to indicate the process. E.g. query path, server response time.
5. Between client and local DNS server, the transport layer should be TCP. Between DNS servers, the transport layer should be UDP.
6. Application layer protocol should be DNS, all DNS messages used in the communication process must be correctly parsed by Wireshark.
7. The data maintenance mode of the server can use the file;
8. The program should contain detailed code comments, use a good programming style. Program should run stably and support error handling, such as invalid command, missing parameters, the same name processing and blank characters.

3. Preliminary Design

3.1. Decomposition of functional modules

According to requirement analysis, we divide the tasks into 4 parts: TCP/UDP Communication, Encode the DNS Packet into Buffer, Decode Buffer into DNS Packet and File Operation.

3.1.1. TCP/UDP Communication

In the project, TCP communication needs to be used between the client and the Local Server. UDP communication is used between the root side, TLD, 2LD, and the Local Server.

3.1.2. Encode the DNS Packet into Buffer

When the sender is ready to send a DNS packet, this module will be called to write the contents of the packet into the buffer.

3.1.3. Decode Buffer into DNS Packet

When the receiver receives data, it needs to use this module to read the data from buffer and construct a DNS packet to store the data.

3.1.4. File Operation

When the receiver receives data, this module would help read resource records from the database and write resource records to cache.

3.2. Relationship and Interface Between the Modules

The sender will encode DNS packet to buffer and send the data to the receiver by using TCP or UDP communication. The receiver decodes DNS packet from buffer and use the File Operation to query the local database. Local server would have a cache to efficient the process.

3.3. Overall Flow Chart

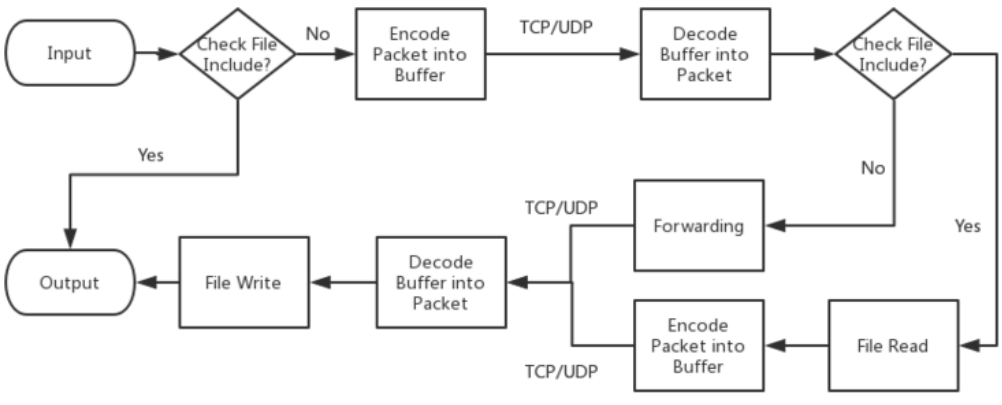


Figure 1. Flow Chart

3.4. Design of data structures

3.4.1. DNS Packet Format

Following the DNS Message Format(from RFC 1035), we design the DNS packet format.

(1) DNS Packet Header

```
struct dns_header{
  uint16_t id;
  uint16_t tag;
  uint16_t queryNum;
  uint16_t answerNum;
  uint16_t authorityNum;
  uint16_t additionNum;
};
```

(2) DNS Packet Question

```
struct dns_query{
  char *name;
  uint16_t qType;
  uint16_t qClass;
};
```

(3) DNS Packet RR

```
//A & CNAME Record
struct dns_rr{
  char *name;
  uint16_t type;
  uint16_t class;
  uint32_t ttl;
  uint16_t data_len;
  char *Data;
};
//MX Record
```

```
struct dns_MX_rr{
  char *name;
  uint16_t type;
  uint16_t class;
  uint32_t ttl;
  uint16_t data_len;
  uint16_t Preference;
  char *exchange;
};
```

3.4.2. Resource Record

The file format of each server file is not the same, therefore, we set up with an individual file for each server, a total of 6 files for the project.

(1) Local Server: The Local Server should set cache file and store every valid query record.

Figure 2. Cache File of Local Server

(2) Root Server: The Root Server should use file to store domain names and IP Address of all top-level domains.

Figure 3. Database of Root Server

(3) TLD Server: The TLD Server should set file to store domain names and IP Address of all second-level domains.

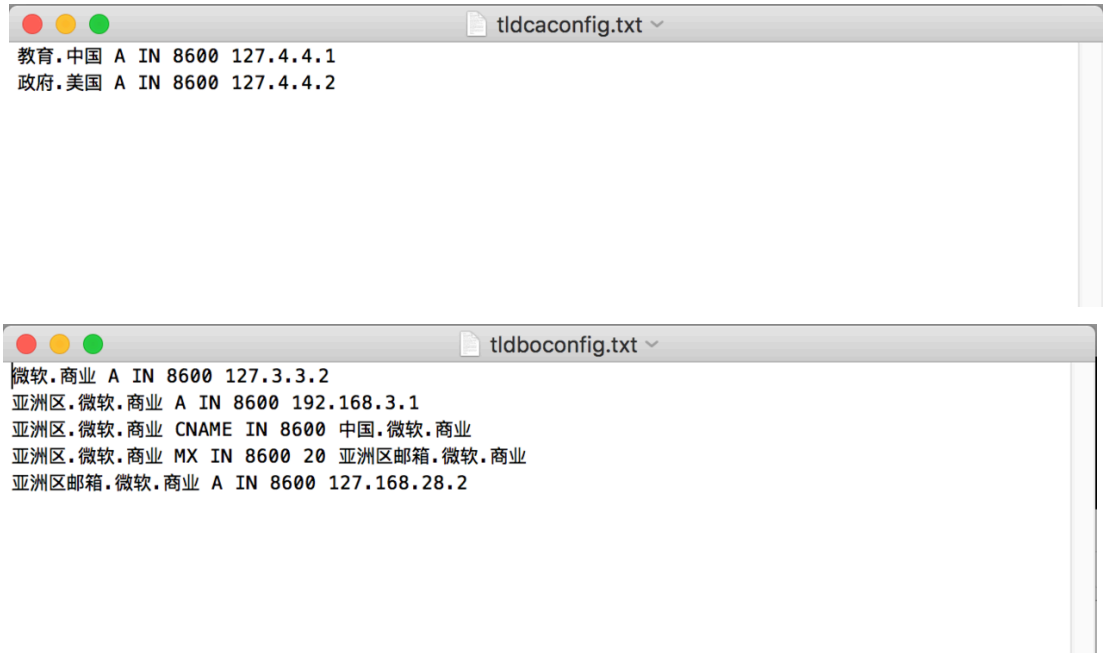


Figure 4. Database of TLD Server

(4) 2LD Server: 在 2LD Server 服务器需要设置数据库文件，存储所有三级域的 resource record of A, CNAME and MX. The 2LD Server should set up a file to store the Resource Record of all third level domain, include A, CNAME and MX type.

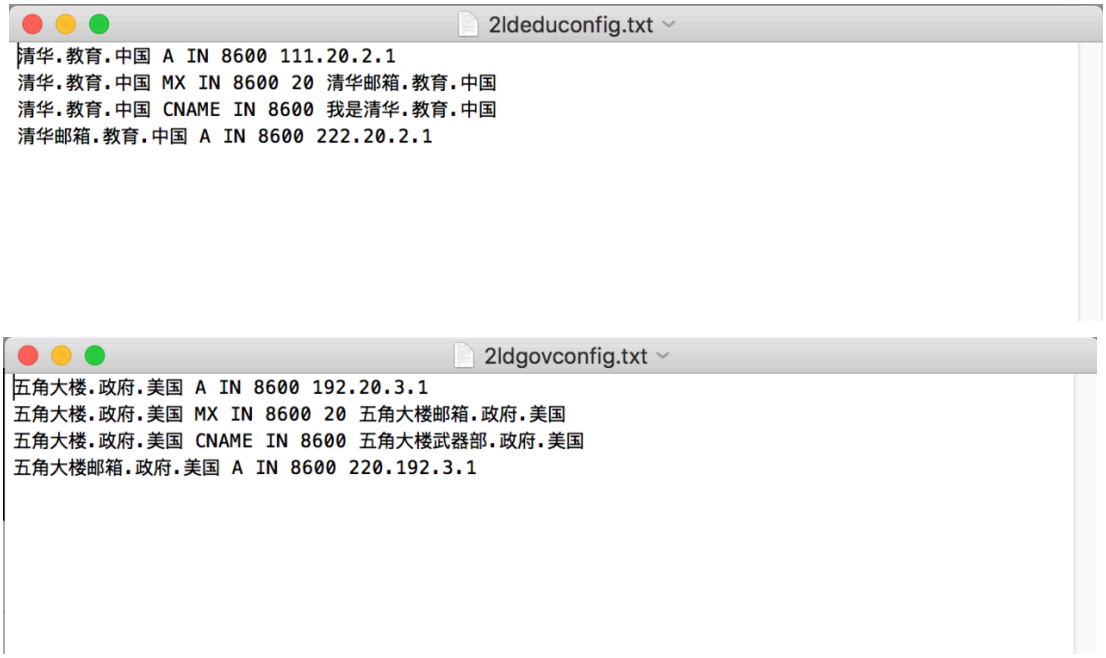


Figure 5. Database of 2LD Server

4. Detailed Design

According to the decomposition of the module 3.1, we introduce the detailed design of the module as following.

4.1. TCP/UDP Communication

When performing TCP communication, we should bind the server's IP and socket, and start monitoring. Therefore, there is listen( ) and accept( ). Once a client initiates a connection request, a TCP connection will be established, then it can realize data transmission. The methods are send( ) and receive( ).

When performing UDP communication, we also should bind the Server'IP and socket, then it is possible to perform data transmission with the client. There is no connection in UDP communication. The methods are sendto( ) and receivefrom( ).

The flow chart of function module is:

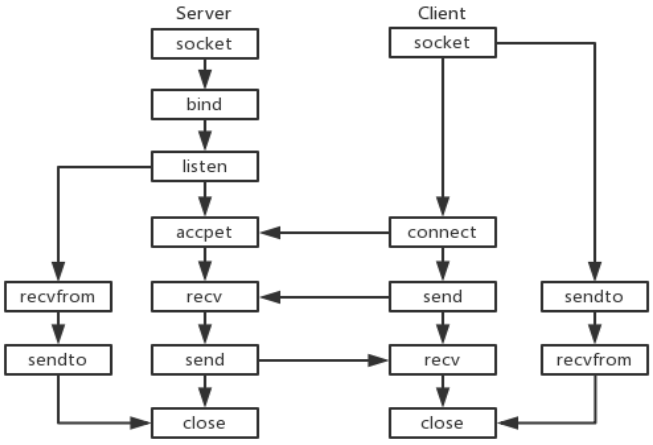


Figure 6. TCP Flow

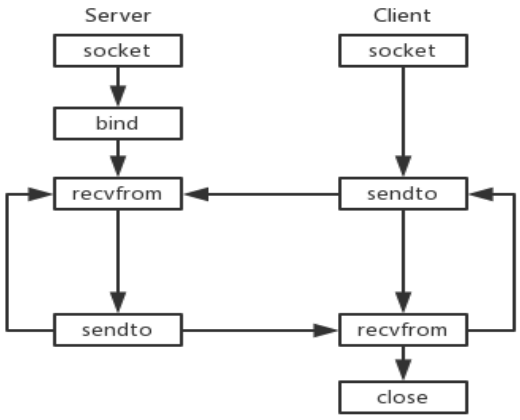


Figure 7. UDP Flow

4.2. Encode the DNS Packet into Buffer

In the process of encoding DNS packet into buffer, we need to write some method to put 2 byte or 4 byte char into buffer. Since the domain name is not a fix length, we need to write another method to encode domain name into buffer, also we should write some method to change the format of domain and IP Address so that it can be caught and decoded by Wireshark.

For example:

domain: **北邮. 教育. 中国** → 6 北邮 6 教育 6 中国 6

IP: 192.168.100.98 ——> c0a86462

The flow chart of function module is:

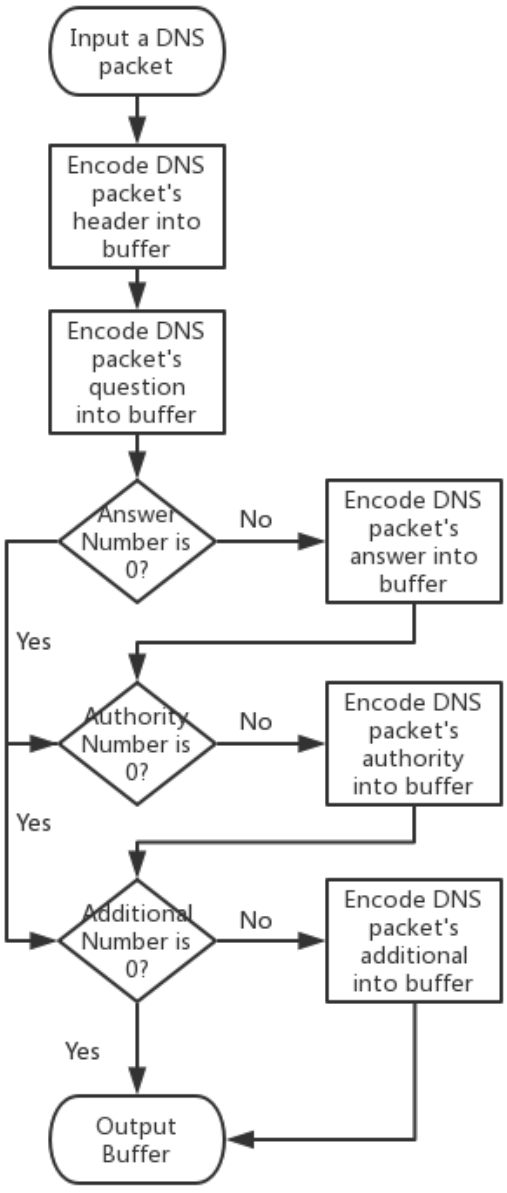


Figure 8. Encode Packet

4.3. Decode Buffer into DNS Packet

In the process of decoding buffer into DNS packet, we need to write get2byte or get4byte method to put char into DNS packet. Since the domain name is not the fix length, we need to write another method to decode domain name into DNS packet, there are some methods to change the format of domain and IP Address and they can be understood by user.

For Example:

domain: 6 北邮 6 教育 6 中国 0 → 北邮. 教育. 中国

IP: c0a86462 ——> 192.168.100.98

The flow chart of function module is:

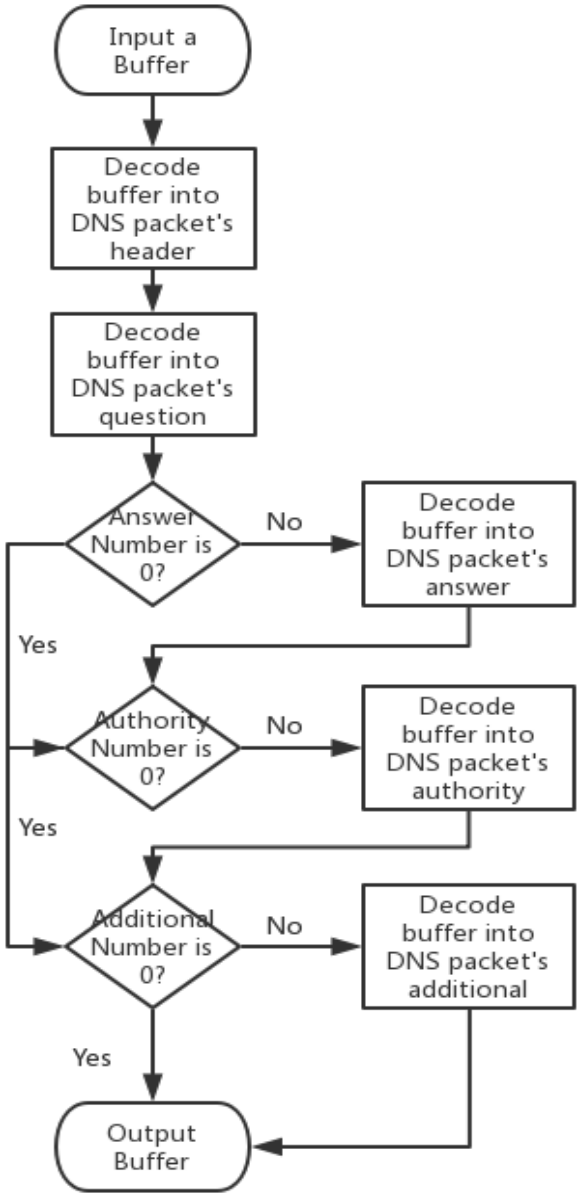


Figure 9. Decode Packet

4.4. File Operation

(1) File Reading

The function is used to get the Resource Record by checking the domain name and type. In the TLD and Root server, it only need to match the Second Level Domain or Top Level Domain, so we have to write a method to split the domain name.

For Example:

In the Root server: **北邮. 教育. 中国 → 中国**

In the TLD server: **北邮. 教育. 中国 → 教育. 中国**

The flow chart of function module is:

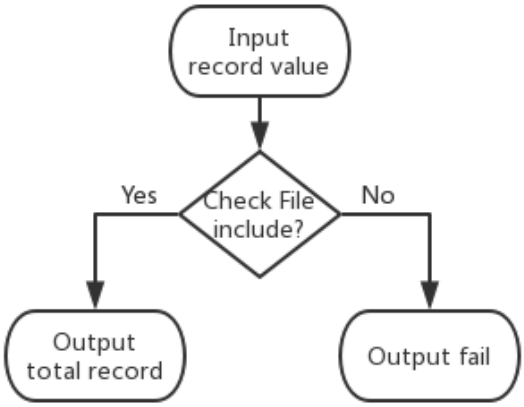


Figure 10. File Reading

(2) File Writing

The function is used to write the resource record into the cache of the Local Server' when get the correct record.

The flow chart of function module is:

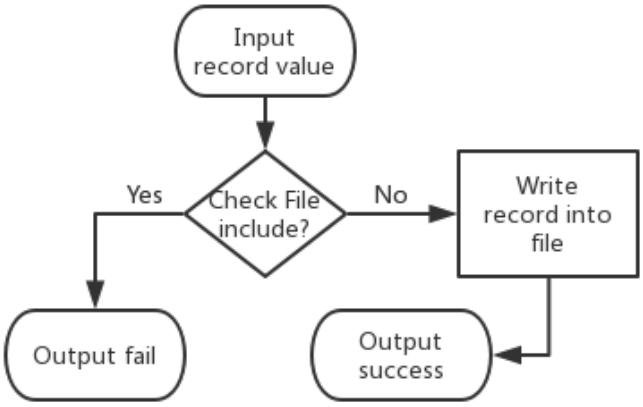


Figure 11. File Writing

5. Results

5.1. Trace and Display Time in Command Line (A, CNAME, MX record)

(1) A Record

When you enter value that domain name is “清华.教育.中国” and query type is “A”, system will give you the IP of the domain name. And the trace of querying will be printed on Local Server (show in the Figure 12,13).

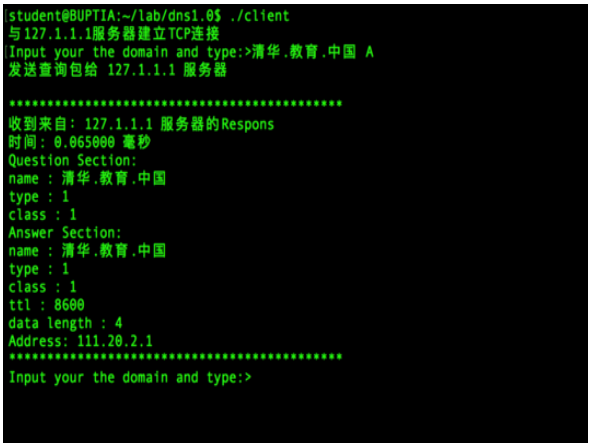


Figure 12. Details of A Record at Client

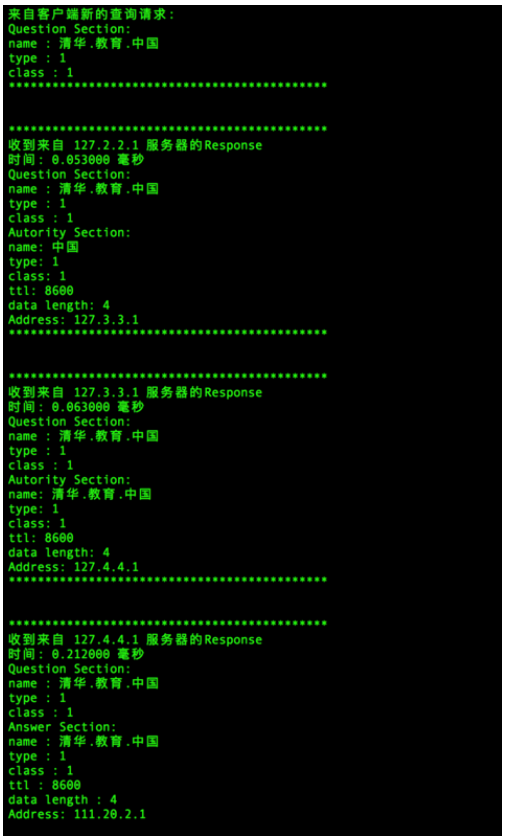


Figure 13. Trace of A Record at Local Server

(2) CNAME Record

When you enter value that domain name is “清华.教育.中国” and query type is “CNAME”, system will give you the Primary Name of the domain name. And the trace of querying will be printed on Local Server (show in the Figure 14,15).

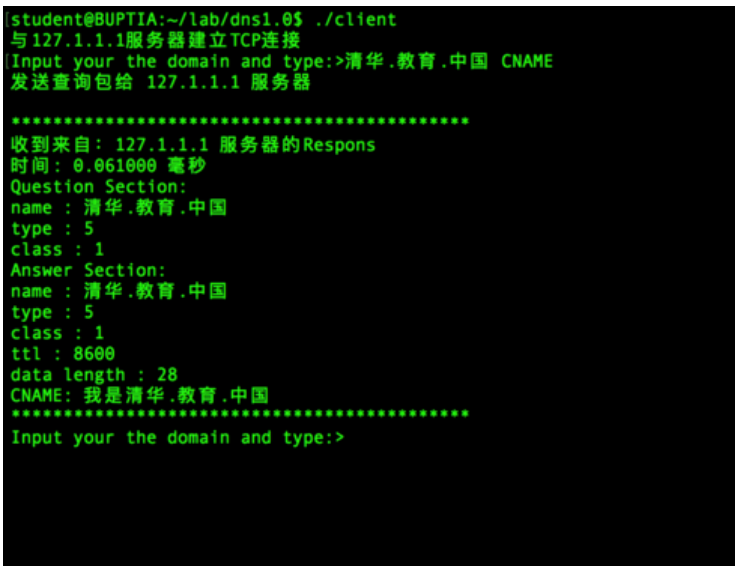


Figure 14. Details of CNAME Record at Client

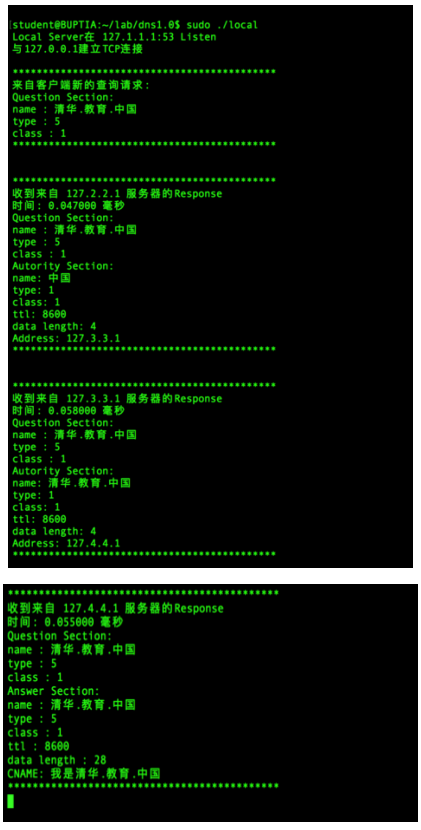


Figure 15. Trace of CNAME Record at Local Server

(3) MX Record

When you enter value that domain name is “清华.教育.中国” and query type is “MX”, system will give you the IP of the Mail Server of the domain name. And the trace of querying will be printed on Local Server (show in the Figure 16,17).

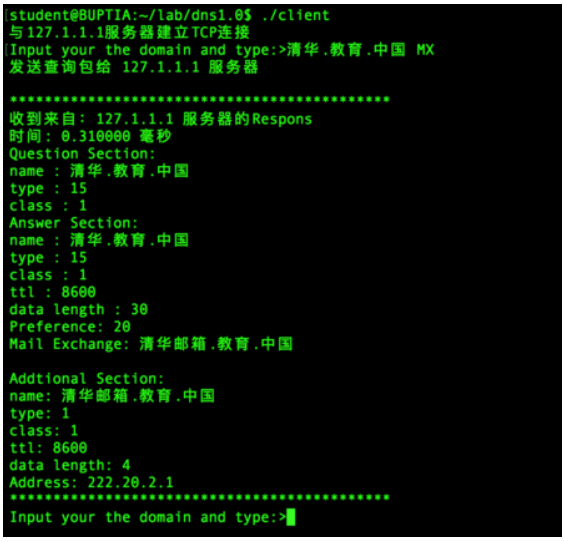


Figure 16. Details of MX Record at Client

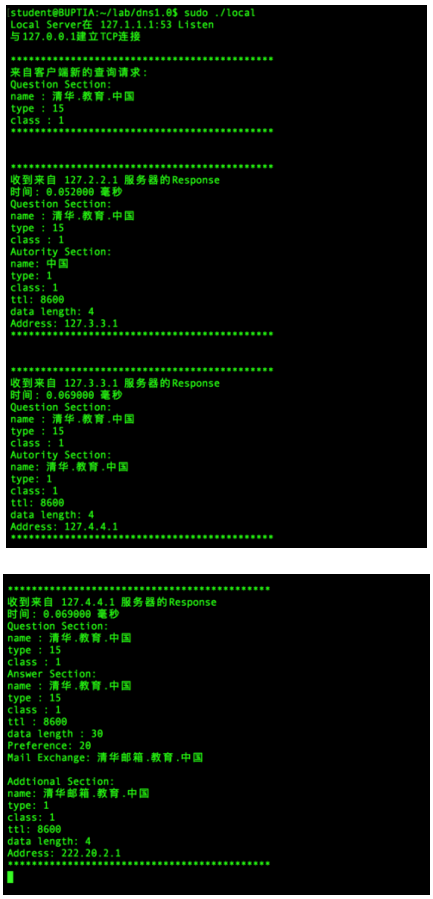


Figure 17. Trace of MX Record at Local Server

5.2. Result in Wireshark

(1) A Record

In the Wireshark, we can look through the path of querying, and check the final response from Local Server which IP is 127.1.1.1, the response packet's DNS flag is 0x8000 which means it is a standard response. Also the IP of domain will put into Answer Section of DNS packet. However, during the process of querying, if a server doesn't include the IP of the domain name, it will put next domain server's IP into Authority Section (show in the Figure 18,19).

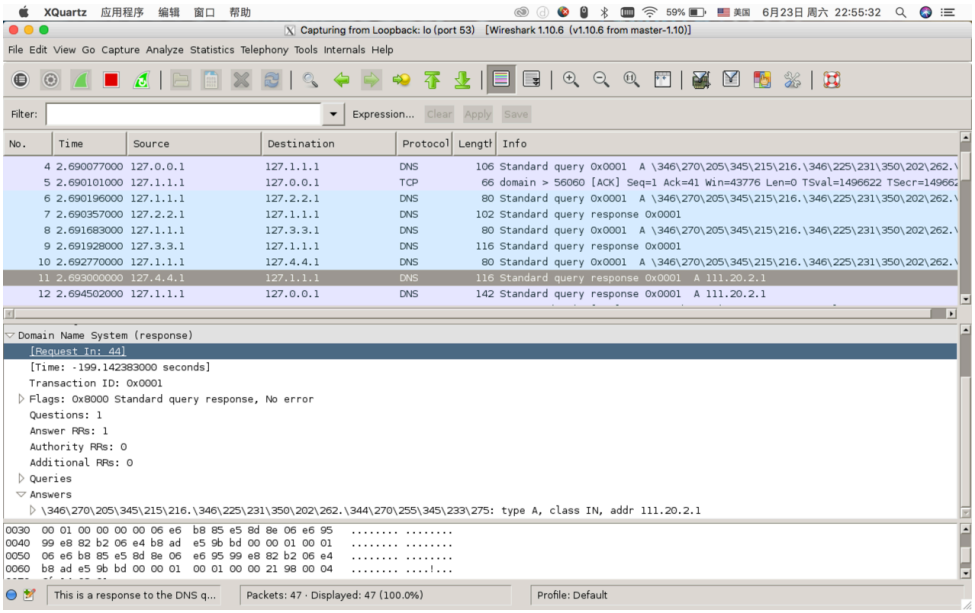


Figure 18. A Record Path in Wireshark

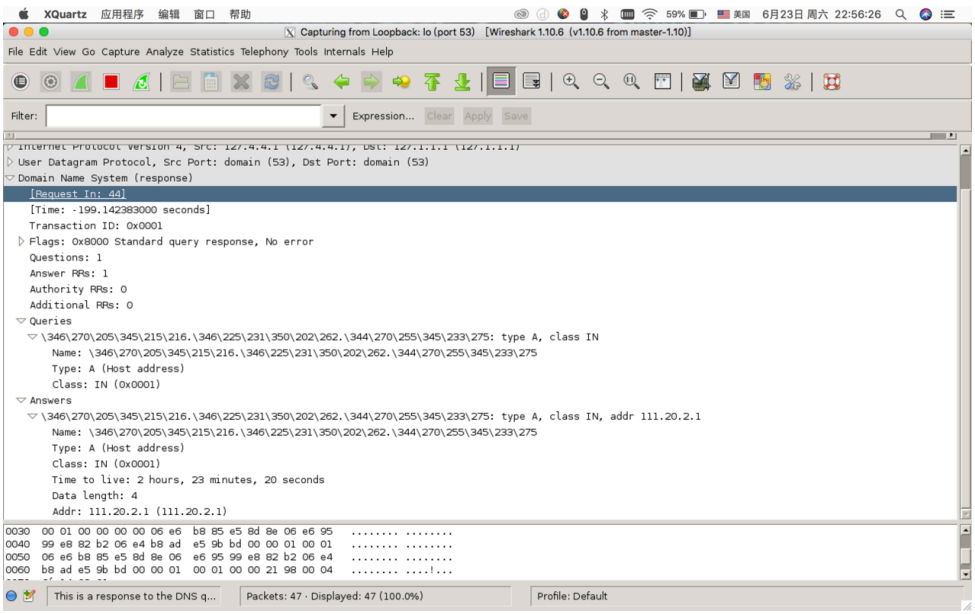


Figure 19. Information of A Record Answer Packet in Wireshark

(2) CNAME Record

In the Wireshark, we can look through the path of querying, and check the final response from Local Server which IP is 127.1.1.1, the response packet's DNS flag is 0x8000 which means it is a standard response. Also the Primary Domain of domain will put into Answer Section of DNS packet. However, during the process of querying, if a server doesn't include the IP of the domain name, it will put next domain server's IP into Authority Section (show in the Figure 20,21).

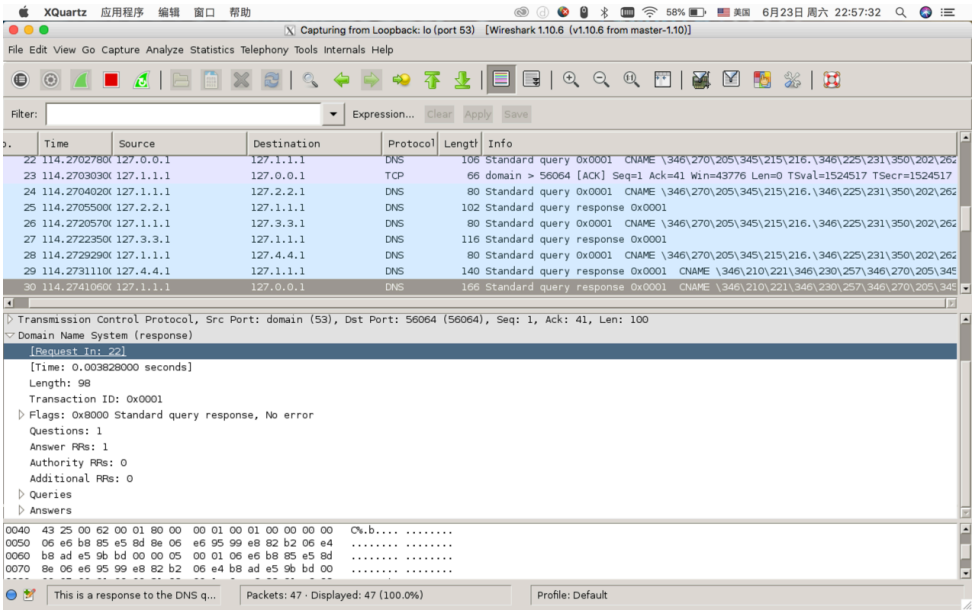


Figure 20. CNAME Record Path in Wireshark

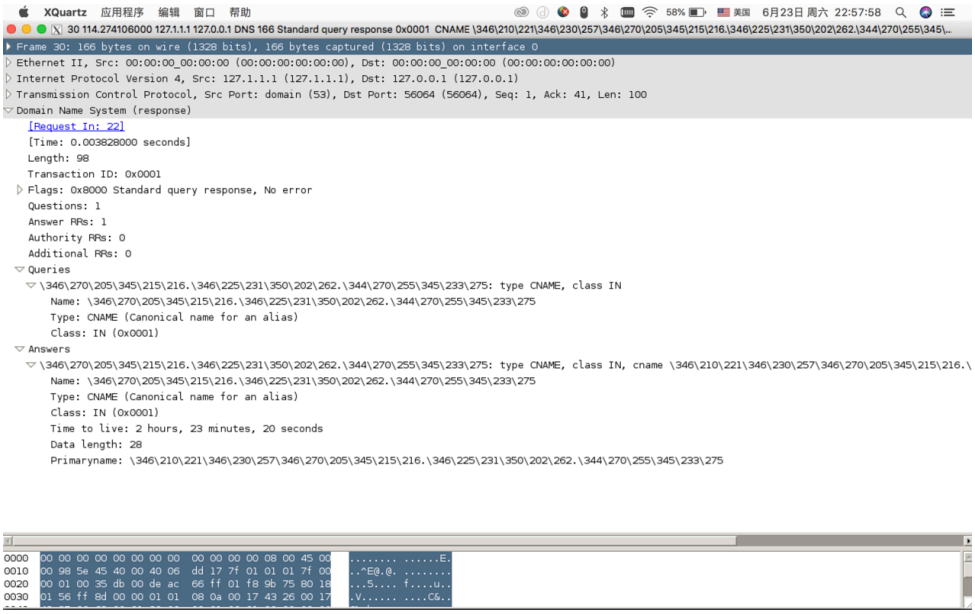


Figure 21. Information of CNAME Record Answer Packet in Wireshark

(3) MX Record

In the Wireshark, we can look through the path of querying, and check the final response from Local Server which IP is 127.1.1.1, the response packet's DNS flag is 0x8000 which means it is a standard response. Also the Mail Exchange Domain of the domain will put into Answer Section of DNS packet, the IP of Mail Exchange Domain will put into Additional. However, during the process of querying, if a server doesn't include the IP of the domain name, it will put next domain server's IP into Authority Section (show in the Figure 22,23).

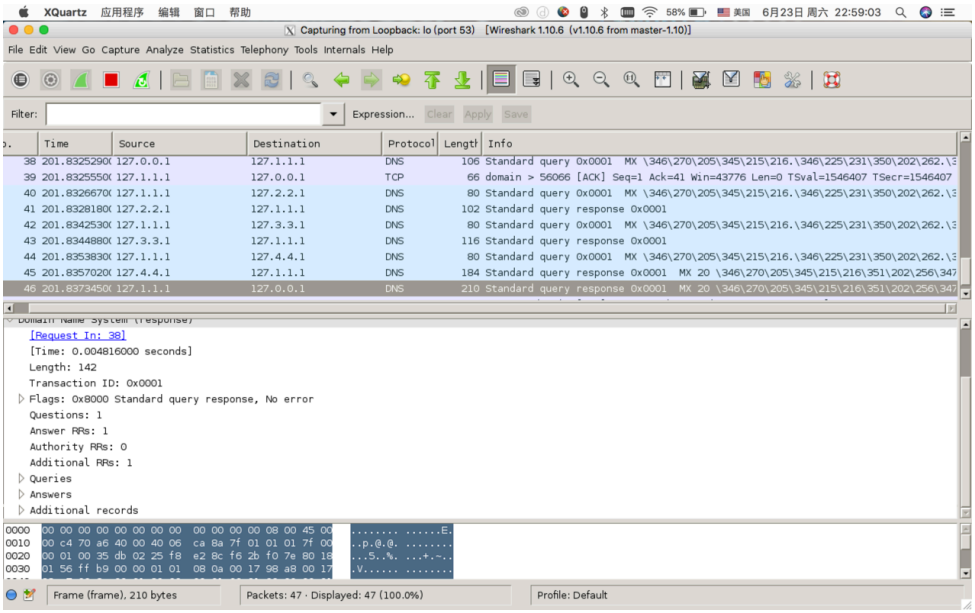
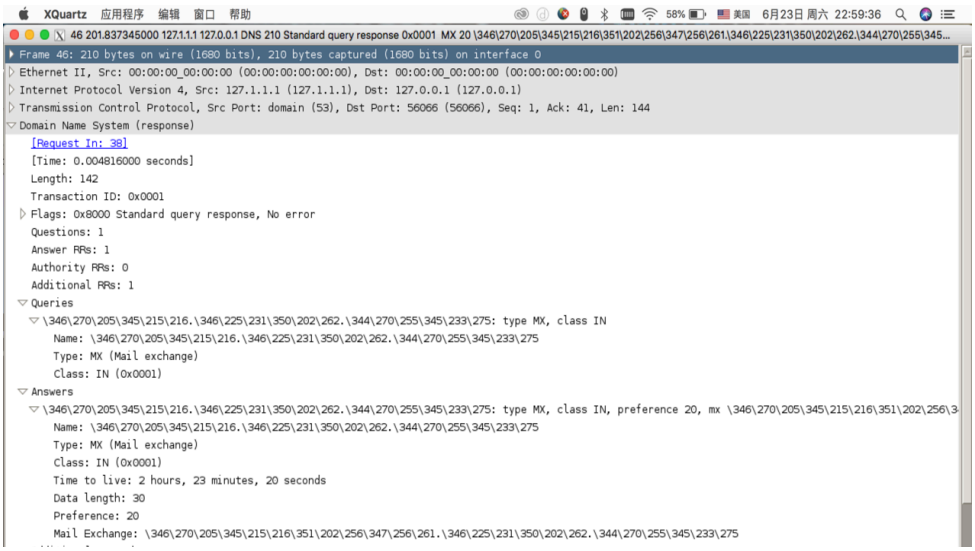


Figure 22. MX Record Path in Wireshark



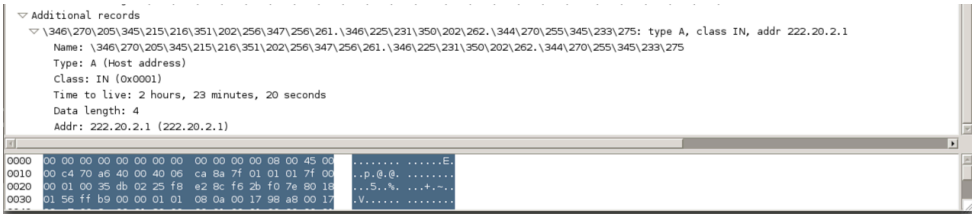


Figure 23. Information of MX Record Answer Packet in Wireshark

5.3. Result of Error Query (no domain name)

Demonstration of A Record

(1) Error at Root Server

When the domain does not exist, the final response packet's flag will be set to 0x8003. And the last query will end at Root Server. The response packet from Local Server which IP is 127.1.1.1 not include Answer Section (show in the Figure 24,25).

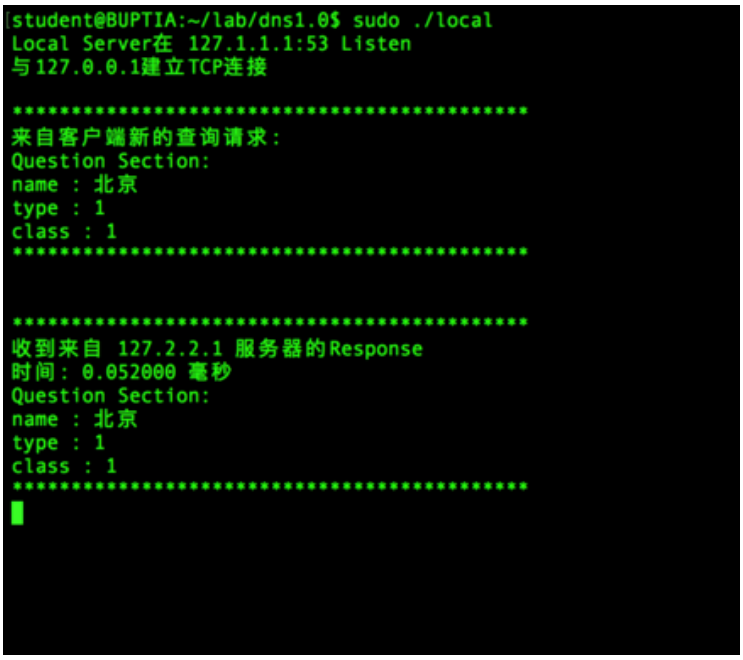


Figure 24. Trace of Error at Root

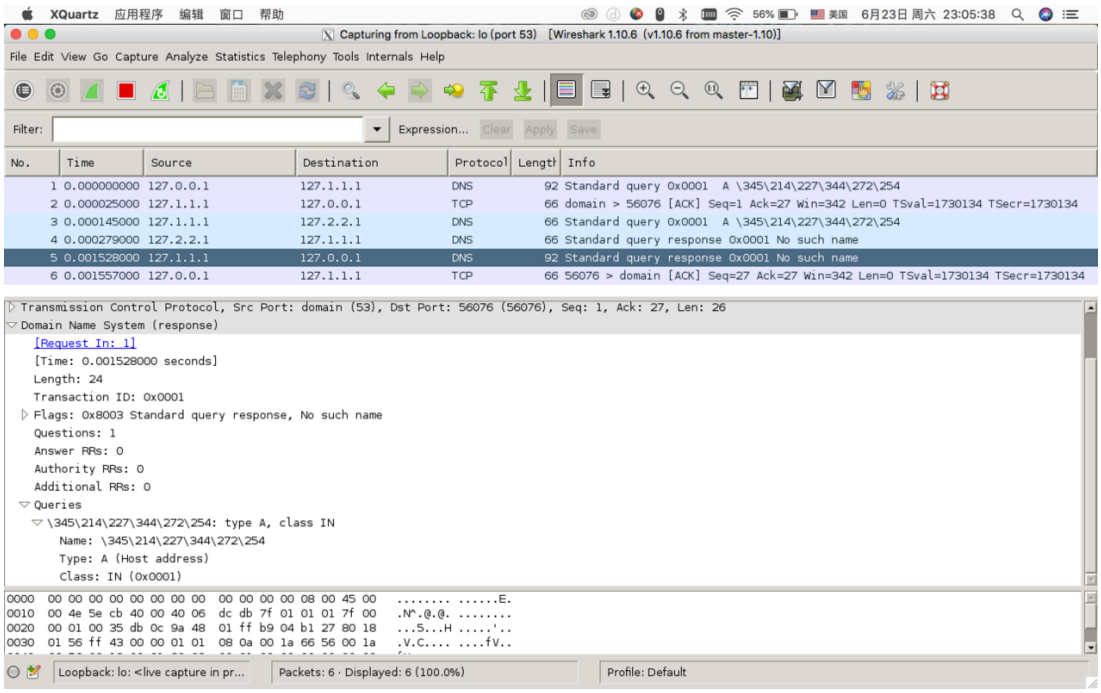


Figure 25. Error Path of Root in Wireshark

(2) Error at TLD Server

When the domain does not exist, the final response packet's flag will be set to 0x8003. And the last query will end at TLD Server. The response packet from Local Server which IP is 127.1.1.1 not include Answer Section (show in the Figure 26,27).

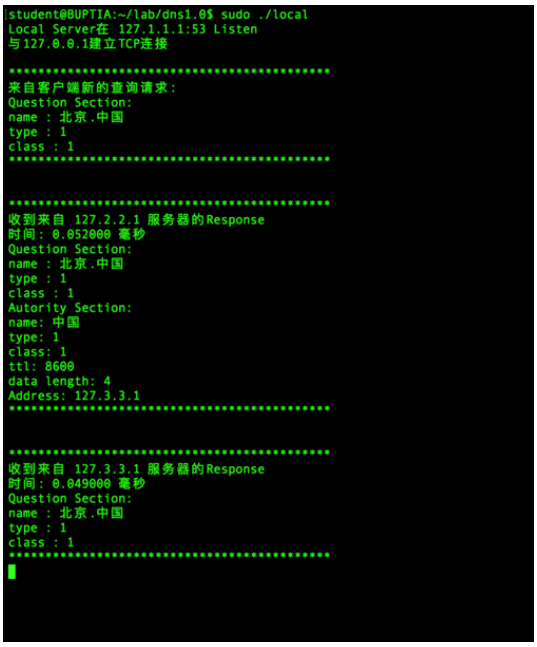


Figure 26. Trace of Error at TLD

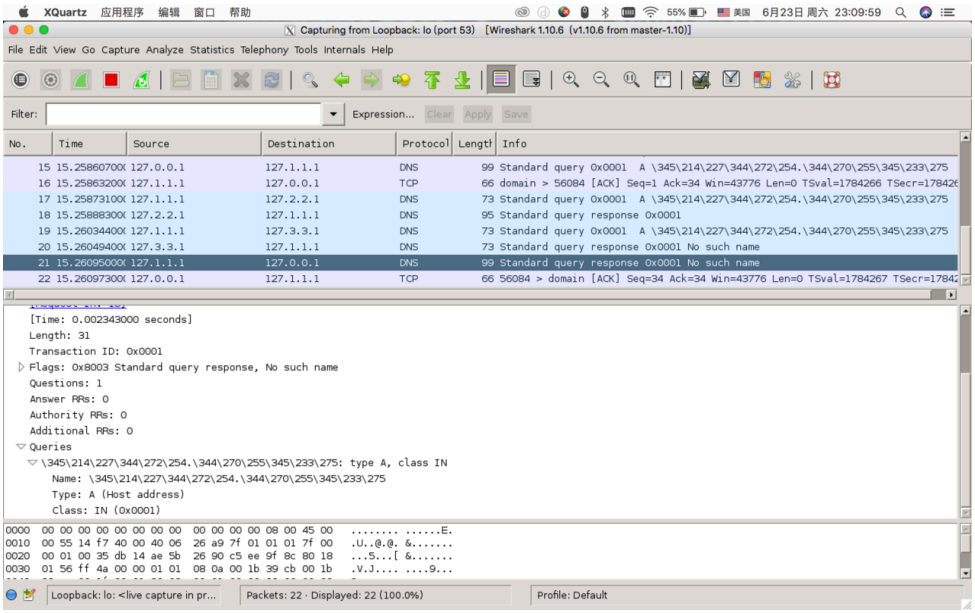


Figure 27. Error Path of TLD in Wireshark

(3) Error at 2LD Server

When the domain does not exist, the final response packet's flag will be set to 0x8003. And the last query will end at 2LD Server. The response packet from Local Server which IP is 127.1.1.1 not include Answer Section (show in the Figure 28,29).

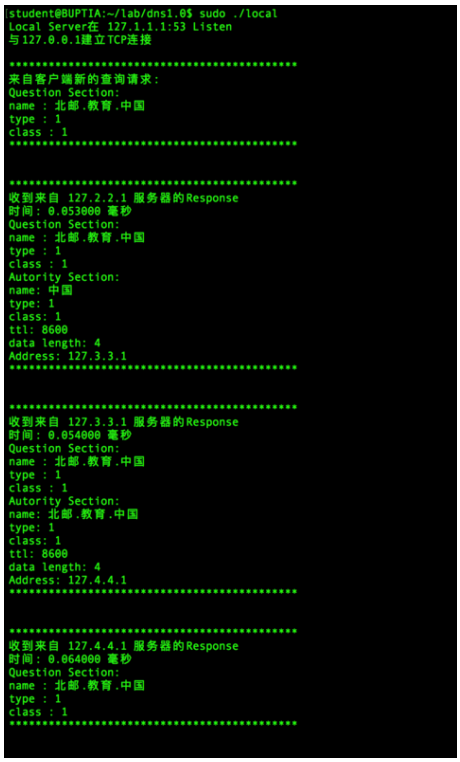


Figure 28. Trace of Error at 2LD