DESIGN AND IMPLEMENTATION OF A DNS RELAY

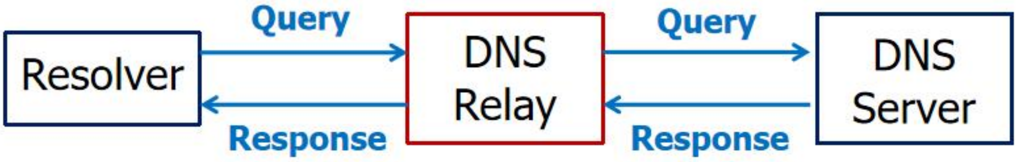
HAN ZHUOHANG (2019213419)
LIU SENDONG (2019213420)
LIAO HAOTIAN (2019213418)

Date: 2022/7/6

Index

1. Requirement analysis

As shown in the figure below:



The main work of our project is to implement a DNS relay server that receives DNS queries from DNS client and forward them to a given DNS server, and receives DNS responses from DNS server and forward them to the client.

The DNS relay will maintain a local database to store the domain name and corresponding IPv4 & IPv6 address and also their TTL(Time To Live). The database has its mechanism to delete the expiring data. And also we should consider the probability of several client query the DNS Relay at the same time. So several locks should be attached to the database operations. When a new query is received from the client, the database will be checked, and there are three cases below:

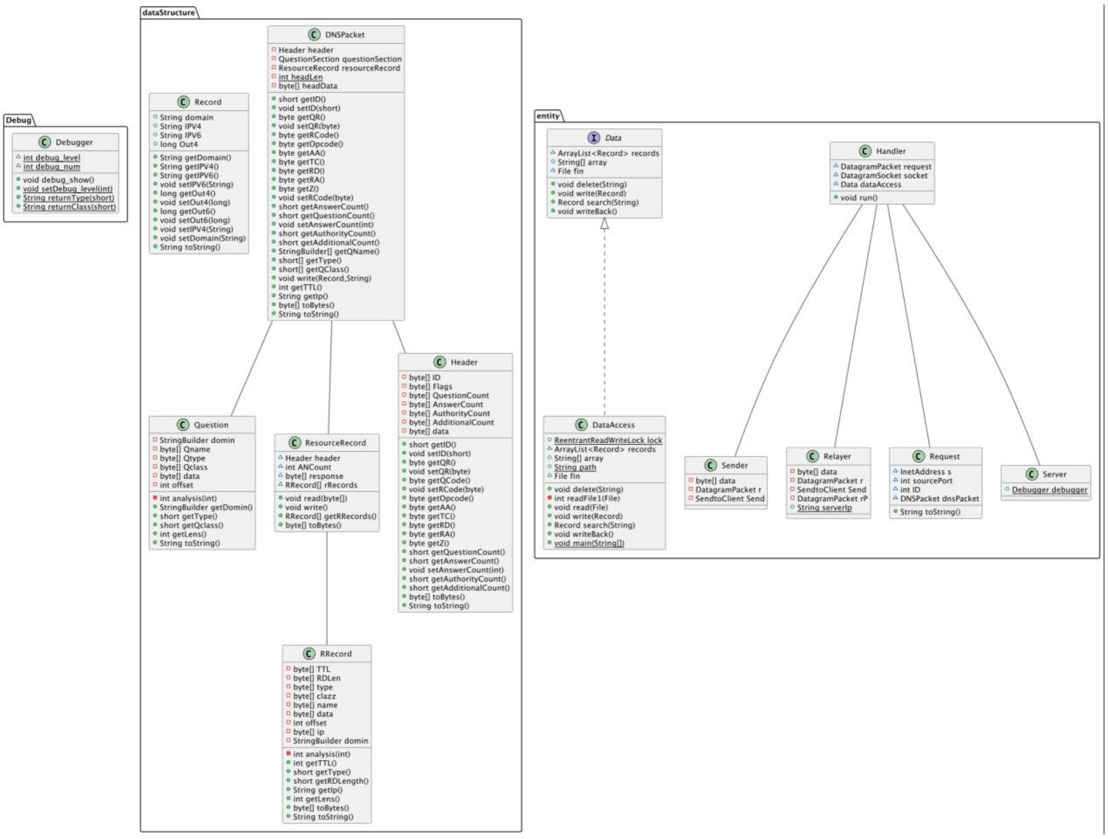
Case 1: Corresponding IP of a given domain name is found in the database. The IP is 0.0.0.0 indicating that the target domain is blocked. The relay will send back a response whose RCODE is set to 0011(The name in the query does not exist) to the client.

Case 2: Corresponding IP of a given domain name is found in the database. But the IP is not 0.0.0.0, which indicates that the existed target domain is valid. In addition to this, we will also calculated the TTL in the database to see whether the data is expired. If it is expired, the database will flush and may enter the case 3. If it isn't expired, the relay will send back a response whose RCODE is set to 0000(no error condition) to the client with the answer containing the IPv4 or IPv6 address.

Case 3: Corresponding IP of a given domain name is not found in the database. Relay forwards query to the local DNS server and forwards response from the server to the client.

Other types of queries excluding type A and AAAA will be directly forwarded to the server.

This is our entire UML Design:



2. Decomposition of functional modules

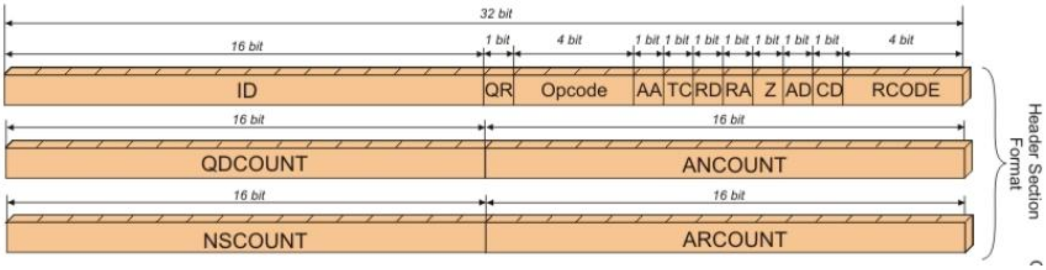
2.1. Local Database

The database is a combination of record.txt, read (File fin) function and the DataAccess class. In the txt file, we store the domain name with their IPv4 address and IPv6 address, and also store their expiration time. When the constructor of the DataAccess class is called, the program will read the record.txt, and store the records in the txt file into a tempt array. Then we create an entity class--record, which store an object just as one line in txt, but the difference is the expiration time in the txt is changed to ttl in record.

The records in the array will be stored in an Arraylist, and all the delete, write and modify operation will only change the record in the Arraylist. In the end of the execution the writeback method will be called to renew the text file. In this way we cut down the I/O time. And to realize the multi-thread access, locks should be attached to the database operation. In the read operation we use write lock(because this operation will change the Arraylist). And in write and delete operation we also use a write lock. In the search method we use read lock. So using the read-write, multiple threads can read at the same time, and only one thread can change the database at one time.

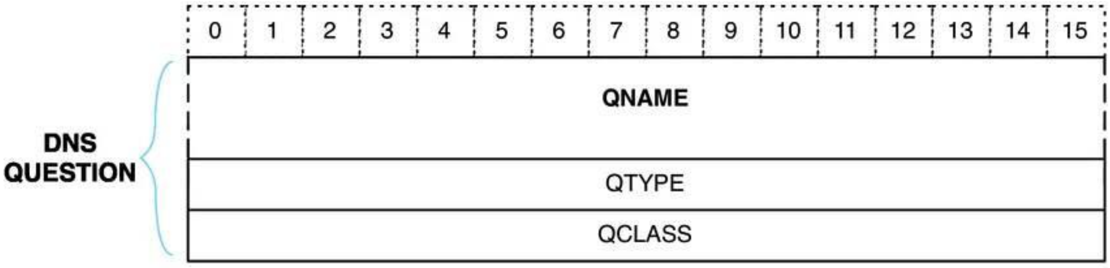
2.2. DNS Packet

Header: The header contains the information as follow:



In our program, we use the Header class to decode & encode the header part. We have set method to set the RCODE and QR part, so if the IP is blocked, the RCODE will be set to 0011 and the QR code will set to 1, which means response.Then the frame will be send back to client. And if not blocked, the RCODE will be set to 0000.

Question: Question analysis part is consist of Question.java and QuestionSection.java.



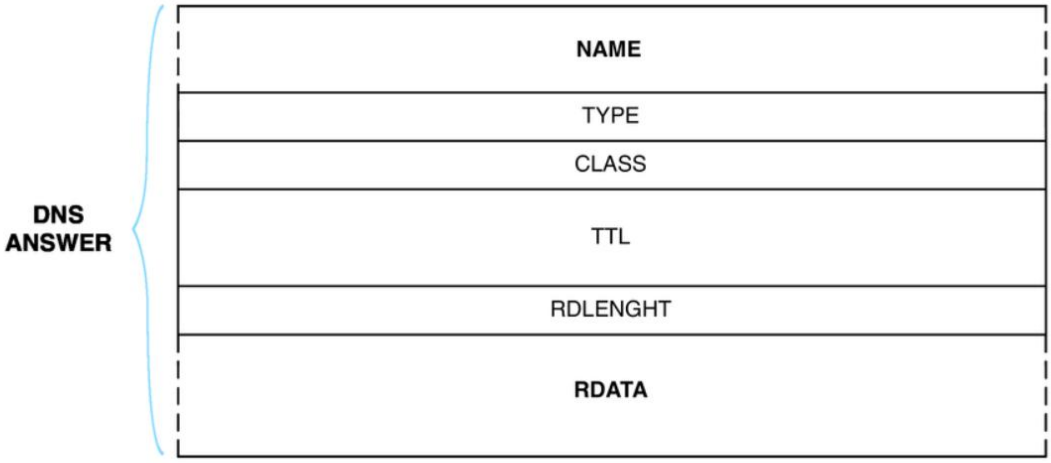
QNAME: QNAME has a unknown length. Use &amp; 0xff to get the length of the domain name field, and then read the entire field

QTYPE: 2bytes. 0x01:A; 0x28:AAAA; 0x12:PTR

QCLASS:2bytes.0x01:IN 0x02:CS 0x03:CH 0x04:HS

Resource Record is consist of RRecord and ResourceRecord class. RRecord will parse one single resource recorde. It will provide method to get the TTL and IP address.

Resource Record will handle the whole answer part which may contain many resource records. It will provide method to parse all the resource record by calling the RRecord. It will pick the first IP address and TTL of type A or AAAA. Besides, it provide a write method to generate a response based on cache information.



Next, I am going to explain the filed.

Name filed has two possibility. One is to write pointer like 0xC00C. Another one is to write the domain name just as question section. For example it may write 3www4bing3com. Type and class both are two bytes just as defined in question section. TTL will take 4 bytes and its unit is millisecond. And next is resource record length that will take 2 bytes. It will say the length of the real data. Resource data filed has variable length.

2.3. DNS Relay

DNS relay module is consist of Server, Handler, Relayer, Sender and Send to Client class.

Server will listen to port 53 and start a new thread of handler if a requirement comes. Handler will transmit the packet to relayer or sender based on whether domain name is cached in the local database. Relayer will relay the packet to the default domain server and relay the packet received to client by calling send to client class. Sender will write the packet based on the cached information and send to client.

In summary, this module will receive the request from client and start a new thread to handle. Then it will relay the request if domain name isn’t cached or it will directly send response to client if request domain name is cached.

Check local database (three cases):

- Blocked(0.0.0.0(IPv4) or 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0(IPv6)): Set RCODE=0011 for both A(1) and AAAA(28) response.
- cached (IPv4 and IPv6): Generate IPv4 respond for A(1) request or IPv6 response for AAAA(28) request.
- no record: Forward request to DNS.

Type without A or AAAA will be forwarded directly.

Receive from DNS and generate a response, and forward to the client.

Cache the A and AAAA response in the local database.

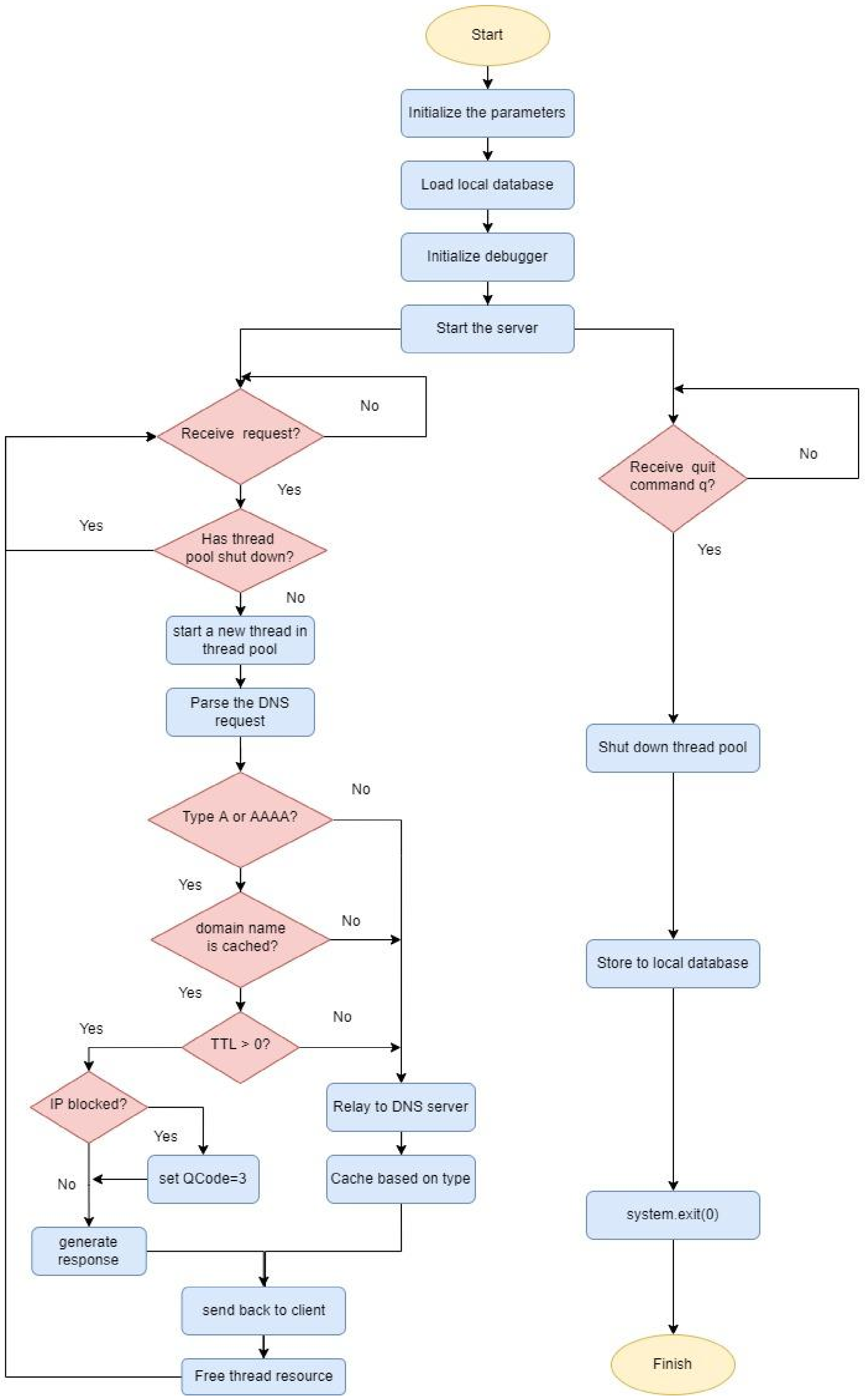
2.4. Debug

Debug level 0: no output.

Debug level 1: Time, ID, IP, DominName, Type, Class, RECV/SEND message

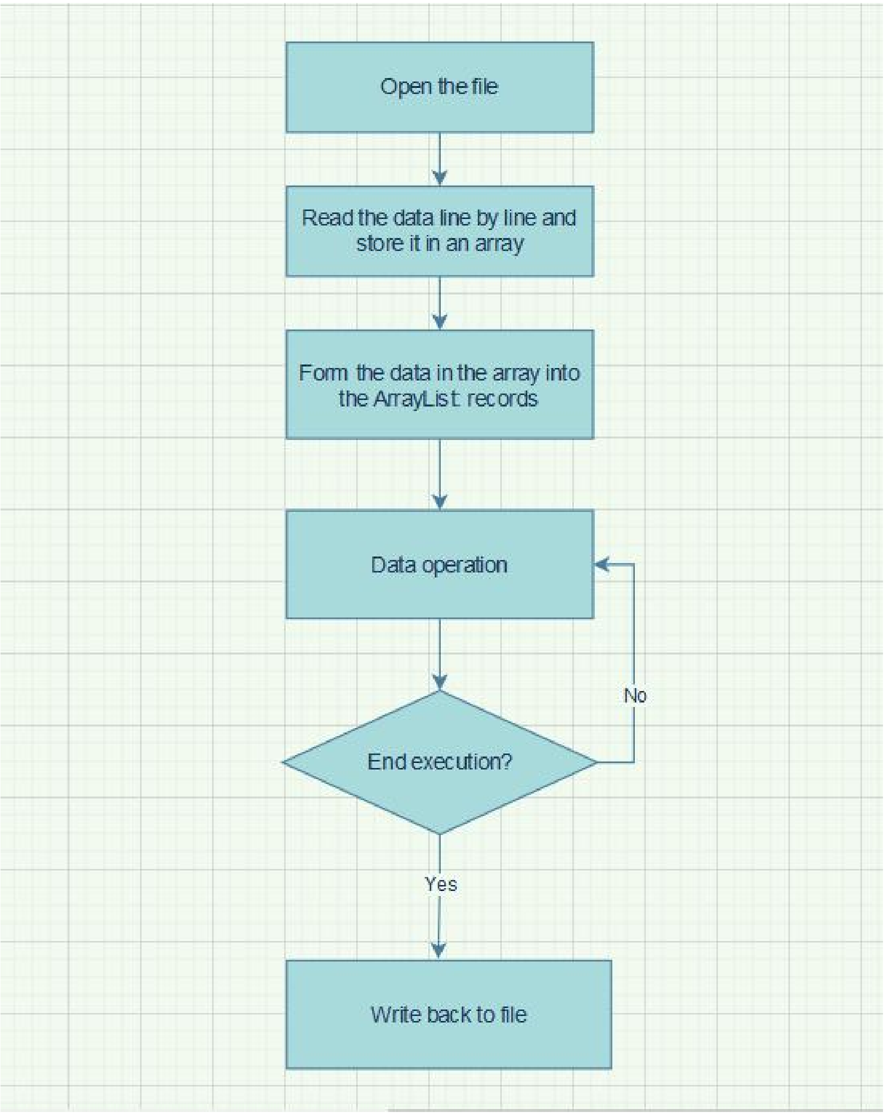
Debug level 2: Time, ID, IP, DominName, Type，Class, RECV/SEND message and Data, QR, Opcode, AA, TC, RD, Rcode, RA, Z, Question count, Answer count, Authority count,Additional count

3. Overall flow chart

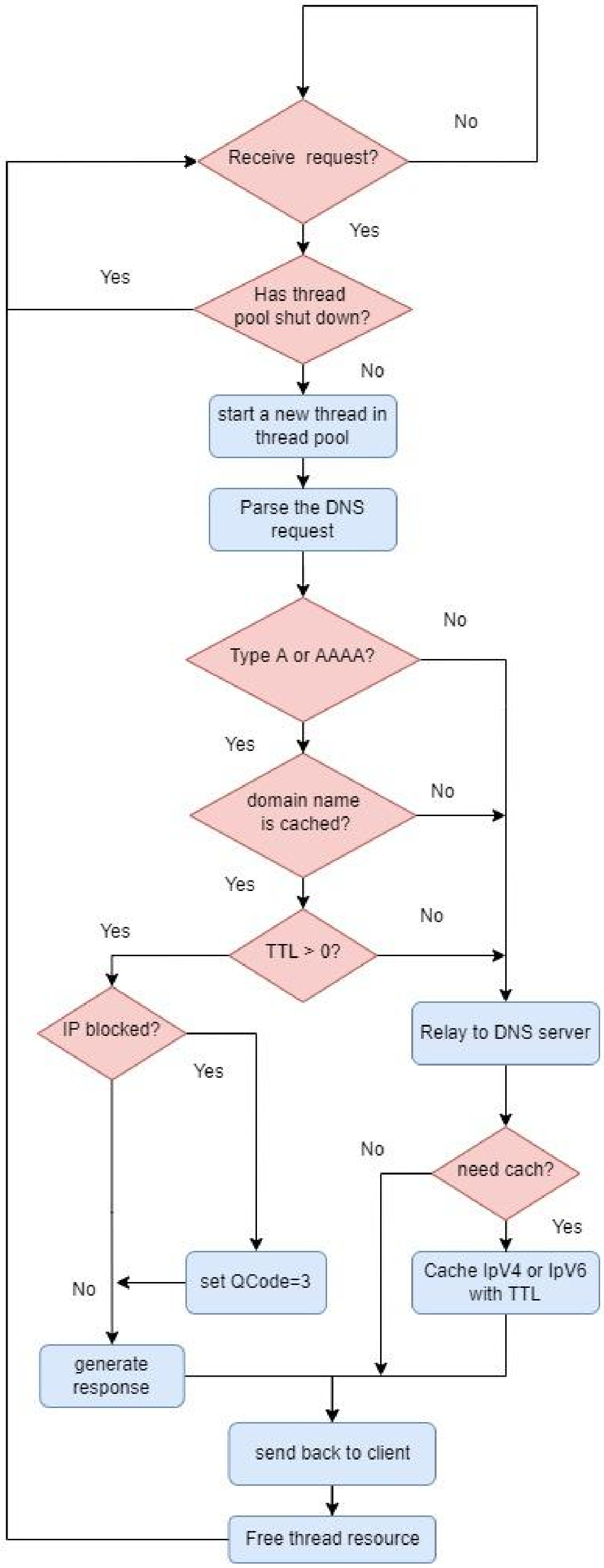


4. Flow charts in each module

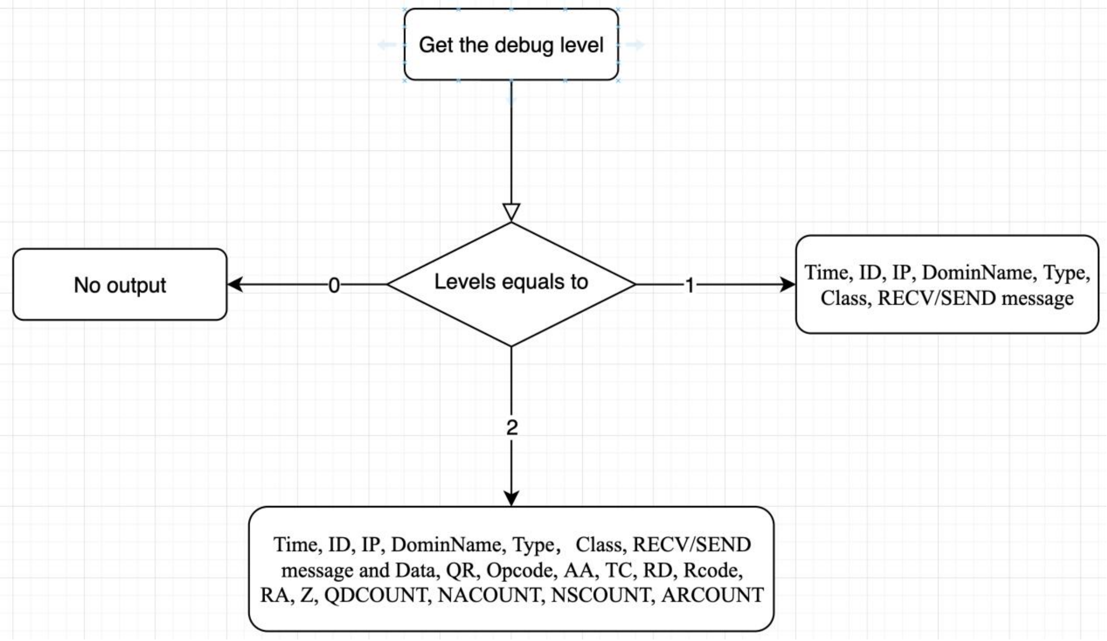
4.1. Load the database


Open the file
Read the data line by line and store it in an array
Form the data in the array into the ArrayList records
Data operation
End execution?
No
Yes
Write back to file

4.2. DNS Relay

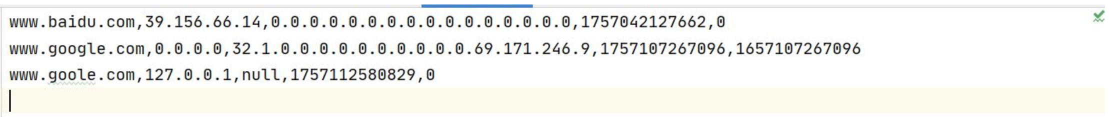

Receive request?
No
Yes
Has thread pool shut down?
Yes
No
start a new thread in thread pool
Parse the DNS request
Type A or AAAA?
No
Yes
domain name is cached?
No
Yes
TTL > 0?
Yes
No
IP blocked?
Yes
No
Relay to DNS server
need cach?
No
Yes
Cache IpV4 or IpV6 with TTL
set QCode=3
generate response
send back to client
Free thread resource

4.3. Debug

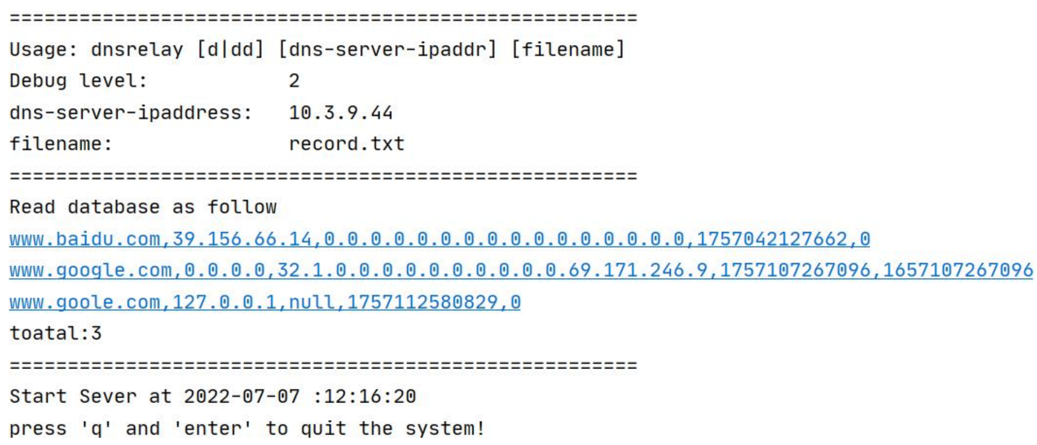

Get the debug level
Levels equals to
0
No output
1
Time, ID, IP, DominName, Type, Class, RECV/SEND message
2
Time, ID, IP, DominName, Type, Class, RECV/SEND message and Data, QR, Opcode, AA, TC, RD, Rcode, RA, Z, QDCOUNT, NACOUNT, NSCOUNT, ARCOUNT

5. Testing and results

local database:


```
www.baidu.com,39.156.66.14,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0,1757042127662,0
www.google.com,0.0.0.0,32.1.0.0.0.0.0.0.0.0.0.0.69.171.246.9,1757107267096,1657107267096
www.goole.com,127.0.0.1,null,1757112580829,0
```

Use ***nslookup [domain] 127.0.0.1*** to test five cases in the requirement. And check the packet information with Wireshark. Catch packet from both backloop and Ethernet3 with ***filter : dns.*** Launch the program with the highest debug level:

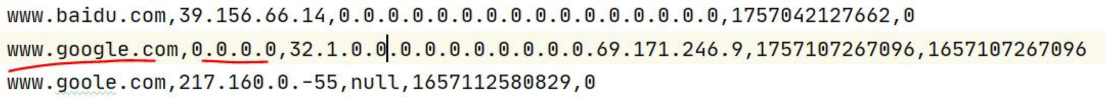


```
=====================================================
Usage: dnsrelay [d|dd] [dns-server-ipaddr] [filename]
Debug level:              2
dns-server-ipaddress:     10.3.9.44
filename:                 record.txt
=====================================================
Read database as follow
www.baidu.com,39.156.66.14,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0,1757042127662,0
www.google.com,0.0.0.0,32.1.0.0.0.0.0.0.0.0.0.0.69.171.246.9,1757107267096,1657107267096
www.goole.com,127.0.0.1,null,1757112580829,0
toatal:3
=====================================================
Start Sever at 2022-07-07 :12:16:20
press 'q' and 'enter' to quit the system!
```

5.1. Case 1: Send a request with IP blocked.

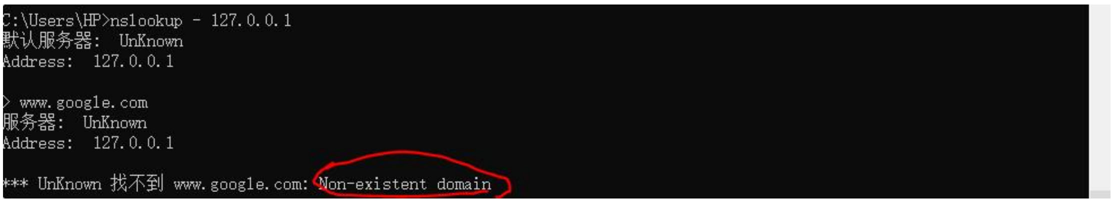
Test 1: IpV4 is 0.0.0.0

First, let’s see the local database. We can see www.google.com map to IpV4 = 0.0.0.0.



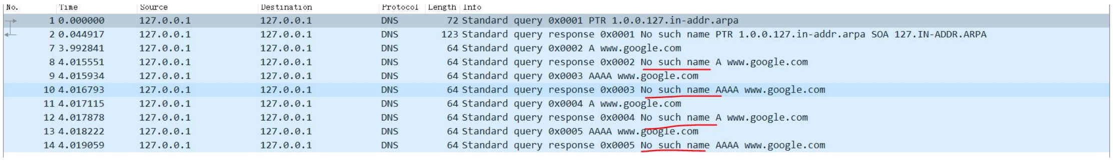
```
www.baidu.com,39.156.66.14,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0,1757042127662,0
www.google.com,0.0.0.0,32.1.0.0.0.0.0.0.0.0.0.0.69.171.246.9,1757107267096,1657107267096
www.goole.com,217.160.0.-55,null,1657112580829,0
```

Next let’s using ns-lookup to look for www.google.com. We can see the address is blocked.



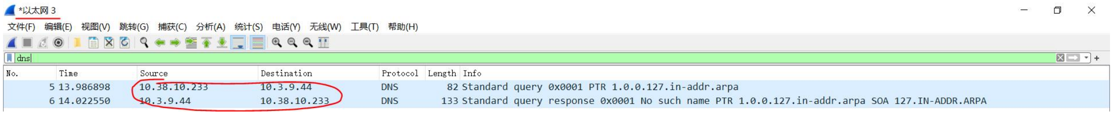
```
C:\Users\HP>nslookup - 127.0.0.1
默认服务器:  UnKnown
Address:  127.0.0.1

> www.google.com
服务器:  UnKnown
Address:  127.0.0.1

*** UnKnown 找不到 www.google.com: Non-existent domain
```

Next let’s look at the result of wire-shark.



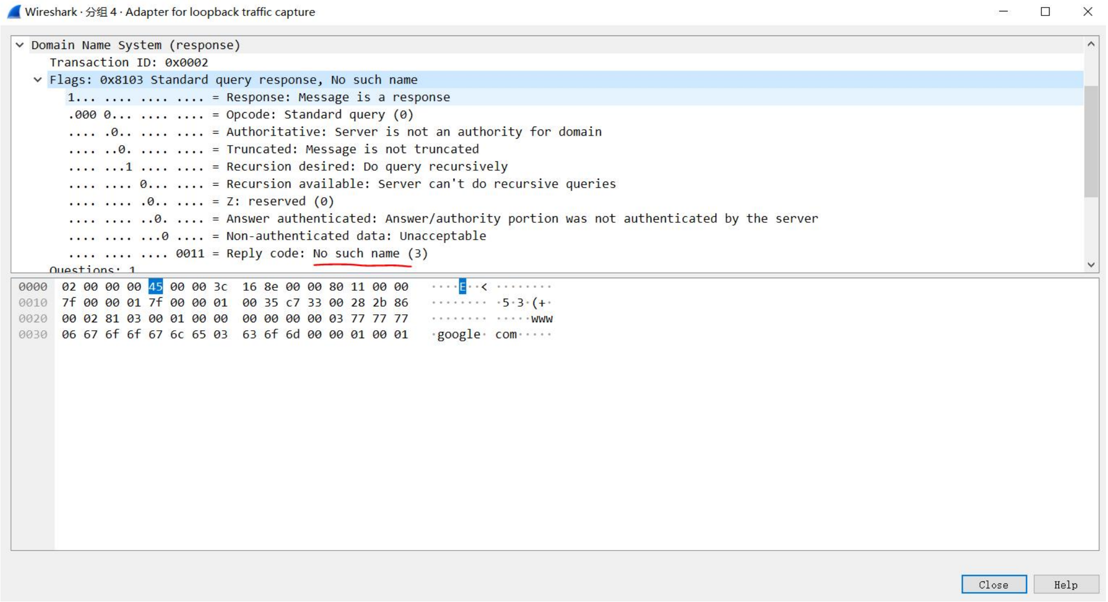
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	DNS	72	Standard query 0x0001 PTR 1.0.0.127.in-addr.arpa
2	0.044917	127.0.0.1	127.0.0.1	DNS	123	Standard query response 0x0001 No such name PTR 1.0.0.127.in-addr.arpa SOA 127.IN-ADDR.ARPA
7	3.992841	127.0.0.1	127.0.0.1	DNS	64	Standard query 0x0002 A www.google.com
8	4.015551	127.0.0.1	127.0.0.1	DNS	64	Standard query response 0x0002 No such name A www.google.com
9	4.015934	127.0.0.1	127.0.0.1	DNS	64	Standard query 0x0003 AAAA www.google.com
10	4.016793	127.0.0.1	127.0.0.1	DNS	64	Standard query response 0x0003 No such name AAAA www.google.com
11	4.017115	127.0.0.1	127.0.0.1	DNS	64	Standard query 0x0004 A www.google.com
12	4.017878	127.0.0.1	127.0.0.1	DNS	64	Standard query response 0x0004 No such name A www.google.com
13	4.018222	127.0.0.1	127.0.0.1	DNS	64	Standard query 0x0005 AAAA www.google.com
14	4.019059	127.0.0.1	127.0.0.1	DNS	64	Standard query response 0x0005 No such name AAAA www.google.com

From this result, we can see they are all no such name, which means they are blocked.


*以太网 3
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
dns

```
No.   Time        Source         Destination    Protocol  Length Info
    5 13.986898   10.38.10.233   10.3.9.44      DNS          82 Standard query 0x0001 PTR 1.0.0.127.in-addr.arpa
    6 14.022550   10.3.9.44      10.38.10.233   DNS         133 Standard query response 0x0001 No such name PTR 1.0.0.127.in-addr.arpa SOA 127.IN-ADDR.ARPA
```

From this we can see it won’t block other type DNS request.

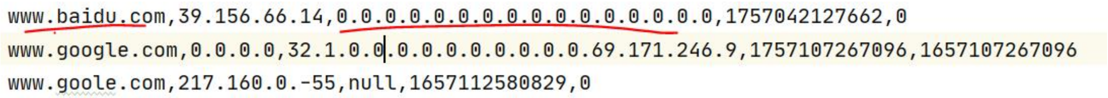


```
Domain Name System (response)
    Transaction ID: 0x0002
  Flags: 0x8103 Standard query response, No such name
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 0... .... = Recursion available: Server can't do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0011 = Reply code: No such name (3)
    Questions: 1

0000  02 00 00 00 45 00 00 3c  16 8e 00 00 80 11 00 00   ····E··< ········
0010  7f 00 00 01 7f 00 00 01  00 35 c7 33 00 28 2b 86   ········ ·5·3·(+·
0020  00 02 81 03 00 01 00 00  00 00 00 00 03 77 77 77   ········ ·····www
0030  06 67 6f 6f 67 6c 65 03  63 6f 6d 00 00 01 00 01   ·google· com·····
```

From this we can see reply code set to 3.

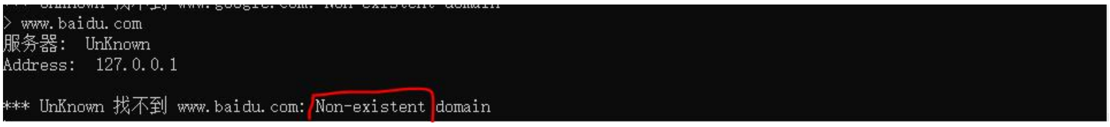
Test 2: IpV6 is 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0

First, let’s see the local database. www.baidu.com has blocked IpV6.



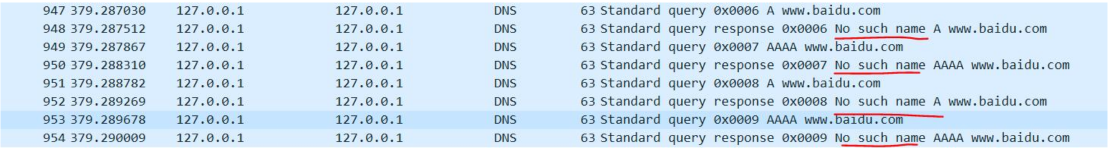
```
www.baidu.com,39.156.66.14,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0,1757042127662,0
www.google.com,0.0.0.0,32.1.0.0.0.0.0.0.0.0.0.0.69.171.246.9,1757107267096,1657107267096
www.goole.com,217.160.0.-55,null,1657112580829,0
```

Next we see result from ns-lookup. We can see it say it doesn’t exist.



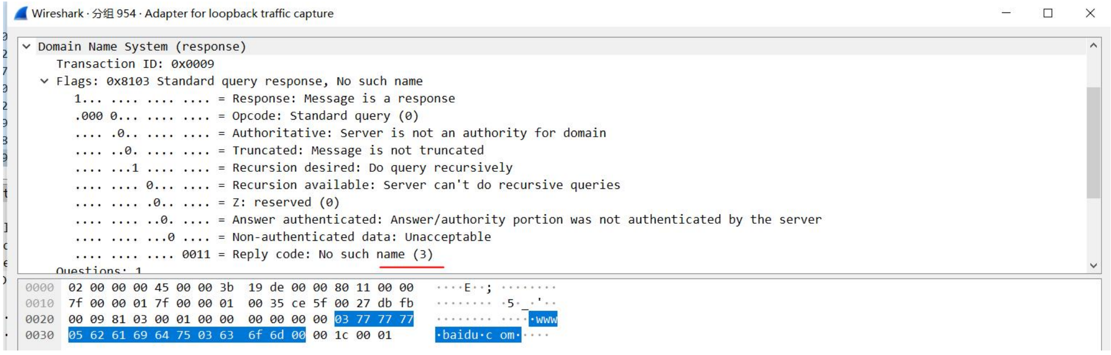
```
*** UnKnown 找不到 www.google.com: Non-existent domain
> www.baidu.com
服务器:  UnKnown
Address:  127.0.0.1

*** UnKnown 找不到 www.baidu.com: Non-existent domain
```

Last let’s see the result from wire-shark.



```
    947 379.287030    127.0.0.1        127.0.0.1        DNS        63 Standard query 0x0006 A www.baidu.com
    948 379.287512    127.0.0.1        127.0.0.1        DNS        63 Standard query response 0x0006 No such name A www.baidu.com
    949 379.287867    127.0.0.1        127.0.0.1        DNS        63 Standard query 0x0007 AAAA www.baidu.com
    950 379.288310    127.0.0.1        127.0.0.1        DNS        63 Standard query response 0x0007 No such name AAAA www.baidu.com
    951 379.288782    127.0.0.1        127.0.0.1        DNS        63 Standard query 0x0008 A www.baidu.com
    952 379.289269    127.0.0.1        127.0.0.1        DNS        63 Standard query response 0x0008 No such name A www.baidu.com
    953 379.289678    127.0.0.1        127.0.0.1        DNS        63 Standard query 0x0009 AAAA www.baidu.com
    954 379.290009    127.0.0.1        127.0.0.1        DNS        63 Standard query response 0x0009 No such name AAAA www.baidu.com
```

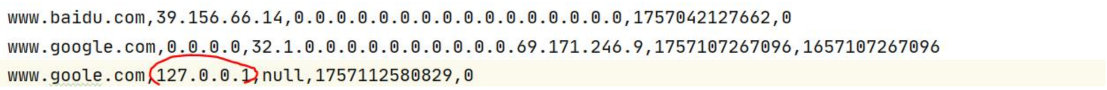
We can see reply code is 3.



```
Domain Name System (response)
    Transaction ID: 0x0009
  Flags: 0x8103 Standard query response, No such name
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 0... .... = Recursion available: Server can't do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0011 = Reply code: No such name (3)
    Questions: 1

0000  02 00 00 00 45 00 00 3b  19 de 00 00 80 11 00 00   ····E··; ········
0010  7f 00 00 01 7f 00 00 01  00 35 ce 5f 00 27 db fb   ········ ·5·_·'··
0020  00 09 81 03 00 01 00 00  00 00 00 00 03 77 77 77   ········ ·····www
0030  05 62 61 69 64 75 03 63  6f 6d 00 00 1c 00 01      ·baidu·c om·····
```

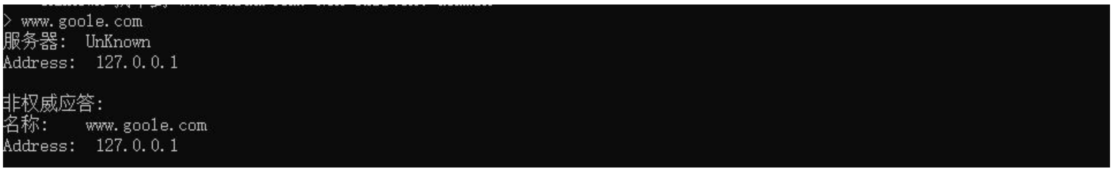
5.2. Case 2: Domain name in the local database.

First let’s see the local database. I change the IpV4 to 127.0.0.1 for www.goole.com.



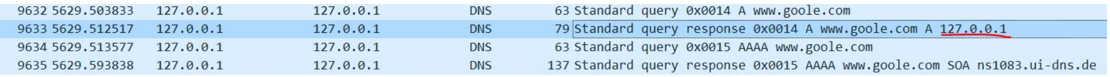
```
www.baidu.com,39.156.66.14,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0,1757042127662,0
www.google.com,0.0.0.0,32.1.0.0.0.0.0.0.0.0.0.0.0.0.0.69.171.246.9,1757107267096,1657107267096
www.goole.com,127.0.0.1,null,1757112580829,0
```

Next, let’s see result from ns-lookup. The result address is 127.0.0.1.

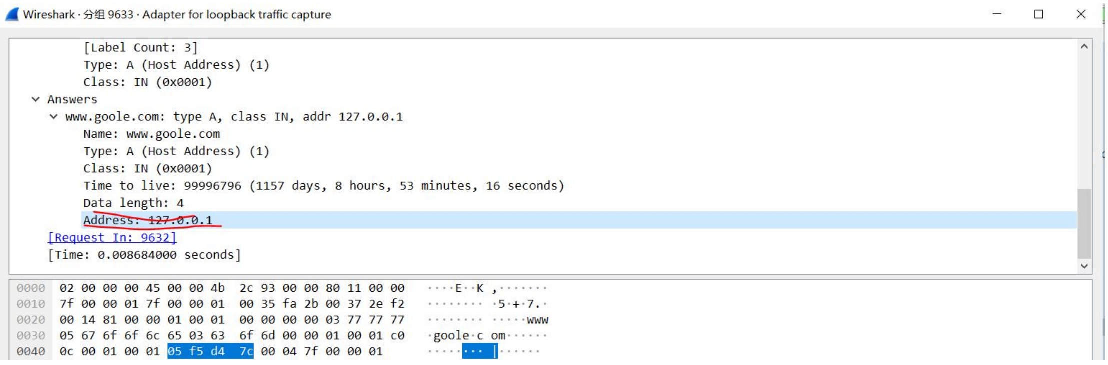


```
> www.goole.com
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    www.goole.com
Address:  127.0.0.1
```

Next, let’s see result from the wire-shark. We can see the 127.0.0.1 for type A. And as type AAAA isn’t in local database, we just forward request to DNS server and send received data back to client.



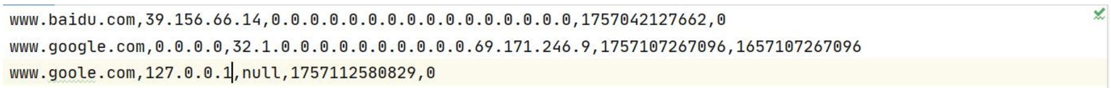
```
9632 5629.503833   127.0.0.1         127.0.0.1         DNS        63 Standard query 0x0014 A www.goole.com
9633 5629.512517   127.0.0.1         127.0.0.1         DNS        79 Standard query response 0x0014 A www.goole.com A 127.0.0.1
9634 5629.513577   127.0.0.1         127.0.0.1         DNS        63 Standard query 0x0015 AAAA www.goole.com
9635 5629.593838   127.0.0.1         127.0.0.1         DNS       137 Standard query response 0x0015 AAAA www.goole.com SOA ns1083.ui-dns.de
```


Wireshark · 分组 9633 · Adapter for loopback traffic capture

```
            [Label Count: 3]
            Type: A (Host Address) (1)
            Class: IN (0x0001)
    Answers
      www.goole.com: type A, class IN, addr 127.0.0.1
            Name: www.goole.com
            Type: A (Host Address) (1)
            Class: IN (0x0001)
            Time to live: 99996796 (1157 days, 8 hours, 53 minutes, 16 seconds)
            Data length: 4
            Address: 127.0.0.1
    [Request In: 9632]
    [Time: 0.008684000 seconds]

0000  02 00 00 00 45 00 00 4b  2c 93 00 00 80 11 00 00   ····E··K ,·······
0010  7f 00 00 01 7f 00 00 01  00 35 fa 2b 00 37 2e f2   ········ ·5·+·7.·
0020  00 14 81 00 00 01 00 01  00 00 00 00 03 77 77 77   ········ ·····www
0030  05 67 6f 6f 6c 65 03 63  6f 6d 00 00 01 00 01 c0   ·goole·c om······
0040  0c 00 01 00 01 05 f5 d4  7c 00 04 7f 00 00 01      ········ |······
```

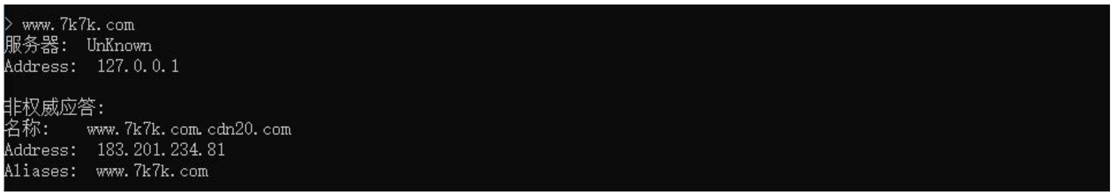
5.3. Case 3: Domain name not in the local database.

First let’s see the local database. We can see www.7k7k.com isn’t here.



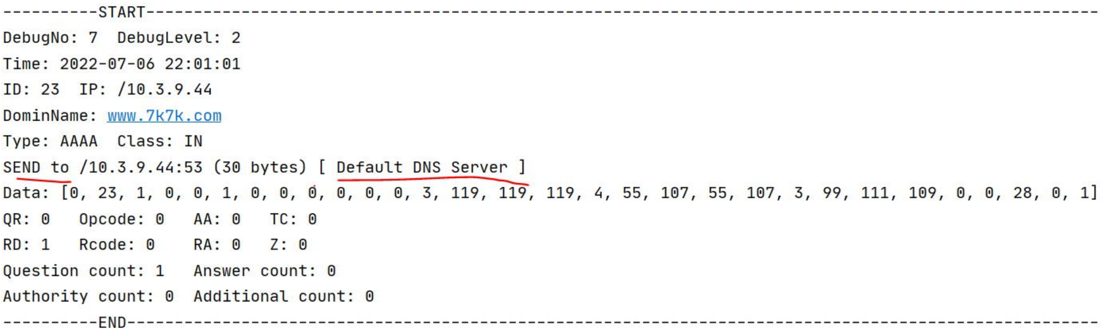
```
www.baidu.com,39.156.66.14,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0,1757042127662,0
www.google.com,0.0.0.0,32.1.0.0.0.0.0.0.0.0.0.0.69.171.246.9,1757107267096,1657107267096
www.goole.com,127.0.0.1,null,1757112580829,0
```

Next we search it in ns-lookup. Result should below.



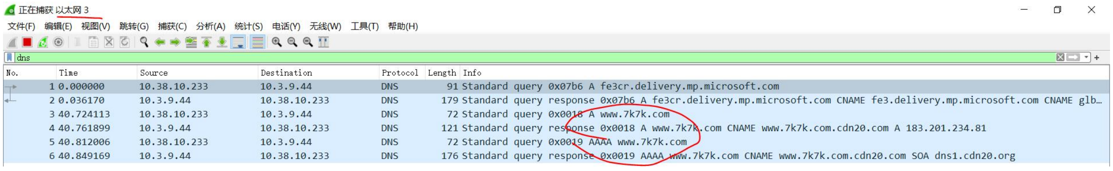
```
> www.7k7k.com
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    www.7k7k.com.cdn20.com
Address:  183.201.234.81
Aliases:  www.7k7k.com
```

Next we see the debug information. We can see we forward it to DNS server.



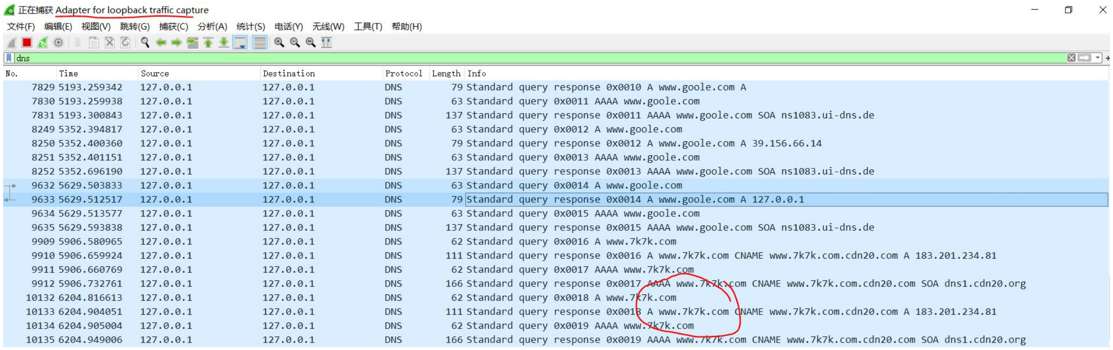
```
----------START----------------------------------------------------------------------------------------------------
DebugNo: 7  DebugLevel: 2
Time: 2022-07-06 22:01:01
ID: 23  IP: /10.3.9.44
DominName: www.7k7k.com
Type: AAAA  Class: IN
SEND to /10.3.9.44:53 (30 bytes) [ Default DNS Server ]
Data: [0, 23, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 3, 119, 119, 119, 4, 55, 107, 55, 107, 3, 99, 111, 109, 0, 0, 28, 0, 1]
QR: 0   Opcode: 0   AA: 0   TC: 0
RD: 1   Rcode: 0    RA: 0   Z: 0
Question count: 1   Answer count: 0
Authority count: 0  Additional count: 0
----------END------------------------------------------------------------------------------------------------------
```

And next we see the result in wire-shark. We can see we can capture packet in this place which means it is send to DNS server.


正在捕获 以太网 3
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
dns

```
No.  Time       Source         Destination    Protocol Length Info
   1 0.000000   10.38.10.233   10.3.9.44      DNS      91 Standard query 0x07b6 A fe3cr.delivery.mp.microsoft.com
   2 0.036170   10.3.9.44      10.38.10.233   DNS     179 Standard query response 0x07b6 A fe3cr.delivery.mp.microsoft.com CNAME fe3.delivery.mp.microsoft.com CNAME glb…
   3 40.724113  10.38.10.233   10.3.9.44      DNS      72 Standard query 0x0018 A www.7k7k.com
   4 40.761899  10.3.9.44      10.38.10.233   DNS     121 Standard query response 0x0018 A www.7k7k.com CNAME www.7k7k.com.cdn20.com A 183.201.234.81
   5 40.812006  10.38.10.233   10.3.9.44      DNS      72 Standard query 0x0019 AAAA www.7k7k.com
   6 40.849169  10.3.9.44      10.38.10.233   DNS     176 Standard query response 0x0019 AAAA www.7k7k.com CNAME www.7k7k.com.cdn20.com SOA dns1.cdn20.org
```

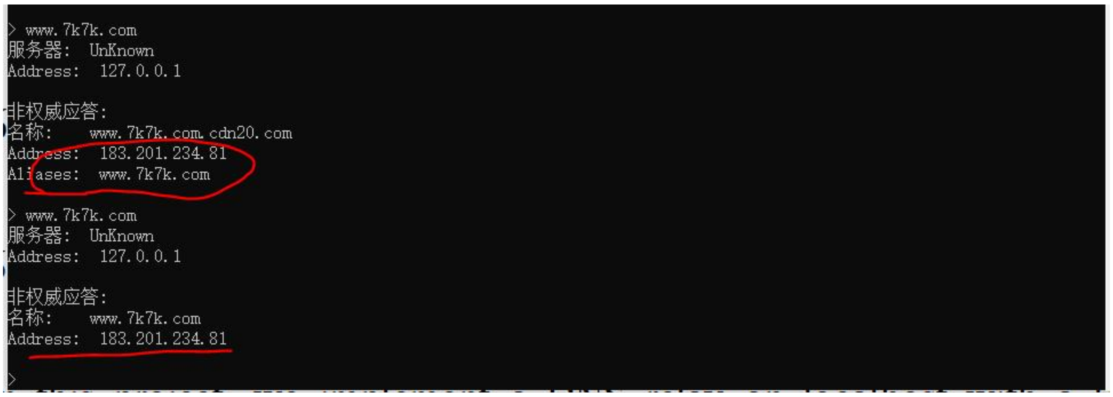
And next figure showed response is forwarded to client.


正在捕获 Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
dns

```
No.   Time         Source      Destination  Protocol Length Info
 7829 5193.259342  127.0.0.1   127.0.0.1    DNS       79 Standard query response 0x0010 A www.goole.com A
 7830 5193.259938  127.0.0.1   127.0.0.1    DNS       63 Standard query 0x0011 AAAA www.goole.com
 7831 5193.300843  127.0.0.1   127.0.0.1    DNS      137 Standard query response 0x0011 AAAA www.goole.com SOA ns1083.ui-dns.de
 8249 5352.394817  127.0.0.1   127.0.0.1    DNS       63 Standard query 0x0012 A www.goole.com
 8250 5352.400360  127.0.0.1   127.0.0.1    DNS       79 Standard query response 0x0012 A www.goole.com A 39.156.66.14
 8251 5352.401151  127.0.0.1   127.0.0.1    DNS       63 Standard query 0x0013 AAAA www.goole.com
 8252 5352.696190  127.0.0.1   127.0.0.1    DNS      137 Standard query response 0x0013 AAAA www.goole.com SOA ns1083.ui-dns.de
 9632 5629.503833  127.0.0.1   127.0.0.1    DNS       63 Standard query 0x0014 A www.goole.com
 9633 5629.512517  127.0.0.1   127.0.0.1    DNS       79 Standard query response 0x0014 A www.goole.com A 127.0.0.1
 9634 5629.513577  127.0.0.1   127.0.0.1    DNS       63 Standard query 0x0015 AAAA www.goole.com
 9635 5629.593838  127.0.0.1   127.0.0.1    DNS      137 Standard query response 0x0015 AAAA www.goole.com SOA ns1083.ui-dns.de
 9909 5906.580965  127.0.0.1   127.0.0.1    DNS       62 Standard query 0x0016 A www.7k7k.com
 9910 5906.659924  127.0.0.1   127.0.0.1    DNS      111 Standard query response 0x0016 A www.7k7k.com CNAME www.7k7k.com.cdn20.com A 183.201.234.81
 9911 5906.660769  127.0.0.1   127.0.0.1    DNS       62 Standard query 0x0017 AAAA www.7k7k.com
 9912 5906.732761  127.0.0.1   127.0.0.1    DNS      166 Standard query response 0x0017 AAAA www.7k7k.com CNAME www.7k7k.com.cdn20.com SOA dns1.cdn20.org
10132 6204.816613  127.0.0.1   127.0.0.1    DNS       62 Standard query 0x0018 A www.7k7k.com
10133 6204.904051  127.0.0.1   127.0.0.1    DNS      111 Standard query response 0x0018 A www.7k7k.com CNAME www.7k7k.com.cdn20.com A 183.201.234.81
10134 6204.905004  127.0.0.1   127.0.0.1    DNS       62 Standard query 0x0019 AAAA www.7k7k.com
10135 6204.949006  127.0.0.1   127.0.0.1    DNS      166 Standard query response 0x0019 AAAA www.7k7k.com CNAME www.7k7k.com.cdn20.com SOA dns1.cdn20.org
```

5.4. Case 4: Domain name not in the local database at first but cache after first search.

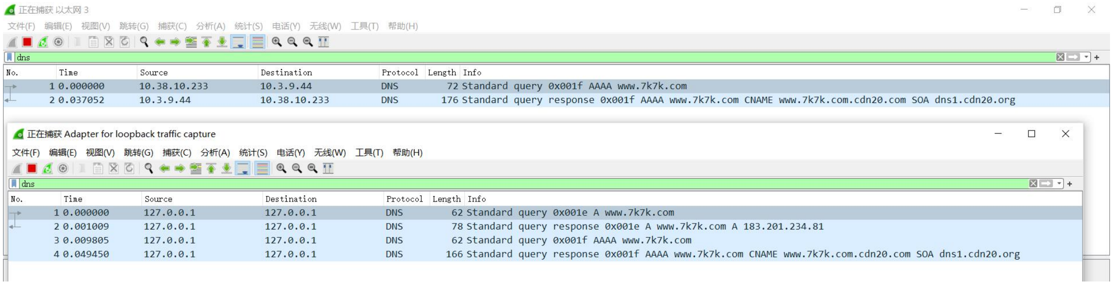
This is the continuous part from 5.3. From 5.3, we know that www.7k7k.com isn't in local database at first. However if we search it second time. Result will be different. We can see it from ns-lookup.



```
> www.7k7k.com
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    www.7k7k.com.cdn20.com
Address:  183.201.234.81
Aliases:  www.7k7k.com

> www.7k7k.com
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    www.7k7k.com
Address:  183.201.234.81

>
```

We can see the difference here. As we didn't cache aliases information, we only write back the address here.

Next, if we see it from wire-shark.


正在捕获 以太网 3
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
dns

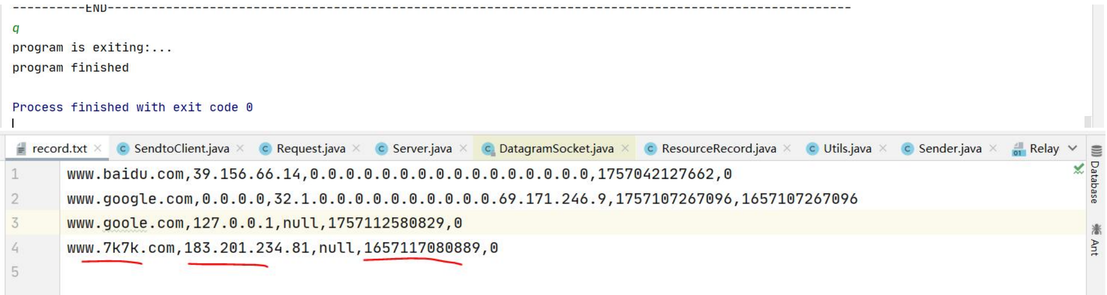
```
No.  Time      Source       Destination  Protocol Length Info
   1 0.000000  10.38.10.233 10.3.9.44    DNS       72 Standard query 0x001f AAAA www.7k7k.com
   2 0.037052  10.3.9.44    10.38.10.233 DNS      176 Standard query response 0x001f AAAA www.7k7k.com CNAME www.7k7k.com.cdn20.com SOA dns1.cdn20.org
```

正在捕获 Adapter for loopback traffic capture
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
dns

```
No.  Time      Source     Destination  Protocol Length Info
   1 0.000000  127.0.0.1  127.0.0.1    DNS       62 Standard query 0x001e A www.7k7k.com
   2 0.001009  127.0.0.1  127.0.0.1    DNS       78 Standard query response 0x001e A www.7k7k.com A 183.201.234.81
   3 0.009805  127.0.0.1  127.0.0.1    DNS       62 Standard query 0x001f AAAA www.7k7k.com
   4 0.049450  127.0.0.1  127.0.0.1    DNS      166 Standard query response 0x001f AAAA www.7k7k.com CNAME www.7k7k.com.cdn20.com SOA dns1.cdn20.org
```

We can see type A isn’t relay to DNS server， because IpV4 has been cached in our system. However, it doesn’t have IpV6, so IpV6 hasn’t been cached and this request will relay to DNS server.

Next, if we input q to our program, it will run the exit process and write cache back to the local database. Result showed next.

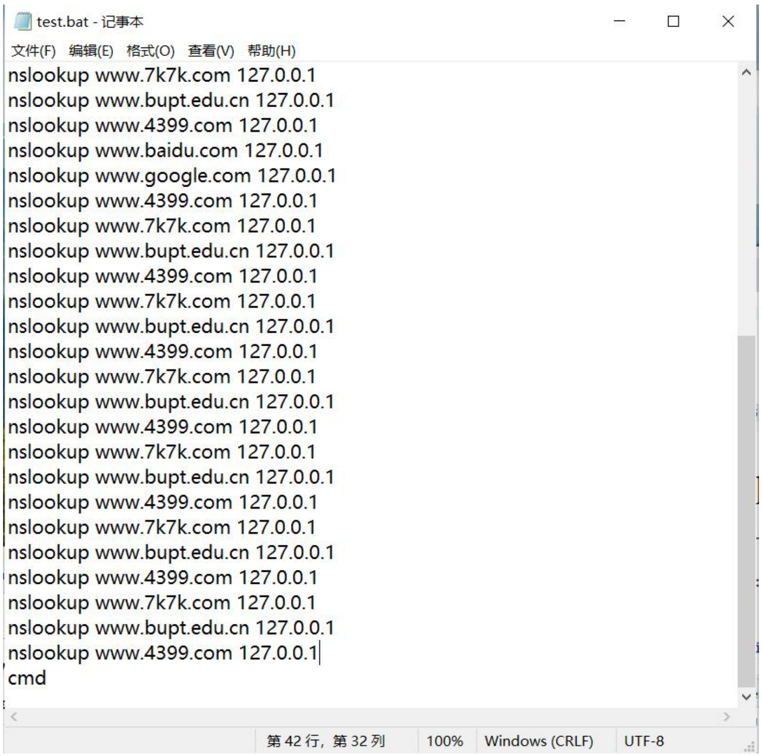


```
----------END------------------------------------------------------------------------------------------
q
program is exiting:...
program finished

Process finished with exit code 0
```

```
www.baidu.com,39.156.66.14,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0,1757042127662,0
www.google.com,0.0.0.0,32.1.0.0.0.0.0.0.0.0.0.0.69.171.246.9,1757107267096,1657107267096
www.goole.com,127.0.0.1,null,1757112580829,0
www.7k7k.com,183.201.234.81,null,1657117080889,0
```

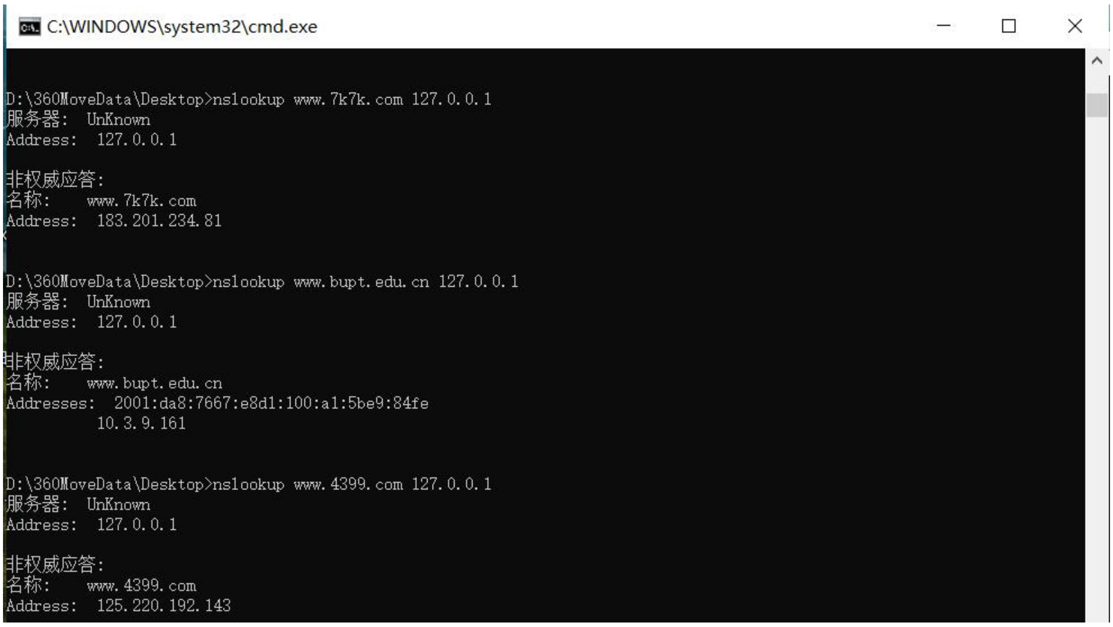
5.5. Case 5: Run the script to test the concurrent requests.

First let me show our script. The total number of requests are 42. Some are same to each other and some are new. And some domain names are blocked.



```
nslookup www.7k7k.com 127.0.0.1
nslookup www.bupt.edu.cn 127.0.0.1
nslookup www.4399.com 127.0.0.1
nslookup www.baidu.com 127.0.0.1
nslookup www.google.com 127.0.0.1
nslookup www.4399.com 127.0.0.1
nslookup www.7k7k.com 127.0.0.1
nslookup www.bupt.edu.cn 127.0.0.1
nslookup www.4399.com 127.0.0.1
nslookup www.7k7k.com 127.0.0.1
nslookup www.bupt.edu.cn 127.0.0.1
nslookup www.4399.com 127.0.0.1
nslookup www.7k7k.com 127.0.0.1
nslookup www.bupt.edu.cn 127.0.0.1
nslookup www.4399.com 127.0.0.1
nslookup www.7k7k.com 127.0.0.1
nslookup www.bupt.edu.cn 127.0.0.1
nslookup www.4399.com 127.0.0.1
nslookup www.7k7k.com 127.0.0.1
nslookup www.bupt.edu.cn 127.0.0.1
nslookup www.4399.com 127.0.0.1
nslookup www.7k7k.com 127.0.0.1
nslookup www.bupt.edu.cn 127.0.0.1
nslookup www.4399.com 127.0.0.1
cmd
```

Next I will run this script and show the command line result. We can see they are all answered well.



```
D:\360MoveData\Desktop>nslookup www.7k7k.com 127.0.0.1
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    www.7k7k.com
Address:  183.201.234.81


D:\360MoveData\Desktop>nslookup www.bupt.edu.cn 127.0.0.1
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    www.bupt.edu.cn
Addresses:  2001:da8:7667:e8d1:100:a1:5be9:84fe
          10.3.9.161


D:\360MoveData\Desktop>nslookup www.4399.com 127.0.0.1
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    www.4399.com
Address:  125.220.192.143
```

6. Summary of works and future improvement

6.1. Summary:

1. The project involves high-performance network server programming, which realizes DNS resolution independently with Java and applies it into DNS-Relay;

2. We need to be familiar with DNS datagrams and find a way to generate DNS datagrams when the relay needs to send a response to a request with a blocked or cached domain name. We check the materials and use Wireshark to understand the structure of datagrams, and find a way to express and parse datagrams in Java language

3. Establish a local database to support the cache of unknown DNS records, and use thread pool and thread synchronization to realize a DNS-Relay that supports high concurrency.

4. Use read lock, write lock and read-write lock to solve the thread conflict in the process of caching, sending, receiving and parsing

6.2. Individual Summary & Reflection:

	This project is a very good example for high performance network server programming and gives me a deep understanding of how to process the DNS packet which is also a great example for me to understand how to use the network protocol. First, let me talk about the high performance

HAN ZHUOHANG	network server. In this project, we try our best to improve the performance. So we use the threads pool to support the concurrent requests. Threads pool is better than pure thread as it can manage the thread automatically and it won't need to create and destroy thread every time. And we choose to implement read lock and write lock to synchronize the threads. The reason why we use two kind of locks is we want to improve the performance and read lock won't exclude each other if there is only read operation. Second 1 am going to talk about the protocol understanding. In year 2, we have learned many protocols like TCP. We know each protocol will define the datagram format. However, we don't have a close touching to the format. In exam, we only need to remember the function of each filed. However, in this project, I learned how to parse the datagram. I learned that they are all byte stream and we need to map them to meaningful filed. I learned how to get or set one bit using mask and & or |. Besides, I learned that we should process the variable length part using offset or pipeline concept. At the end, I want to thank for my teammates as good teammates will make make the team as a team. We have efficient division of labour. They help me a lot to solve some lower level problem so that I can focus more on high level of this project.
LIU SENDONG	Through this project, I have a deeper understanding of DNS resolution. I am responsible for the resolution of question section, in which the domain name resolution part is very meaningful. I need to use &amp; 0xff to convert byte to int to extract the number of fields. Then I learned the thread synchronization method of Java to support the highly concurrent send and recv process. Finally, I realized two debug modes and embedded them in the program to print debug

	information in real time, This is very meaningful for verifying the performance and concurrency of the program. In general, we completed a DNS relay in this project, which gave me more understanding and application of high-performance network programming.
LIAO HAOTIAN	Through this project, I have a deep understanding of the DNS protocol, and also have more deep understanding of the concurrency. I learned about what the role the DNS relay play between the client and the DNS server. And I also knew about the DNS datagram structure, ans also learned how to decode the header of the datagram as well as the functions of each byte. In our DNS relay, we maintain a local database. And my main work is to realize the basic functions of the database. We take the TTL into consideration, so the database should be renewed every time we check it. To cut down the I/O time, we only read the text or write to the text once. And also the DNS relay should support multi-access. So we also add read write lock to each read or write operation. In this way we make sure all the cached data is valid data.

6.3. Future Improvement:

1. For cache in the disk improvement, we can change the data structure of the local database so that it can cache more than one record of IpV4 or IpV6. And we can cache alias address information in the future.
2. For type improvement, we could process more type than IPv4 and IPv6.

3. For cache in the main memory, we can adapt more efficient algorithm to search the information rather than search it one by one.
4. For debug information, we can define more different debug levels.