DNS Relay 实验报告

大数据学院 和泳毅 PB19010450

一、实验目标

1. 编写一个程序，该程序从配置文件加载“域名 - IP地址”列表，并按以下要求处理DNS请求：
    - **拦截**：如果待查询的域名在列表中且对应IP地址为 0.0.0.0 ，则向客户发送0.0.0.0。
    - **本地**：如果待查询的域名在列表中且对应IP地址有意义，则将对应的IP地址返回给客户。
    - **中继**：如果带查询域名不在列表中，则将查询报文转交给外部服务器并中继响应报文 。
2. 对于每个DNS报文，输出其域名，并显示应用了哪一种处理方式和响应时间

二、实验原理

1.DNS报文结构

DNS 分为查询请求和查询响应，请求和响应的报文结构基本相同，主要分为 3 部分：基础结构部分、问题部分、资源记录部分。

DNS 报文格式如下：

事务ID（Transaction ID）	标志（Flags）
问题计数（Questions）	回答资源记录数（Answer RRs）
权威名称服务器计数（Authority RRs）	附加资源记录数（Additional RRs）
查询问题区域（Queries）	
回答问题区域（Answers）	
权威名称服务器区域（Authoritative nameservers）	
附加信息区域（Additional records）	

其中DNS的报文首部共12个字节，包括事务 ID、标志、问题计数、回答资源记录数、权威名称服务器计数和附加资源记录数6 个字段。

2.首部

DNS 报文的基础结构部分指的是报文首部：

事务ID（Transaction ID）	标志（Flags）
问题计数（Questions）	回答资源记录数（Answer RRs）
权威名称服务器计数（Authority RRs）	附加资源记录数（Additional RRs）

该部分中每个字段含义如下。

- 事务 ID：DNS 报文的 ID 标识。对于请求报文和其对应的应答报文，该字段的值是相同的。通过它可以区分 DNS 应答报文是对哪个请求进行响应的。

- 标志：DNS 报文中的标志字段。
- 问题计数：DNS 查询请求的数目。
- 回答资源记录数：DNS 响应的数目。
- 权威名称服务器计数：权威名称服务器的数目。
- 附加资源记录数：额外的记录数目。

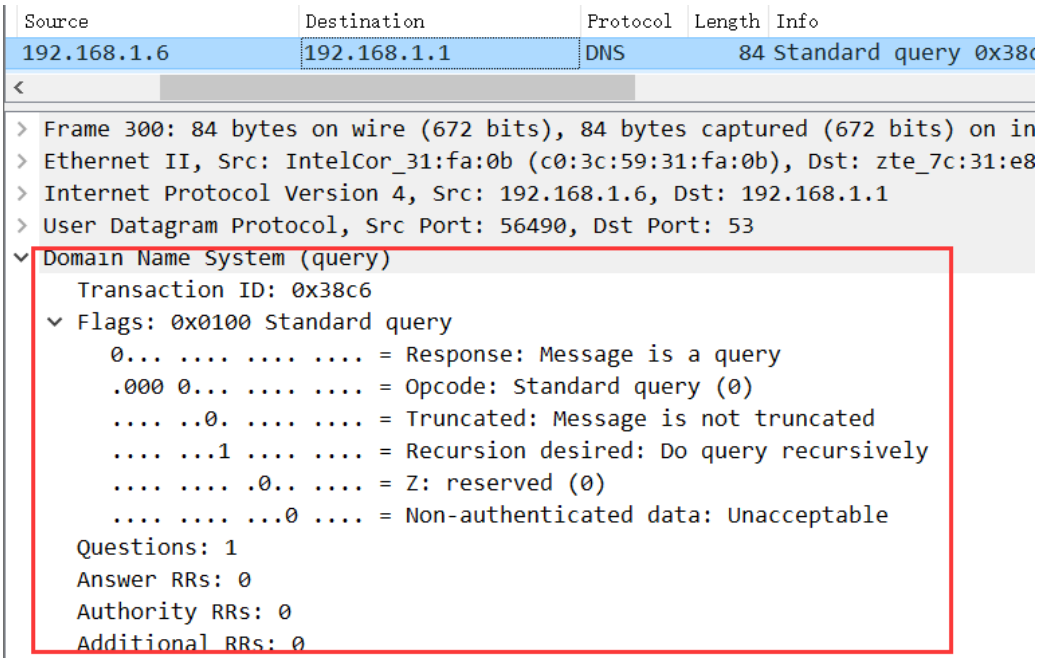
基础结构部分中的标志字段又分为若干个字段：

QR	Opcode	AA	TC	RD	RA	Z	rcode

标志字段中每个字段的含义如下：

- QR：查询请求/响应的标志信息。查询请求时，值为 0；响应时，值为 1。
- Opcode：操作码。其中，0 表示标准查询；1 表示反向查询；2 表示服务器状态请求。
- AA：授权应答，该字段在响应报文中有效。值为 1 时，表示名称服务器是权威服务器；值为 0 时，表示不是权威服务器。
- TC：表示是否被截断。值为 1 时，表示响应已超过 512 字节并已被截断，只返回前 512 个字节。
- RD：期望递归。该字段能在一个查询中设置，并在响应中返回。该标志告诉名称服务器必须处理这个查询，这种方式被称为一个递归查询。如果该位为 0，且被请求的名称服务器没有一个授权回答，它将返回一个能解答该查询的其他名称服务器列表。这种方式被称为迭代查询。
- RA：可用递归。该字段只出现在响应报文中。当值为 1 时，表示服务器支持递归查询。
- Z：保留字段，在所有的请求和应答报文中，它的值必须为 0。
- rcode：返回码字段，表示响应的差错状态。当值为 0 时，表示没有错误；当值为 1 时，表示报文格式错误，服务器不能理解请求的报文；当值为 2 时，表示域名服务器失败，因为服务器的原因导致没办法处理这个请求；当值为 3 时，表示名字错误，只有对授权域名解析服务器有意义，指出解析的域名不存在；当值为 4 时，表示查询类型不支持，即域名服务器不支持查询类型；当值为 5 时，表示拒绝，一般是服务器由于设置的策略拒绝给出应答，如服务器不希望对某些请求者给出应答。

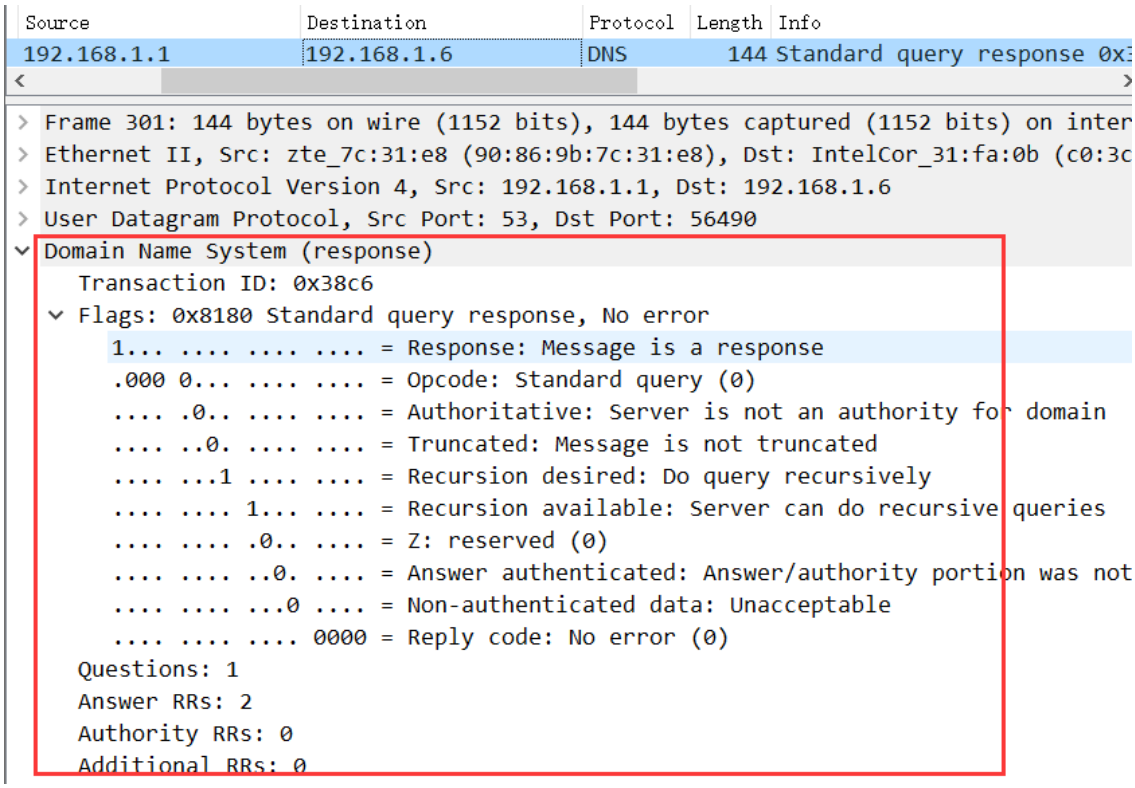
具体请求数据报首部如下：




```
Domain Name System (query)
    Transaction ID: 0x38c6                          #事务ID
    Flags: 0x0100 Standard query                    #报文中的标志字段
        0... .... .... .... = Response: Message is a query
                                                    #QR字段，值为0，因为是一个请求包
        .000 0... .... .... = Opcode: Standard query (0)
                                                    #Opcode字段，值为0，因为是标准查询
        .... ..0. .... .... = Truncated: Message is not truncated
                                                    #TC字段
        .... ...0 .... .... = Recursion desired: Don't do query recursively
                                                    #RD字段
        .... .... .0.. .... = Z: reserved (0)       #保留字段，值为0
        .... .... ...0 .... = Non-authenticated data: Unacceptable
                                                    #保留字段，值为0
    Questions: 1                                    #问题计数，这里有1个问题
    Answer RRs: 0                                   #回答资源记录数
    Authority RRs: 0                                #权威名称服务器计数
    Additional RRs: 0                               #附加资源记录数
```

需要注意的是，在请求中 Questions 的值不可能为 0；Answer RRs，Authority RRs，Additional RRs 的值都为 0，因为在请求中还没有响应的查询结果信息。这些信息在响应包中会有相应的值。

具体响应数据报首部如下：




```
Domain Name System (response)
    Transaction ID: 0x38c6                                  #事务ID
    Flags: 0x8180 Standard query response, No error         #报文中的标志字段
        1... .... .... .... = Response: Message is a response
                                                            #QR字段，值为1，因为是一个响应包
        .000 0... .... .... = Opcode: Standard query (0)  # Opcode字段
        .... .0.. .... .... = Authoritative: Server is not an authority for domain
                                                              #AA字段
        .... ..0. .... .... = Truncated: Message is not truncated
                                                            #TC字段
        .... ...1 .... .... = Recursion desired: Do query recursively
                                                            #RD字段
        .... .... 1... .... = Recursion available: Server can do recursive queries
                                                              #RA字段
        .... .... .0.. .... = Z: reserved (0)
        .... .... ..0. .... = Answer authenticated: Answer/authority portion was
not authenticated by the server
        .... .... ...0 .... = Non-authenticated data: Unacceptable
        .... .... .... 0000 = Reply code: No error (0)     #返回码字段
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 0
```

在输出信息最后可以看到 Answer RRs，Authority RRs，Additional RRs 都有了相应的值（不一定全为 0）。

3.问题

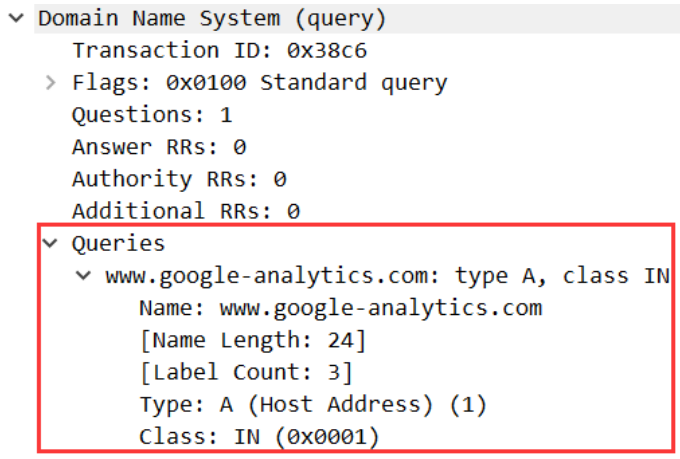
问题部分指的是报文格式中查询问题区域（Queries）部分。该部分是用来显示 DNS 查询请求的问题，通常只有一个问题。该部分包含正在进行的查询信息，包含查询名（被查询主机名字）、查询类型、查询类。问题部分格式如下：

<table>
<tr><td colspan="2">查询名</td></tr>
<tr><td>查询类型</td><td>查询类</td></tr>
</table>

该部分中每个字段含义如下：

- 查询名：一般为要查询的域名，有时也会是 IP 地址，用于反向查询。
- 查询类型：DNS 查询请求的资源类型。通常查询类型为 A 类型，表示由域名获取对应的 IP 地址。
- 查询类：地址类型，通常为互联网地址，值为 1。

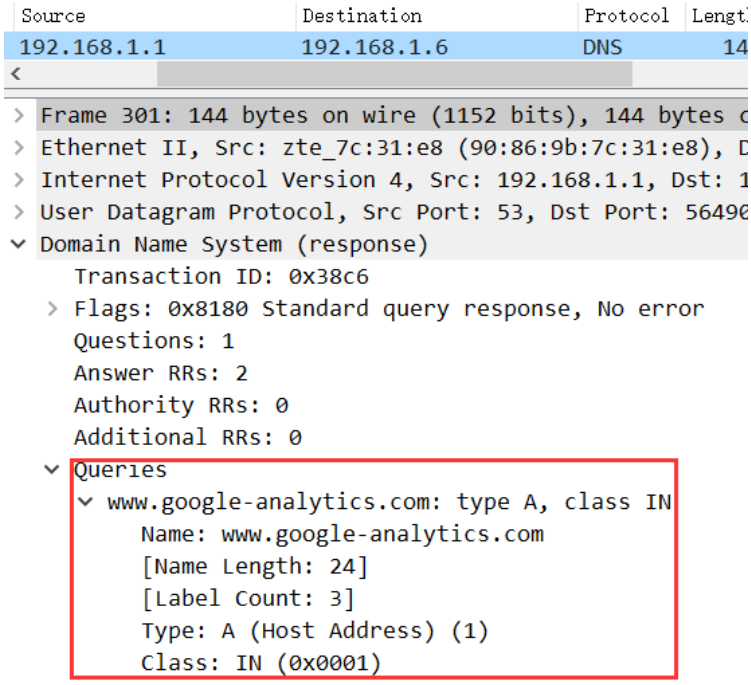
具体请求数据报问题部分如下：




```
Domain Name System (query)                          #查询请求
    Queries                                         #问题部分
        www.google-analytics.com: type A, class IN
            Name: www.google-analytics.com          #查询名字段
            [Name Length: 24]
            [Label Count: 3]
            Type: A (Host Address) (1)              #查询类型字段，这里为A类型
            Class: IN (0x0001)                      #查询类字段，这里为互联网地址
```

可以看到 DNS 请求类型为 A，那么得到的响应信息也应该为 A 类型。

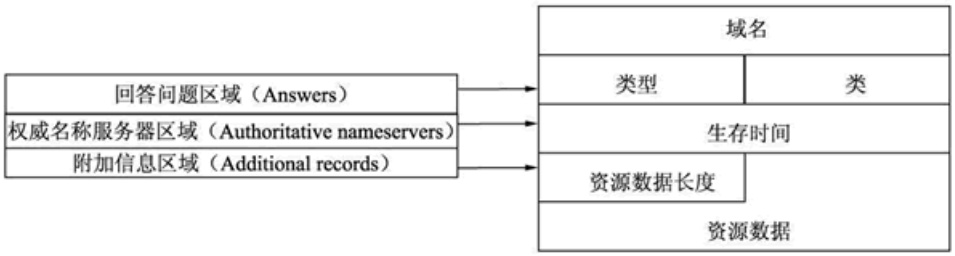
具体响应数据报问题部分如下：



从图中 Queries 部分中可以看到，响应包中的查询类型也是 A，与请求包的查询类型是一致的。

4.资源记录

资源记录部分是指 DNS 报文格式中的最后三个字段，包括回答问题区域字段、权威名称服务器区域字段、附加信息区域字段。这三个字段均采用一种称为资源记录的格式，格式如图所示。



资源记录格式中每个字段含义如下：

- 域名：DNS 请求的域名。
- 类型：资源记录的类型，与问题部分中的查询类型值是一样的。
- 类：地址类型，与问题部分中的查询类值是一样的。
- 生存时间：以秒为单位，表示资源记录的生命周期，一般用于当地址解析程序取出资源记录后决定保存及使用缓存数据的时间。它同时也可以表明该资源记录的稳定程度，稳定的信息会被分配一个很大的值。
- 资源数据长度：资源数据的长度。
- 资源数据：表示按查询段要求返回的相关资源记录的数据

资源记录部分只有在 DNS 响应包中才会出现。具体响应数据报资源记录部分如下：

```
Answers
  www.google-analytics.com: type CNAME, class IN, cname www-google-analyti
      Name: www.google-analytics.com
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
      Time to live: 65859 (18 hours, 17 minutes, 39 seconds)
      Data length: 32
      CNAME: www-google-analytics.l.google.com
  www-google-analytics.l.google.com: type A, class IN, addr 203.208.40.65
      Name: www-google-analytics.l.google.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 39 (39 seconds)
      Data length: 4
      Address: 203.208.40.65
```

```
Answers                                                 #“回答问题区域”字段
    www.google-analytics.com: type CNAME, class IN, cname www-google-
analytics.l.google.com                                  #资源记录
        Name: www.google-analytics.com                  #域名字段，这里请求的域名
        Type: CNAME (Canonical NAME for an alias) (5)   #类型字段，这里为CNAME类型
        Class: IN (0x0001)                              #类字段
        Time to live: 65859 (18 hours, 17 minutes, 39 seconds)#生存时间
        Data length: 32                                 #数据长度
        CNAME: www-google-analytics.l.google.com  #资源数据，这里为规范主机名
    www-google-analytics.l.google.com: type A, class IN, addr 203.208.40.65
        Name: www-google-analytics.l.google.com
        Type: A (Host Address) (1)                      #类型字段，这里为A类型
        Class: IN (0x0001)
        Time to live: 39 (39 seconds)
        Data length: 4
        Address: 203.208.40.65                          #资源数据，这里为IP地址
```

三、详细代码

1.实验软件

本实验使用python 3.8版本

2.第三方库

```
import socket      # 提供套接字接口
import threading   # 并行处理
import struct      # 解码二进制报文
import time        # 计算响应时间
```

3.代码结构

本次实验使用两个类： `DNS_Relay` 和 `DNS_Message`：

```
-DNS_Relay    # 多线程处理请求
--start
--name_to_ip
-DNS_Message  # 解析、生成报文
--get_name
--get_query
-generate_message
```

4.函数设计

DNS_Relay:

__init__：初始化。

```
def __init__(self, file_name, server_name):
    self.file_name = file_name
    self.url_ip = {}  # 存入本地文件
    self.server_name = server_name
    self.dns_id = {}  # 用于计算响应时间
```

start：读取本地文件，并多线程处理报文。

```
def start(self):
    f = open(self.file_name, 'r', encoding='utf-8')
    for line in f:
        ip, name = line.split(' ')
        self.url_ip[name.strip('\n')] = ip
    f.close()

    buffer_size = 1024
    server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server.bind(('', 53))
    server.setblocking(False)
    while True:
        try:
            data, addr = server.recvfrom(buffer_size)
            threading.Thread(target=self.name_to_ip, args=(server, data,
addr)).start()
        except:
            pass
```

name_to_ip：具体处理报文，决定是否本地/中继/拦截处理，并计算响应时间。其中调用 `DNS_Message` 类。

```
def name_to_ip(self, server, data, addr):
    start_time = time.time()
    dns_message = DNS_Message(data)
    if dns_message.QR == 0:  # 查询报文
        name = dns_message.name
        if name in self.url_ip and dns_message.type == 1:
            ip = self.url_ip[name]
            res = dns_message.generate_message(ip)
            server.sendto(res, addr)
            res_time = time.time() - start_time # 计算响应时间
            if ip == '0.0.0.0':  # 拦截类型
                print('【拦截】   域名: {}'.format(name).ljust(50),'IP: {}'.format(ip).ljust(15),'时间: {} s\n'.format(res_time),end='')
            else: # 本地类型
                print('【本地】   域名: {}'.format(name).ljust(50),'IP: {}'.format(ip).ljust(15),'时间: {} s\n'.format(res_time),end='')
        else:
            server.sendto(data, self.server_name)
            self.dns_id[dns_message.ID] = (name, addr, start_time) # 存入事件ID和响应开始时间
    if dns_message.QR == 1:  # 响应报文
        if dns_message.ID in self.dns_id:
            name, destination, start_time = self.dns_id[dns_message.ID]
            server.sendto(data, destination)
            res_time = time.time()-start_time # 计算响应时间
            print('【中继】   域名: {}'.format(name).ljust(50),'时间: {} s\n'.format(res_time),end='')
            try:
                del self.dns_id[dns_message.ID]
            except:
                pass
```

DNS_Message:

__init__：初始化。

```
def __init__(self, data):
    self.ID, self.flags, self.quests, self.answers, self.author, self.addition =
struct.unpack('>HHHHHH',data[0:12])
    self.QR = (self.flags & 0x8000) >> 15
    self.name = self.get_name(data)
    self.querybytes, self.type, self.classify = self.get_query(data[12:])
```

其中，ID为事务ID，flags为首部标志字段，quests为问题计数，answers为回答资源记录数，author为权威名称服务器计数，addition为附加资源记录数，QR为查询请求/响应的标志信息，name为域名。

get_name：提取域名。

```
def get_name(self, data):
    name = ''
    i = 12
    while data[i] != 0:
        for len in range(1, data[i] + 1):
            name += chr(data[i+len])
        name += '.'
        i += data[i] + 1
    return name[:-1].strip()
```

根据域名的规则来解析，并人为地添加字符"."。

get_query：提取问题部分的type、class字段。

```
def get_query(self, data):
    i, length = 0, 0
    while True:
        if length == 0:
            if data[i] == 0:
                break
            else:
               length = int(data[i])
        else:
            length -= 1
       i += 1
    querybytes = data[0:i+1]
    type, classify = struct.unpack('>HH', data[i+1:i+5])
    return querybytes, type, classify
```

generate_message：生成响应报文。

```
def generate_message(self, ip):
    message = struct.pack('>H', self.ID)
    message += struct.pack('>H', 0x8583 if ip == '0.0.0.0' else 0x8180)
    answers = 1
    message += struct.pack('>HHHH', self.quests, answers, self.author,
self.addition)

    message += self.querybytes + struct.pack('>HH', self.type, self.classify)

    name, type, classify, ttl, datalength = 0xc00c, 1, 1, 200, 4
    message += struct.pack('>HHHLH', name, type, classify, ttl, datalength)
    s_ip = ip.split('.')
    message += struct.pack('BBBB', int(s_ip[0]), int(s_ip[1]), int(s_ip[2]),
int(s_ip[3]))
    return message
```

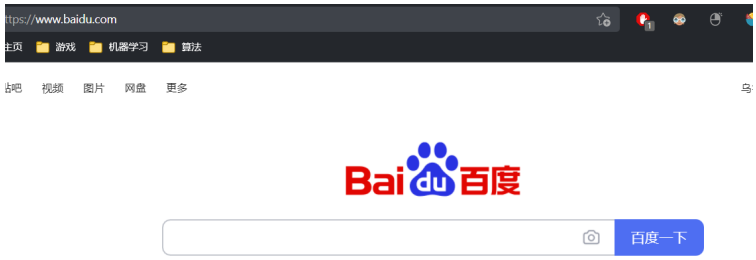
按照首部、问题部分、回答部分、权威名称服务器部分，附加资源部分的DNS报文结构构造报文。其中对于拦截类型的报文，标志字段为0x8583，rcode为3。对于非拦截的报文，标志字段为0x8180。

四、测试案例

本地文件：

```
0.0.0.0 pic1.zhimg.com
0.0.0.0 pic2.zhimg.com
0.0.0.0 pic3.zhimg.com
0.0.0.0 pic4.zhimg.com
0.0.0.0 static.zhimg.com
0.0.0.0 picb.zhiming.com
182.61.200.7 www.baidu.com
127.0.0.1 www.test1.com
```

1. www.baidu.com



网页被正常打开。

```
C:\Users\Yorick He>nslookup www.baidu.com
服务器:  localhost
Address:  127.0.0.1

非权威应答:
名称:    www.baidu.com
Address:  182.61.200.7
```

nslookup运行正常。

```
【中继】    域名：api1.origin.com                          时间：0.017213821411132812 s
【中继】    域名：clientconfig.akamai.steamstatic.com      时间：0.014714956283569336 s
【中继】    域名：gz1yd.uda.udauda.me                      时间：0.013733863830566406 s
【中继】    域名：steamcdn-a.akamaihd.net                  时间：0.011772632598876953 s
【中继】    域名：policy.video.iqiyi.com                   时间：0.020123958587646484 s
【中继】    域名：activity.windows.com                     时间：0.027020931243896484 s
【中继】    域名：btrace.qq.com                            时间：0.08864331245422363 s
【中继】    域名：1.0.0.127.in-addr.arpa                   时间：0.08098030090332031 s
【本地】    域名：www.baidu.com                            IP：182.61.200.7 时间：0.0 s
```

可以看到百度被本地解析了。

2.www.sohu.com

网页被正常打开。

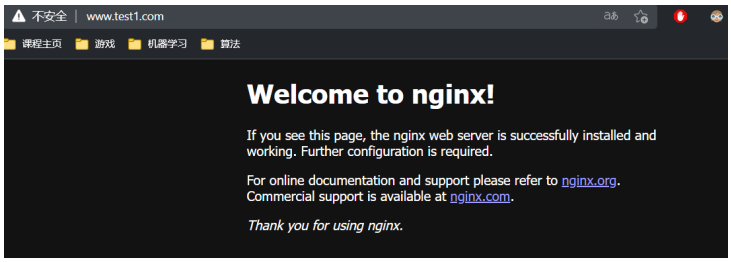
```
C:\Users\Yorick He>nslookup www.sohu.com
服务器:  localhost
Address:  127.0.0.1

非权威应答:
名称:    fwh.a.sohu.com
Addresses:  240e:95c:3005::20b
          240e:95c:3005::20a
          119.96.200.204
Aliases:  www.sohu.com
          gs.a.sohu.com
```

nslookup运行正常。

```
【中继】    域名：pv.sohu.com                          时间：0.003924369812011719 s
【中继】    域名：edge.microsoft.com                   时间：0.006867408752441406 s
【中继】    域名：1.0.0.127.in-addr.arpa               时间：0.04269051551818848 s
【中继】    域名：www.sohu.com                         时间：0.04120206832885742 s
【中继】    域名：www.sohu.com                         时间：0.04120159149169922 s
【中继】    域名：pcrec.baidu.com                      时间：0.019620418548583984 s
【中继】    域名：nav.smartscreen.microsoft.com        时间：0.019620418548583984 s
```

可以看到搜狐被中继解析了。

3.www.test1.com



test1.com本身不存在，但在本地文件中被改写为127.0.0.1. 并且提前使用nginx作为本地服务器，打开后可以看到欢迎页面。

```
C:\Users\Yorick He>nslookup www.test1.com
服务器:  localhost
Address:  127.0.0.1

非权威应答:
名称:    www.test1.com
Address:  127.0.0.1
```

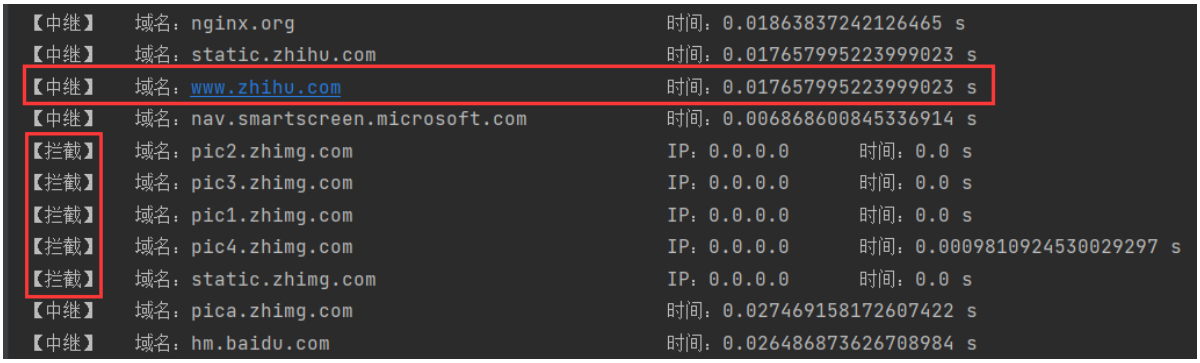
nslookup运行正常。

```
【中继】    域名：aefd.nelreports.net                  时间：0.011294126510620117 s
【本地】    域名：www.test1.com                        IP：127.0.0.1    时间：0.0 s
【本地】    域名：www.test1.com                        IP：127.0.0.1    时间：0.0 s
【中继】    域名：nav.smartscreen.microsoft.com        时间：0.006868124008178711 s
【中继】    域名：flux.hcdn.qiyi.com                   时间：0.0166776180267334 s
【中继】    域名：flux.hcdn.qiyi.com                   时间：0.0166776180267334 s
```

可以看到test1.com被本地解析了。

4.www.zhihu.com

网页被正常打开，但是有些图片无法加载。



可以看到zhihu.com被中继解析了，但一些图片（本地文件中设置的）被拦截。

五、实验总结

在本次实验中，我通过复习课堂上有关DNS报文的结构，并参考了网络上的大量案例，将课堂知识应用于实际，对DNS的知识理解更进一步。在使用python编写程序的过程中学习了threading、struct等库在计算机网络程序中的应用。完成实验后，充分认识到简单理论背后的实际运转是十分复杂的，这让我对计算机网络有了更加深刻的认识。