DNS_Relay实验文档

详尽版实验文档。

DNS_Relay实验文档
- 实验要求
- 背景知识
  - 什么是DNS中继：
  - nslookup命令（windows系统)：
- DNS报文的解析过程
  - 1.想成为一个盘踞在53端口的DNS中继服务器，你如何接受53端口收到的报文?
  - 2.接受53端口收到的报文，你该如何处理它，获取它的信息?
- DNS报文的响应过程
  - 1、构造DNS响应报文
    - 1）下载安装wireshark：wireshark download
    - 2）选择正确的接口进行抓包：
    - 3）使用nslookup发出DNS请求
    - 4）使用dns过滤器筛选出DNS报文：
    - 注意
  - 2、DNS报文的中继
- 测试过程：
- 一些知识
  - 域名缓存
- 参考链接

实验要求

详见　README.md 和PPT

example.txt 是一个配置文件示例，DNS Relay Server 根据配置文件来决定如何响应请求。

1. 当请求的域名不在配置文件中时，将请求转发给一个正常的DNS Server ，然后将收到的response回复客户端。
2. 当请求的域名在配置文件中有对应的非0.0.0.0 ip地址，将该ip地址返回给客户端
3. 当请求的域名在配置文件中对应ip地址为0.0.0.0 ，将DNS response 报文中Rcode置为3表示域名不存在，然后将地址返回给客户端。

背景知识

什么是DNS中继：

DNS的主要任务是将便于记忆的网址转换为路由器能识别的IP地址,在进行DNS查询的时候，由DNS中继服务器将你的DNS查询请求转发到负责你要查询的域的DNS服务器.然后将查询结果返回给你,这个过程就是DNS中继.

DNS协议使用端口号53.

nslookup命令（windows系统)：

nslookup命令用于查询DNS的记录，查看域名解析是否正常

用法：

nslookup –qt=类型 目标域名　（默认查询类型是a 地址记录IPV4）

nslookup -qt=类型 目标域名 指定的DNS服务器IP或域名

#例如，这里127.0.1是指定主机为dns服务器

```
*** UnKnown 找不到 http://home.ustc.edu.cn/~mashuang2/: Non-existent domain
PS C:\Users\mashu> nslookup -qt=A bilibili.com 127.0.0.1
服务器:  UnKnown
Address:  127.0.0.1
```

#如果没指定dns-server，用系统默认的dns服务器

```
PS C:\Users\mashu> nslookup -qt=A bilibili.com
服务器:  UnKnown
Address:  192.168.43.1

非权威应答:
名称:    bilibili.com
Addresses:  110.43.34.66
          139.159.241.37
          120.92.174.135
          120.92.78.97
          119.3.238.64
          119.3.70.188
```

想知道更多nslookup的使用方法请自行搜索。

实验要完成的功能是dns中继服务器，不严谨的描述：假装自己的电脑是一台dns中继服务器，就要在自己的电脑有一个一直运行的程序，蹲在53端口，处理dns解析的请求，根据需要返回DNS解析报文。dns解析的请求是怎么产生的呢？nslookup会将请求发给默认的或者指定的服务器，用浏览器浏览网页的时候，也会有dns解析的步骤。如果要nslookup把dns解析的报文发给本地，要指定127.0.0.1为dns服务器。

DNS报文的解析过程

你可以使用任何语言来完成实验，这里给出一些用python实现可能用到的知识，仅供参考。

如果你从未使用或者接触过python,可以阅读以下链接快速入门：

环境搭建：Python3 环境搭建 | 菜鸟教程 (runoob.com)

Vscode:Python VScode 配置 | 菜鸟教程 (runoob.com)

基础语法：Python3 基础语法 | 菜鸟教程 (runoob.com)

基本数据类型：Python3 基本数据类型 | 菜鸟教程 (runoob.com)

字典：Python3 字典 | 菜鸟教程 (runoob.com)

面向对象：Python3 面向对象 | 菜鸟教程 (runoob.com)

网络编程：Python 网络编程 | 菜鸟教程 (runoob.com)

1.想成为一个盘踞在53端口的DNS中继服务器，你如何接受53端口收到的报文?

使用套接字。

Socket又称"套接字"，应用程序通常通过"套接字"向网络发出请求或者应答网络请求，使主机间或者一台计算机上的进程间可以通讯。

提问：域名解析时使用UDP还是TCP协议呢？为什么呢?

套接字的使用方法：Python 网络编程 | 菜鸟教程 (runoob.com)

以下是三个可能用到的函数：

```
#Python 中，我们用 socket（）函数来创建套接字，语法格式如下
socket.socket([family[, type[, proto]]])
#family: 套接字家族可以使 AF_UNIX 或者 AF_INET。
#type: 套接字类型可以根据是面向连接的还是非连接分为 SOCK_STREAM 或 SOCK_DGRAM。
#protocol: 一般不填默认为 0。

#绑定地址（host,port）到套接字， 在 AF_INET下，以元组（host,port）的形式表示地址。
s.bind()

#   接收 UDP 数据，与 recv() 类似，但返回值是（data,address）。其中 data 是包含接收数据的字符串，#address 是发送数据的套接字地址。
s.recvfrom()
```

使用实例：

```
###############套接字######################
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('',port))
server_socket.setblocking(False)

data, addr = server_socket.recvfrom(512)
```

使用如下一段程序，可以从53端口接收UDP报文并打印出来。

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('',53))
server_socket.setblocking(False)
#这里的blocking是什么含义，设置非阻塞和阻塞有什么区别呢？
print("------start-------")
while True:
    try:
        data, addr = server_socket.recvfrom(512)
        print("receive package from:")
        print(addr)
        print("DNS package:")
        print("type:",type(data))
        print("data:",data)

    except:
        continue
#不了解 try except的同学请自行搜索
#注意：像俺这样用try except 会掩盖 try:下面代码的报错信息
#用try except捕捉报错信息：
'''
try:
    ...
except Exception as e:
    print(e)
'''
```

在本地运行这段python代码，与此同时，使用nslookup查询：

```
PS C:\Users\mashu> nslookup -qt=A baidu.com 127.0.0.1
DNS request timed out.
    timeout was 2 seconds.
服务器:  UnKnown
Address:  127.0.0.1

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
*** 请求 UnKnown 超时
```

可以在python程序处得到如下输出：

```
------start-------
receive package from:
('127.0.0.1', 51266)
DNS package:
type: <class 'bytes'>
data: b'\x00\x01\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x011\x010\x010\x03127\x07in-addr\x04arpa\x00\x00\x0c\x00
\x01'
receive package from:
('127.0.0.1', 51267)
DNS package:
type: <class 'bytes'>
data: b'\x00\x02\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x05baidu\x03com\x00\x00\x01\x00\x01'
receive package from:
('127.0.0.1', 51268)
```

ctrl+c可以终止python程序。（这是俺观察到的阻塞和非阻塞的区别之一：非阻塞的套接字可以由ctrl+c中断）

我们观察到，程序收到了来自本地的dns请求报文。

2.接受53端口收到的报文，你该如何处理它，获取它的信息?

首先要知道DNS协议如何规定DNS报文的格式，我们可以阅读rfc1035来获取格式，也可以自行搜索阅读帖子

DNS 报文格式如图所示：

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）	

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、附加资源记录数这 6 个字段是DNS的报文首部，共 12 个字节。每个字段包含两个字节。

一些需要重点关注的字段：

- 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。

  注意PPT里有一句话：Note: the program should be able to handle concurrent quires, by making use of the ID field in the DNS header.

- 标志：DNS 报文中的标志字段。

  基础结构部分中的标志字段又分为若干个字段，如图所示。

QR	Opcode	AA	TC	RD	RA	Z	rcode

标志字段中每个字段的含义如下：

QR（Response）：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。

以刚刚我们接受到的报文为例子：

我们收到的data是bytes类型，bytes由由多个字符组成， 是按照单个字节来处理数据的，Byte代表一字节，1字节包含8位。

```
data = b'\x00\x01\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x011\x010\x010\x03127\x07in-addr\x04arpa\x00\x00\x0c\x00\x01'
print(data)
print(type(data))
'''
ID              A 16 bit identifier assigned by the program that
                generates any kind of query.  This identifier is copied
                the corresponding reply and can be used by the requester
                to match up replies to outstanding queries.
'''
#例如，我们需要第1个和第二个字节来组成ID，也就是将data[0]和data[1]拼合在一起
#data[0]是一个字节，共8位，此处使用了位运算的方法。

ID = (data[0] << 8) + data[1]
QR = data[2] >> 7
'''
QR              A one bit field that specifies whether this message is a
                query (0), or a response (1).

print(QR) #QR只有两种取值0，1，查询请求时，值为 0；响应时，值为 1。
'''
'''
OPCODE          A four bit field that specifies kind of query in this
                message.  This value is set by the originator of a query
                and copied into the response.  The values are:

                0               a standard query (QUERY)

                1               an inverse query (IQUERY)

                2               a server status request (STATUS)

                3-15            reserved for future use
'''
#对于组成flag的各个字段，flag一共2个字节16位，一个小字段也可以通过位运算来获得,这里
# 每一个字段占多少位可以参考rfc1035，也可以用wireshark等软件抓包分析：
Opcode = (data[2] & 0b01111000)>>5
print(Opcode)
Opcode = (data[2] % 128) >> 3
print(Opcode)
'''
```

```
The header contains the following fields:

                                    1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                      ID                       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    QDCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ANCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    NSCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ARCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
'''
#如此这般这般如此，我们就可以解析出其他信息
```

查询问题区域：

QNAME ：域名被编码为一些labels序列，每个labels包含一个字节表示后续字符串长度，以及这个字符串，以**0长度和空字符**串来表示域名结束。注意这个字段可能为奇数字节，不需要进行边界填充对齐。

```
#解释QNAME的这句话听起来怪费解的,我们来观察一下接收到的一些报文：
data:b'\x00\x03\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x08bilibili\x03com\x00\x
00\x01\x00\x01
data:b'\x00\x02\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x05baidu\x03com\x00\x00\
x01\x00\x01'
#在这两组报文里，bilibili和baidu是我们能看清楚的字符，那这个字符前面的东西，\x08bilibili，
\x08代表 了bilibili的字符数，\x05baidu，\x05代表了baidu的字符数。\x03com中,\x03代表了com
的字符数
#那“0长度和空字符串”又是什么鬼呢？这句话换个说法说一遍是：QNAME域被分成很多labels序列，每个
labels包含后续字符长度+字符本身这两个部分（例如上面例子里的\x03com,\x07bilibili)，最后一个
label比较特殊，他标识字符长度的部分写着0，既然写了后面的字符长度是0，后面自然就没有字符啦，也就
是所谓的“空字符串”。
#再换一个说法：QNAME的部分以\x0结束。
```

QTYPE 2个字节表示查询类型，.取值可以为任何可用的类型值，以及通配码来表示所有的资源记录。
QCLASS 2个字节表示查询的协议类，比如，IN代表Internet。 1.3 资源记录格式(Resource record)

QNAME的解析可能有点点麻烦，要有耐心哦！

....

DNS报文的响应过程

解析了报文信息，我们该如何生成回应的报文?

如果收到了一个查询报文（如何判断是查询报文？QR == 0，RecvDp.qtype == 1），解析出它要查询的url(域名），就用上我们的配置文件啦！在配置文件里找，就出现了PPT里讲述的三种情况：

Intercept: If the queried name is in the list and its associated IP address is “0.0.0.0”, responds 0.0.0.0 to the client.

local resolve: If the queried name is in the list and has a meaningful IP address associated, responds that IP address.

Relay: If the domain name is not in the list, relays the query and the response between the server and the client.

附赠一个读取配置文件的函数：

```
#这个函数会将配置文件里的IP，url存为字典，形式为{url:IP}
def read_file(fileName):
    url_ip = {}
    f = open(fileName,'r',encoding = 'utf-8')
    for each_line in f:
        ip_url = each_line.split(' ')
        ip = ip_url[0]
        domain = ip_url[1].strip('\n')
        url_ip[domain] = ip
    f.close()
    return url_ip
url_ip = read_file(fileName)
```

dns请求和应答都是用相同的报文格式，请大家自行查询资料，可能需要重点关注一下字段哦：

QR　1个比特位用来区分是请求（0）还是应答（1）。

RCODE　应答码(Response code) - 这4个比特位在应答报文中设置，代表的含义如下：

0　　没有错误。

1　　报文格式错误(Format error) - 服务器不能理解请求的报文。

2　　服务器失败(Server failure) - 因为服务器的原因导致没办法处理这个请求。

3　　名字错误(Name Error) - 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。

4　　没有实现(Not Implemented) - 域名服务器不支持查询类型。

5　　拒绝(Refused) - 服务器由于设置的策略拒绝给出应答。比如，服务器不希望对某些请求者给出应答，或者服务器不希望进行某些操作（比如区域传送zone transfer）。

6-15　　保留值，暂时未使用。

NAME　资源记录包含的域名，**这个长度是不固定的，要和查询报文保持一致哦！**

TYPE　2个字节表示资源记录的类型，指出RDATA数据的含义　CLASS　2个字节表示RDATA的类

TTL　4字节无符号整数表示资源记录可以缓存的时间。0代表只能被传输，但是不能被缓存。

RDLENGTH　　2个字节无符号整数表示RDATA的长度

RDATA 不定长字符串来表示记录，格式根TYPE和CLASS有关。比如，TYPE是A，CLASS 是 IN，那么RDATA就是一个4个字节的ARPA网络地址。

请大家思考如下几个问题：

- 响应报文有多大？
- 响应报文每个字段填什么？三种情况的响应报文有何不同？
- 当配置文件里没有响应的url，我们又该如何获得DNS的响应报文？

1、构造DNS响应报文

当配置文件中包含DNS请求的域名的IP地址时，我们需要自行构造DNS响应报文并发送给客户端。

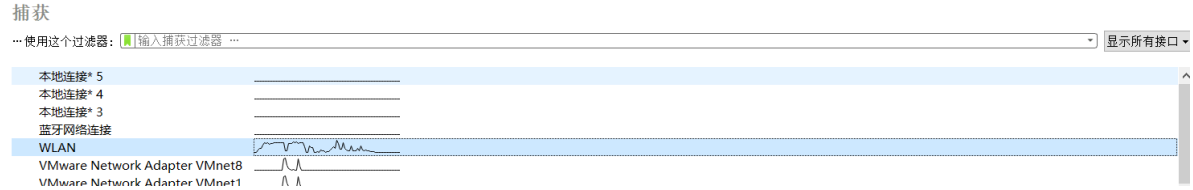
关于如何构造DNS响应报文，请同学们参考RFC1035，并重点关注第四章Messages部分内容。

如果同学们要想更清楚地了解实际的DNS报文的结构，可以使用wireshark抓包进行观察，具体操作如下：

1）下载安装wireshark：wireshark download

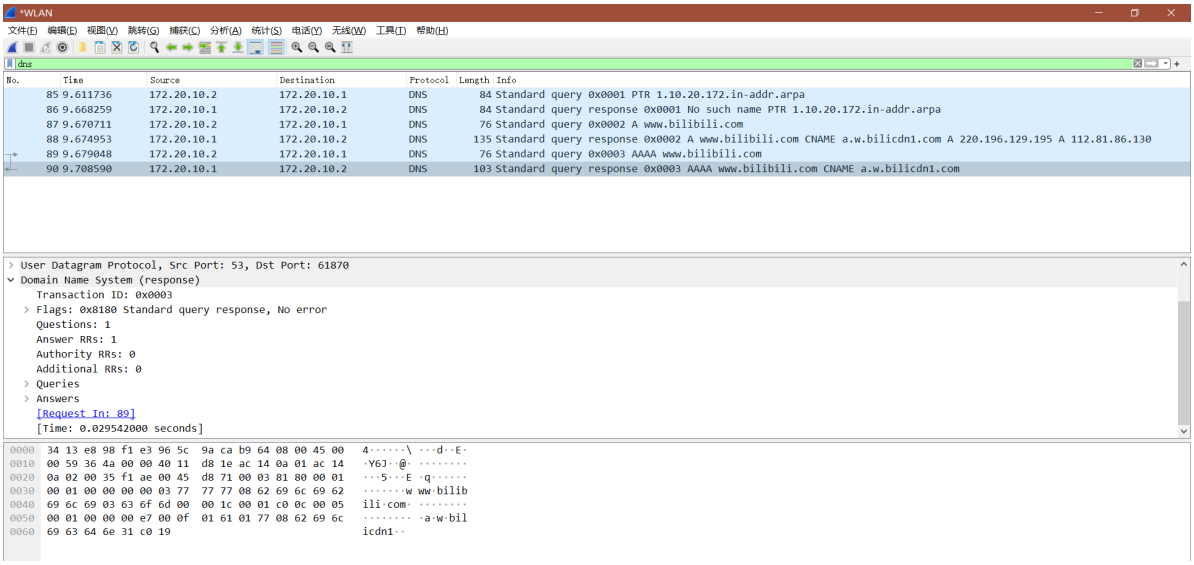
2）选择正确的接口进行抓包：

如下图所示，我们应该捕获WLAN接口的数据包



3）使用nslookup发出DNS请求

4）使用dns过滤器筛选出DNS报文：



注意

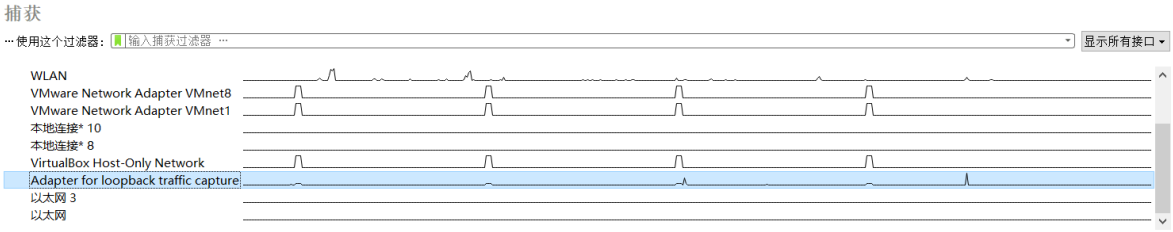
默认情况下wireshark是无法捕获本地环回地址127.0.0.1的数据包的。如果同学们希望捕获自己的dns relay server通过127.0.0.1发回给客户端的数据包，需要进行如下操作：

1)卸载winPcap和Wireshark。

2)下载安装Npcap。

3)重新安装Wireshark。

重新打开Wireshark可以发现多出了一个名为Adapter for loopback traffic capture的接口，从这个接口就能捕获同学们自己的DNS relay server从127.0.0.1接收和发出的数据包了。



2、DNS报文的中继

当本地的配置文件不包含DNS报文请求的内容时，我们的DNS relay server需要在客户端和外部的名称服务器之间提供中继服务，将客户端发出的DNS请求转发给外部名称服务器，并将外部名称服务器发回的响应报文发回到客户端对应的端口。

当我们的服务器收到响应报文的时候，要怎样确定这个报文应该发送到客户端的哪个端口呢?

一种思路是：构造一个key为ID，value为地址的字典/Map，当服务器收到响应报文时，首先提取出该报文的ID，然后从字典中便可以取得这个响应报文应该发往的地址了。

这里有一些常用的DNS服务器的地址：

DNS列表：详解国内常用的 DNS 服务器_IT专家网 (ctocio.com.cn)

大家选择的时候可以先nslookup测试一下：

```
PS C:\Users\mashu> nslookup -qt=A www.bilibili.com 223.5.5.5
服务器:  public1.alidns.com
Address:  223.5.5.5

非权威应答:
名称:    a.w.bilicdn1.com
Address:  223.111.221.130
Aliases:  www.bilibili.com
```

当我们生成了想要的报文，可以用socket.sendto()函数把报文发回去。

测试过程：

为测试同学们编写的DNS Relay Sever能否正常工作，请同学们分别用nslookup和浏览器访问 www.zhihu.com, www.baidu.com, www.test1.com 以及example.txt中不存在的随机外部网站，如www.bilibili.com。

注意到我们将www.test1.com被重定向到了IP地址127.0.0.1，为了正确完成此部分的测试，需要同学们在本地部署一个简单的Web服务器，以在windows上部署nginx服务器为例：

1、下载并解压nginx

2、打开powershell并移动到nginx.exe所在文件夹

3、在power shell中输入：

```
start nginx
```

4、此时我们用浏览器访问127.0.0.1即可看到nginx的欢迎页面

经过同学们编写的Dns Relay Server重定向以后，在浏览器输入www.test1.com就应该能转到本地运行的nginx服务器的欢迎页面了。

在用浏览器测试的时候，记得要修改自己电脑的默认dns服务器、清空浏览器和电脑的dns缓存，清空浏览器的图片缓存哦！

助教也是第一次写实验文档呢，如果有错误请不吝赐教，在群里及时指出问题！如有其他疑问还可联系：mashuang2@mail.ustc.edu.cn, szcw33@mail.ustc.edu.cn

一些知识

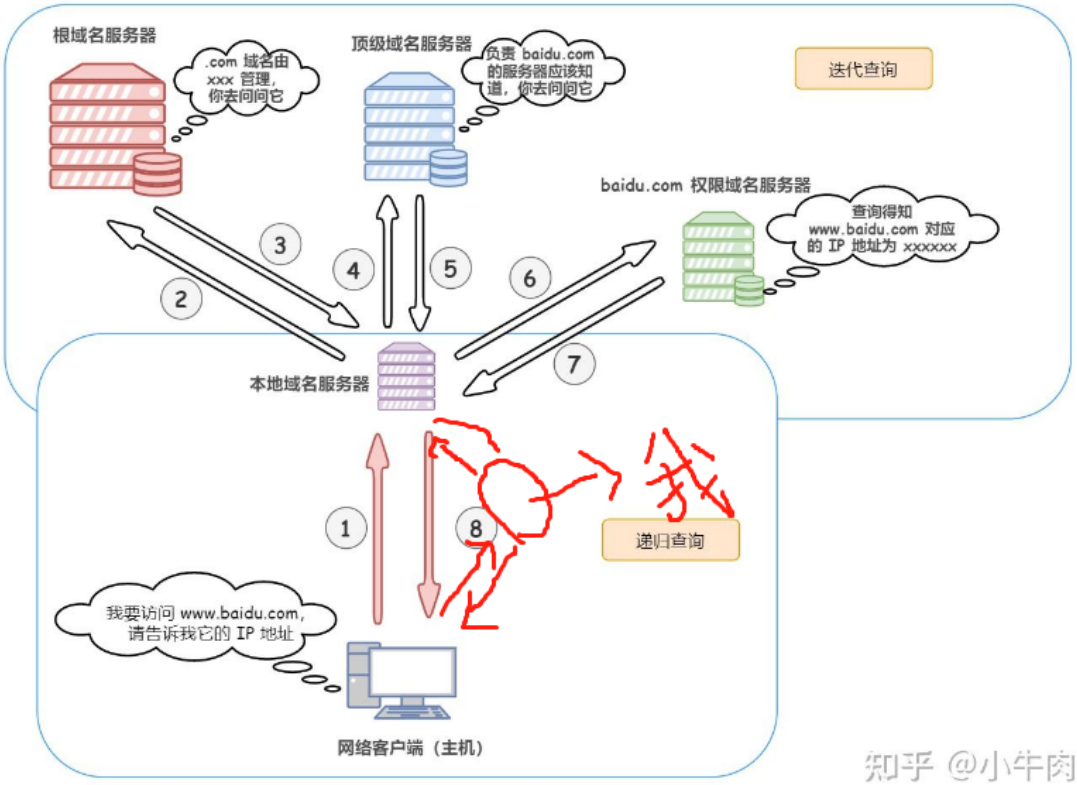
域名缓存

每个时刻都有无数网民要上网，那每次都去访问本地域名服务器去获取 IP 地址显然是不实际的。解决方法就是**使用缓存保存域名和 IP 地址的映射。**

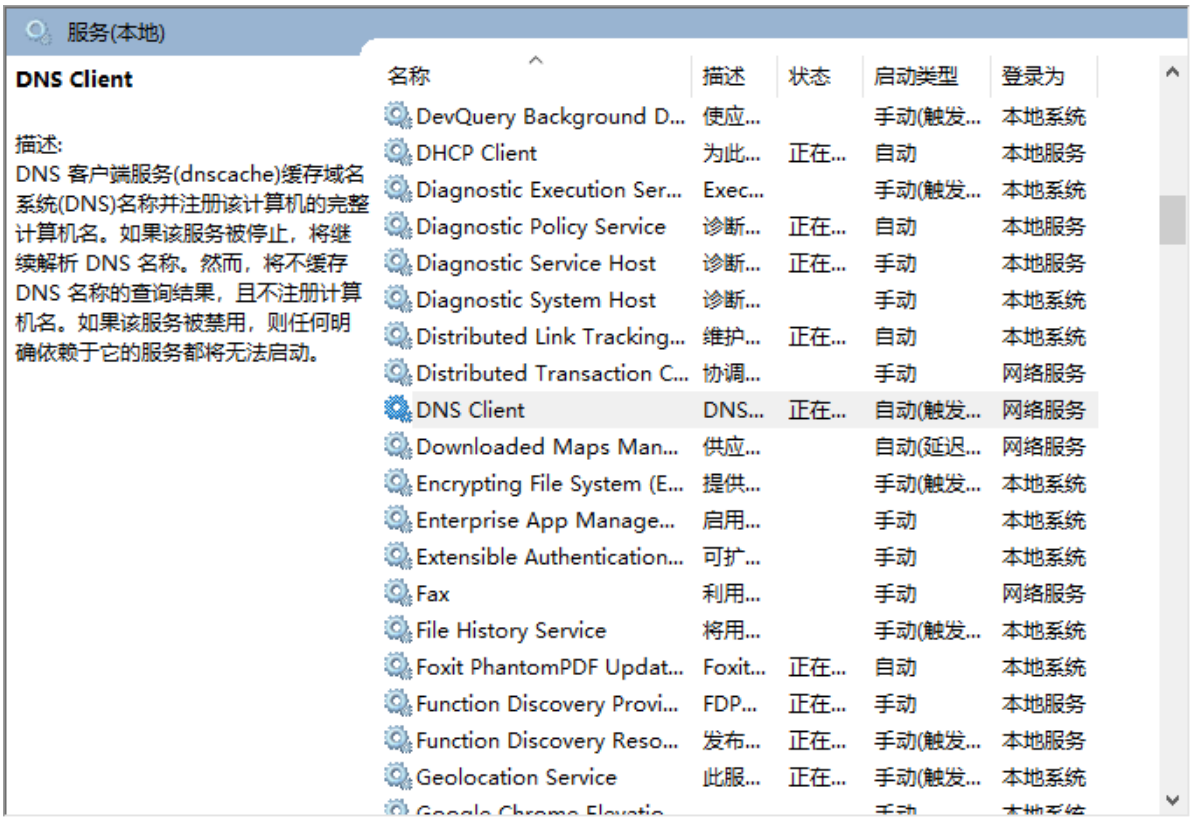
计算机中 DNS 记录在本地有两种缓存方式：浏览器缓存和操作系统缓存。

1）**浏览器缓存**：浏览器在获取网站域名的实际 IP 地址后会对其进行缓存，减少网络请求的损耗。每种浏览器都有一个固定的 DNS 缓存时间，如 Chrome 的过期时间是 1 分钟，在这个期限内不会重新请求 DNS

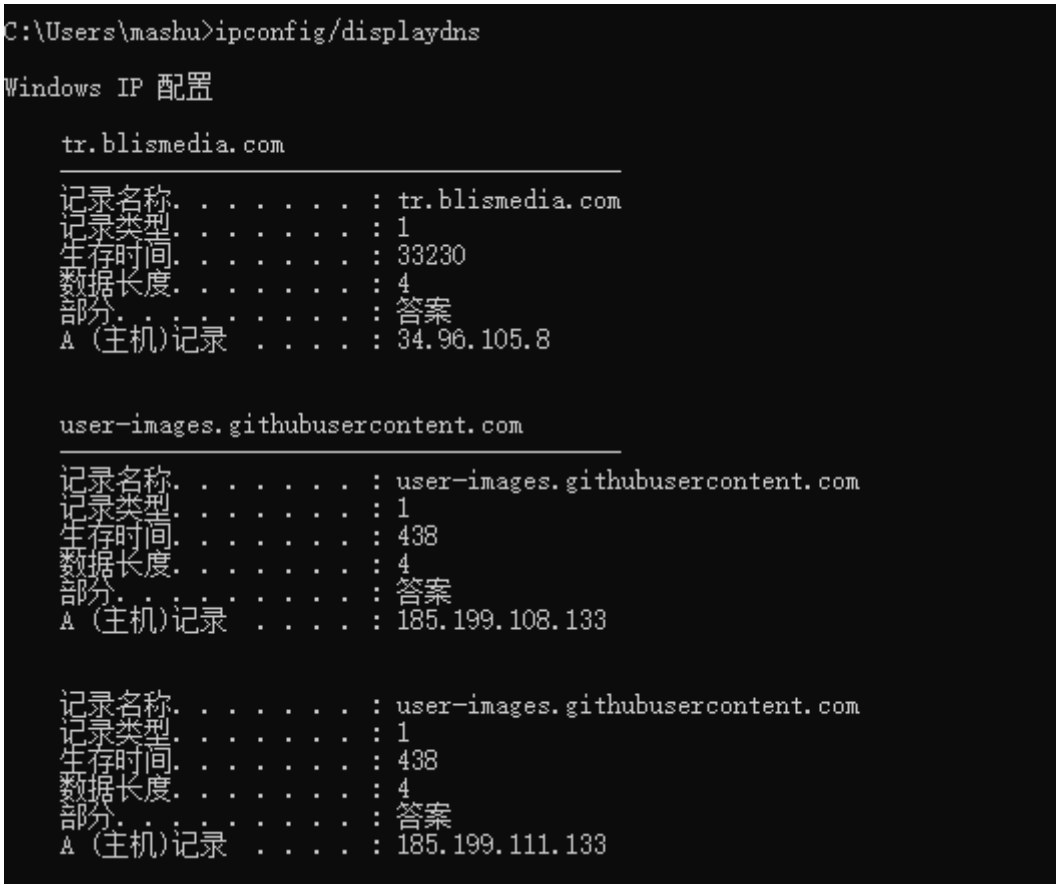
2）**操作系统缓存**：操作系统的缓存其实是用户自己配置的 hosts 文件。比如 Windows10 下的 hosts 文件存放在 C:\Windows\System32\drivers\etc\hosts



Windows 系统默认开启 DNS 缓存服务，服务名是 `DNSClient`，可以缓存一些常用的域名。



使用命令 `ipconfig/displaydns` 可以查看电脑中缓存的域名。



在浏览器中进行访问的时候，会优先查询浏览器缓存，如果未命中则继续查询操作系统缓存，最后再查询本地域名服务器，然后本地域名服务器会递归的查找域名记录，最后返回结果。**主机和本地域名服务器之间的查询方式是递归查询**，也就是说主机请求本地域名服务器，那么本地域名服务器作为请求的接收者一定要给主机想要的答案。

参考链接

nslookup命令详解*在努力！-CSDN博客*nslookup

超详细 DNS 协议解析 - 知乎 (zhihu.com)

DNS使用TCP还是UDP？ - cyjay5un - 博客园 (cnblogs.com)