**DNS中继服务器的实现**

王若楠：2015211700 叶浩葳：2015211685 刘明月：2015211694

一、系统概述

1、运行环境：Windows10

2、编译环境：Pycharm

3、编程语言：Python3.6

二、系统的功能设计

设计一个DNS服务器程序，读入“域名-IP地址”对照表，当客户端查询域名对应的IP地址时，用域名检索该对照表，三种检索结果：

（1）检索结果为IP地址0.0.0.0，则向客户端返回“域名不存在”的报错消息（不良网站拦截功能）

（2）检索结果为普通IP地址，则向客户返回这个地址（服务器功能）

（3）表中未检索到该域名，则向因特网DNS服务器发出查询，并将结果返回给客户端（中继功能）

考虑多个计算机上的客户端会同时查询，需要进行消息ID的转换

三、模块划分

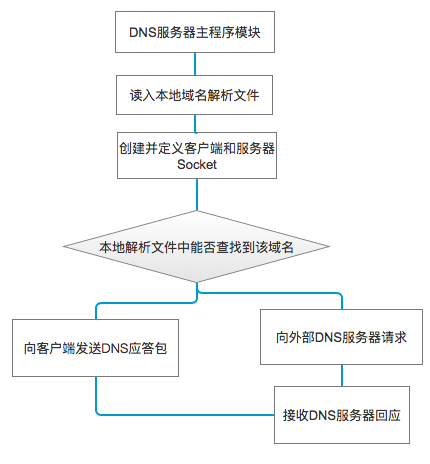
DNS服务器主模块包含三个子模块，分别如下：

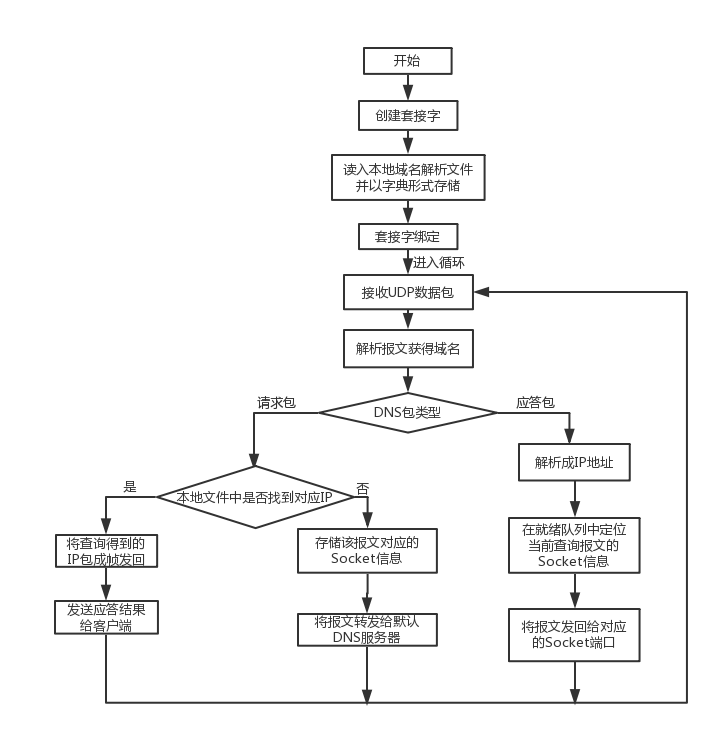
1、命令行参数处理模块：该模块用来处理通过命令行提示符来启动这个DNS服务器时所输入的命令行参数，管理员通过设置不同的参数可以使DNS服务器显示不同程度的提示和调试信息。所以这模块主要是依照输入的参数设置标志数据，以控制最后的各种信息的输出。

2、本地解析模块：本模块是DNS服务器本地保存的曾经解析过的或者需要屏蔽的域名和其对应IP信息,在文件中查找从应用程序来的请求解析的域名，在这个文件中查到需要的域名后取出对应的IP地址，并构造DNS应答数据包返回给发送此DNS域名解析请求的应用程序。

3、外部DNS服务器解析模块：当本地解析失败时，本DNS服务器会调用外部DNS服务器解析模块。此模块将应用程序发送的DNS请求报文转发给外部DNS服务器，然后接收外部服务器返回的应答信息，并根据这个信息给予应用程序相应的DNS应答。

以上三个模块与主模块的关系如下：



四、软件流程图

五、测试用例以及运行结果

调试命令及功能：

命令语法：**dnsrelay** [**-d** | **-dd**] [***dns-server-ipaddr***] [***filename***]

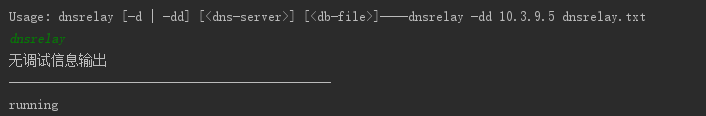
[dns-server-ipaddr]:10.3.9.5

[filename]:dnsrelay.txt

dnsrelay：调试信息级别为0，无调试信息输出

dnsrelay –d：调试级别为1，输出时间坐标，序号，客户端IP地址，查询的域名

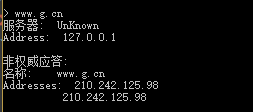
dnsrelay –dd：调试级别为2，输出冗长的调试信息，



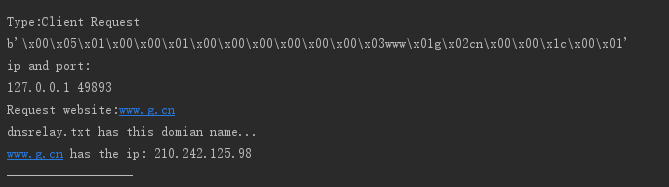
1、本地解析

请求一个本地DNS对照表中已有的记录：

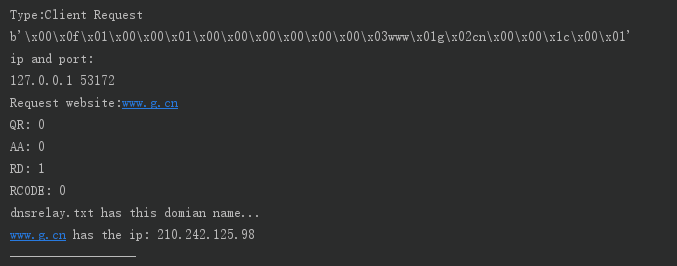
210.242.125.98 g.cn



dnsrelay –d:



dnsrelay –dd:



在本地DNS对照表中找到记录，将对应的IP地址发回给客户端

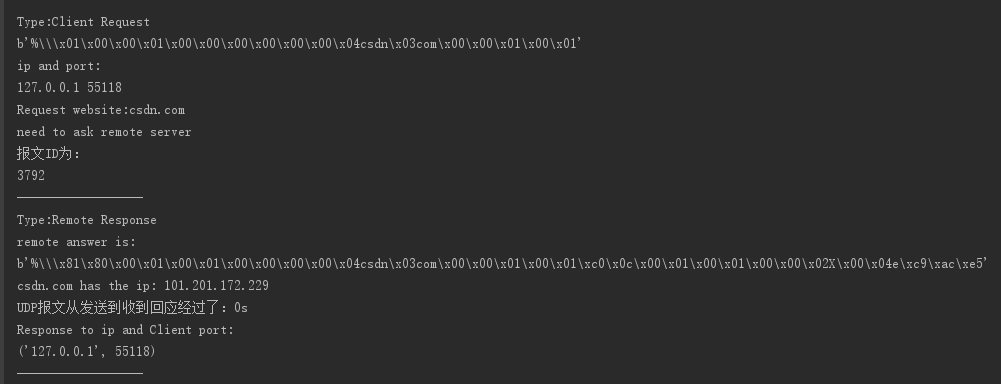
2、中继功能

请求本地DNS对照表中没有的记录，如csdn.com。

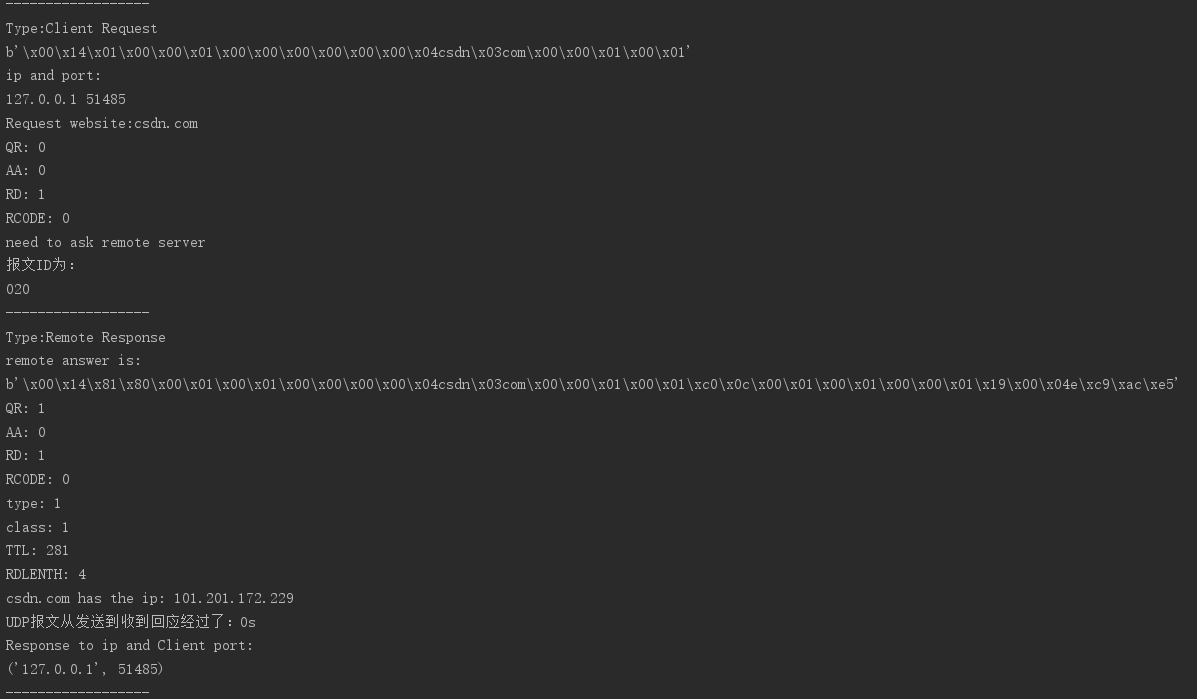
此时，本地DNS服务器没有找到对应记录，于是向外部DNS服务器转发请求，接收对应的应答报文并将IP地址解析出来发送给客户端。



dnsrelay –d:

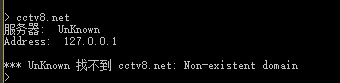


dnsrelay –dd:

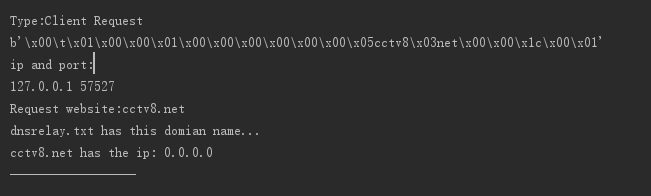


3、不良网站拦截

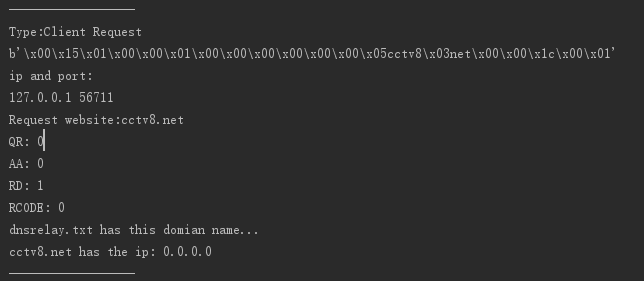
请求一个DNS对照表中的不良网站，如：cctv8.net 0.0.0.0



dnsrelay –d:



dnsrelay –dd:



4、超时处理

由于发出查询请求到收到应答报文之间存在一定时长，所以设定阈值来判断是否超时。

方法一：

当确定本地DNS对照文件中不存在该域名时开始计时，当用户收到应答信息时停止计时，计算时间差来判断是否超时。

超时分为两种情况：

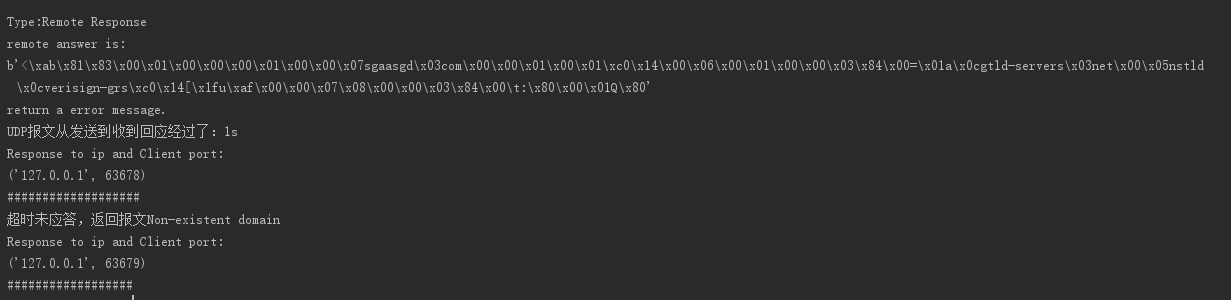
1. 不存在此域名
2. 查询时间过长，但此域名存在

为了展示超时处理效果，我们将阈值设定为0s：

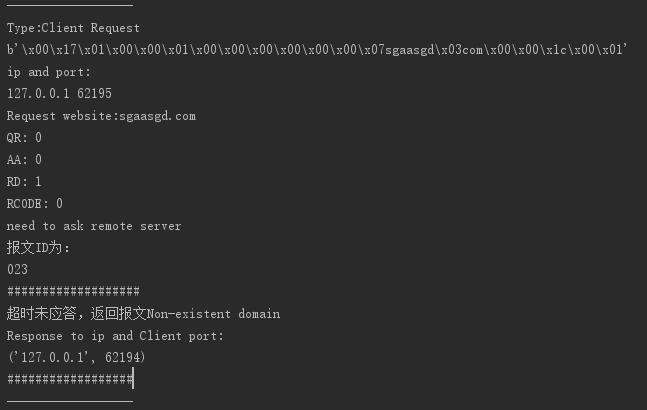
请求一个不存在的域名：sgaasgd.com



dnsrelay –d:



dnsrelay –dd:



方法二：

利用Socketserver模块，客户端每次发出域名查询请求，就开启一个线程实现域名的解析。超时处理：当确认本地解析文件中不存在该域名时，开始计时，中继方法与方法一相同，设定超时阈值为3s，程序运行图与方法一相同。

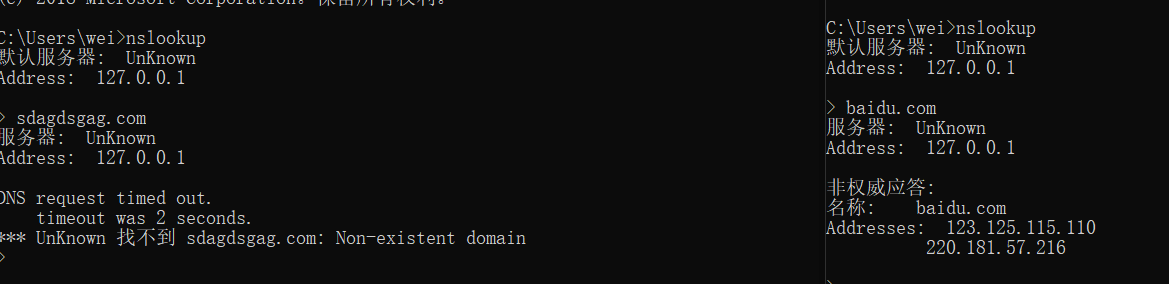
5、多用户同时查询的实现

当客户端发出查询报文时，若本地解析文件中不存在当前域名，则将其IP存储到等待处理的数组（client\_wait[]）中，并设置全局变量key\_records，自增来实现对其编号，建立字典来存储IP与编号的对照关系，当收到回答报文时，依次发回给对应的主机和端口。

多客户端并发：

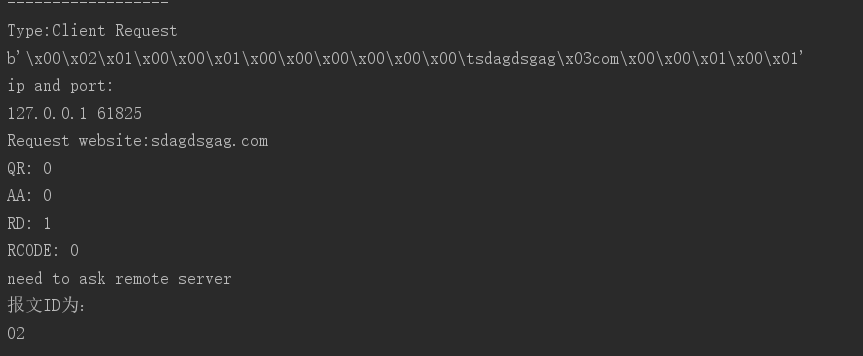
允许多个客户端（可能位于不同的多个计算机）的并发查询，允许第一个查询尚未得到答案前就启动处理另外一个客户端查询请求。

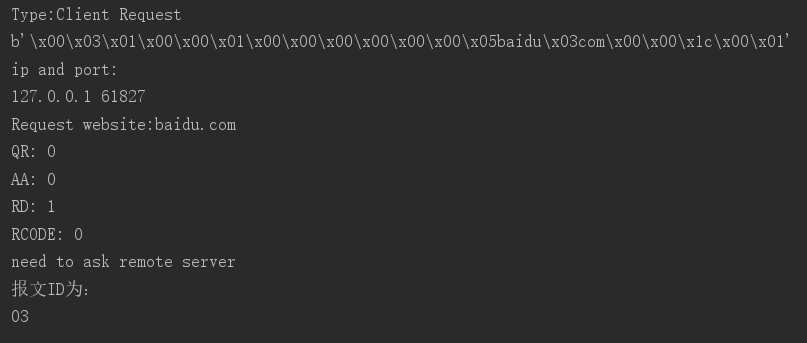
当前先查询一个不存在的域名：tsdagdsgag.com，在未返回查询结果时开启另一个cmd中继查询baidu.com:



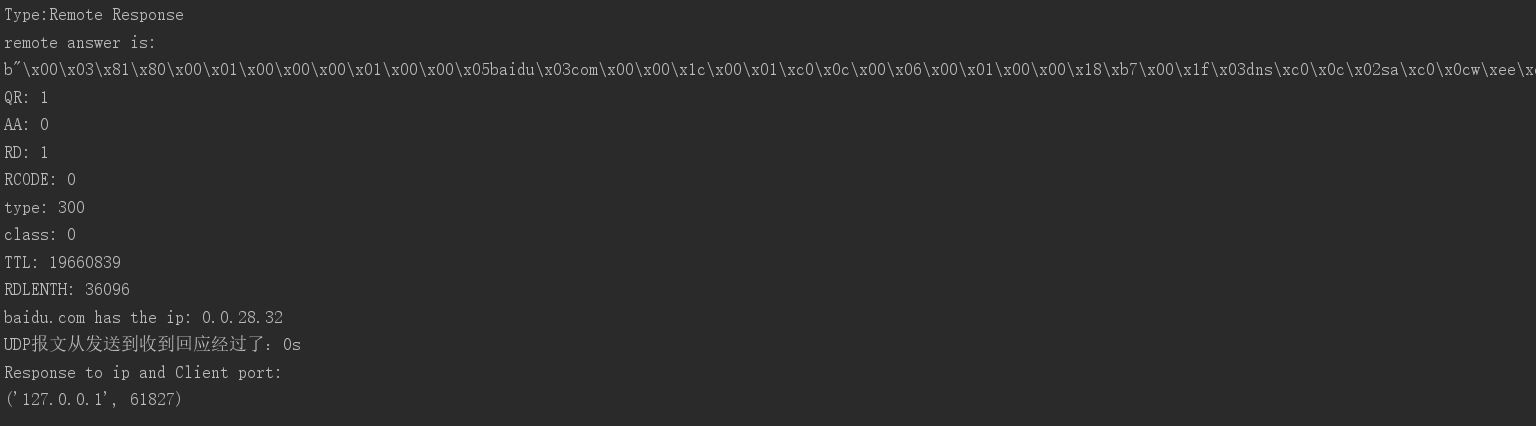
此时优先返回baidu.com对应调试信息，之后返回tsdagdsgag.com的信息：

tsdagdsgag.com与baidu.com查询：

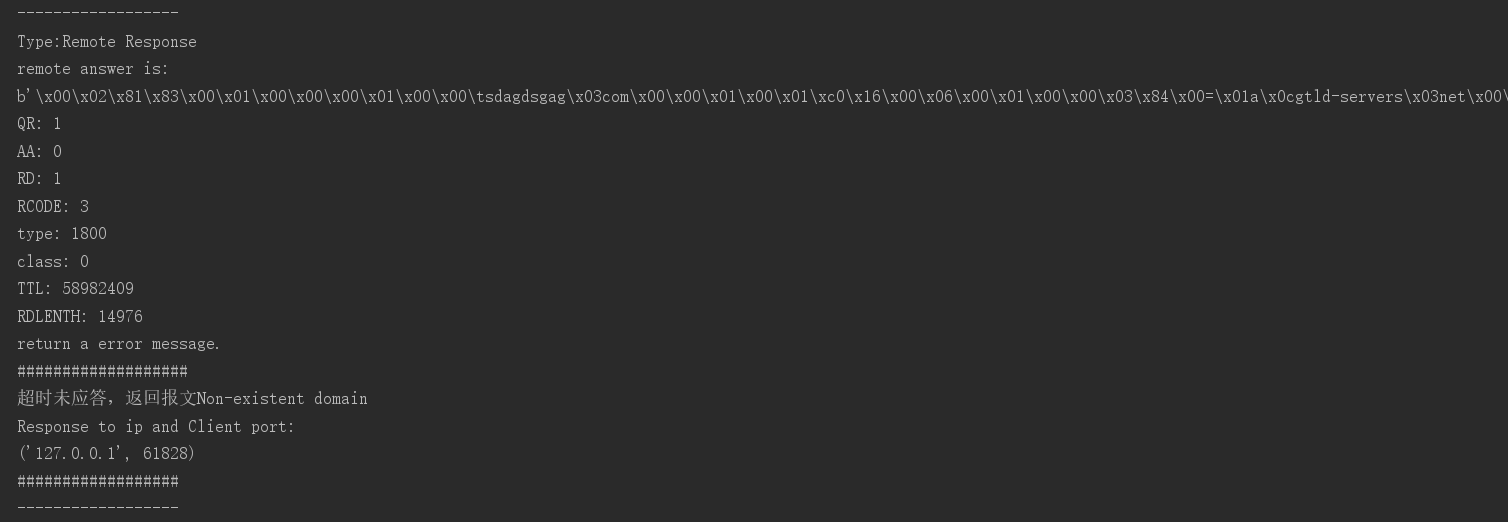




baidu.com应答：



tsdagdsgag.com应答：



六、调试中遇到并解决的问题

1、请求包的域名还原问题，要注意数组下标与实际位数的区别。

2、[Python](http://www.169it.com/tag/python) 3对文本和[二进制](http://www.169it.com/tag/%E4%BA%8C%E8%BF%9B%E5%88%B6)[数据](http://www.169it.com/tag/%E6%95%B0%E6%8D%AE)作了清晰的区分。文本总是[Unicode](http://www.169it.com/tag/unicode)，由str类型表示，二进制数据则由bytes类型表示。[Python](http://lib.csdn.net/base/python) 3不会以任意隐式的方式混用str和bytes，正是这使得两者的区分特别清晰。一开始的时候一直弄混了这两者，在Python中默认的是str，但是网络中UDP报文需要是字节流，需要进行转换。

3、一开始制帧的时候，不知道应该修改哪些位、以及不知道应该怎样将解析得到的IP地址加在返回报文的后面，仔细得研究了DNS报文格式和报头字段后，解决了这个问题。

4、在抓包的过程中发现客户端的每一次查询都会发送一个type为ipv4和一个type为ipv6的两个查询报文，但是ipv6的查询报文的格式与ipv4报文不一致，若做相同的处理会发现输出的转换ip不正确，所以在程序中加入了对于ipv6查询报文的过滤处理。

5、在实现用户多并发时，无法在其他计算机上实现，但是在同一台主机上多个命令行窗口可以实现。

七、心得体会

刘明月：此次作业我们组员三个协作完成了DNS中继服务器的实现。通过完成本地域名解析、中继功能、不良网站拦截、超时处理、多并发等任务，来实现客户端对外部服务器的访问需求。在任务前期，我们阅读了RFC 1035文档，并借助网络和资料熟悉了DNS的原理，从而对其实现方法有了更深刻的理解。在完成此次作业时，我们更加熟悉了Socket的基本操作与部分函数，通过抓包分析UDP数据报文、DNS数据报文，我们对传输层和应用层的数据包发送和接收工作原理有了更清晰的认识。

此次作业用Python语言实现，所以语法方面我们遇到的障碍并不是很多，这让我们更快地掌握了DNS的工作原理与实现方法。这次实验也让原本只存在于书本上的知识得到了实践，让我收获颇丰。

王若楠：通过此次协作完成DNS中继服务器，对DNS的功能和作用有了更加直观的理解。前期查阅资料（如RFC 1035、作业要求PPT），对DNS有了基本的了解以后，组内分工通过Python实现了DNS的本地查询、转发、不良网站拦截、计时等功能，在这个过程中，加深了对于网络、UDP、socket的认识，并且熟悉了Python的Socket模块。在分析包的过程中，我们还采用了wireshark抓包对包进行详细分析，通过查看wireshark抓包的结果使得更好的理解了UDP报文。该次实验使得自己对于计算机网络有了更深入的理解，同时再次体会到了团队合作的乐趣。

叶浩葳：其实在本次实验的开始前，光有着上学期所学到的计算机网络理论知识，对于这次实验的实操细节我是很迷茫的。只有在大体上的关于dns处理器对于域名请求处理的概念，比如：域名->dns->IP。仅此而已，对于内部的细节操作可以说一问三懵逼。但是在组队合作的攻坚过程中，通过对相关资料的检索查询，队友间的讨论交流，我脑海中对于应用层协议dns有了一步一步的深刻认识。借由RFC1035和抓包的结果进行比对，明白了UDP报文的每一个字节的作用，并在每一个字节的变化会造成的后果都有有过一遍体验。可以说这次实验帮助我走进了计算机网络这一宏大领域的内部，而并不只是想上学期一般驻留在外部的远观。