DNS中继服务器设计报告

2017211310班 2017211395 熊智宇

1. 系统的功能设计：

* 读入“IP地址-域名”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，有三种可能检索结果：
* 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）
* 普通IP地址，则向客户端返回该地址（服务器功能）
* 表中未检到该域名，则向因特网DNS服务器发出查询，并将结果返给客户端（中继功能）

实验环境：

操作系统：Windows 10

环境：Python 3.7

IDE：PyCharm Community

1. 模块划分：

存储：用全局变量list namemap，服务器运行时先把dnsrelay.txt全部导入namemap。

函数模块：

DNS Server：

建立字节流的套接字，建立多线程，监听接口并获取DNS包

DNS UDPHandler：

1. 载入字典列表。

2. 处理获得的包，并对其进行解析。

3. 针对解析的不同结果，选择屏蔽、查询或转发。

4. 构造完整的DNS应答包，向客户端发送。

DNS Header：

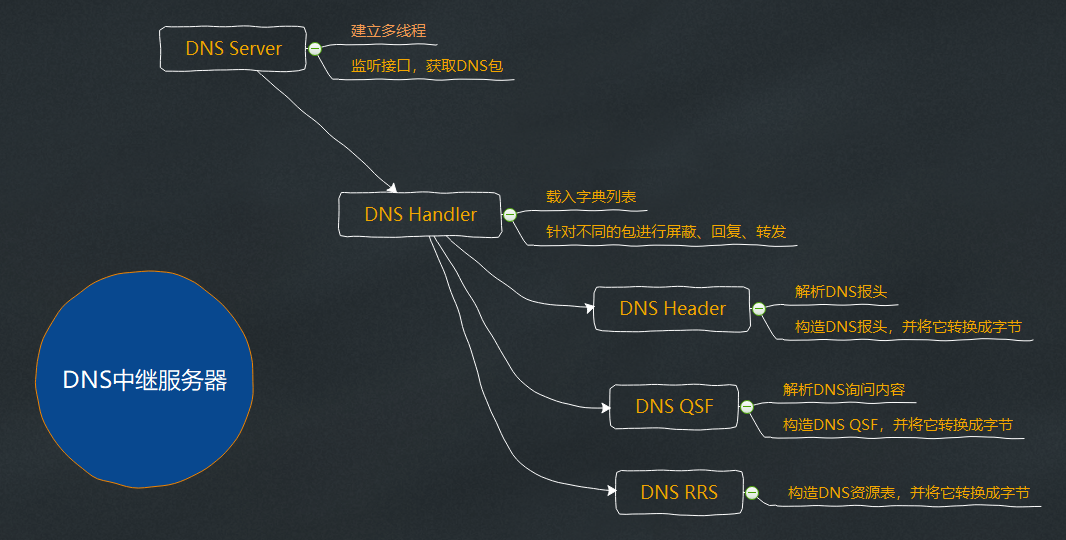
根据输入的报头信息，构造DNS应答包的报头字节

DNS QSF：

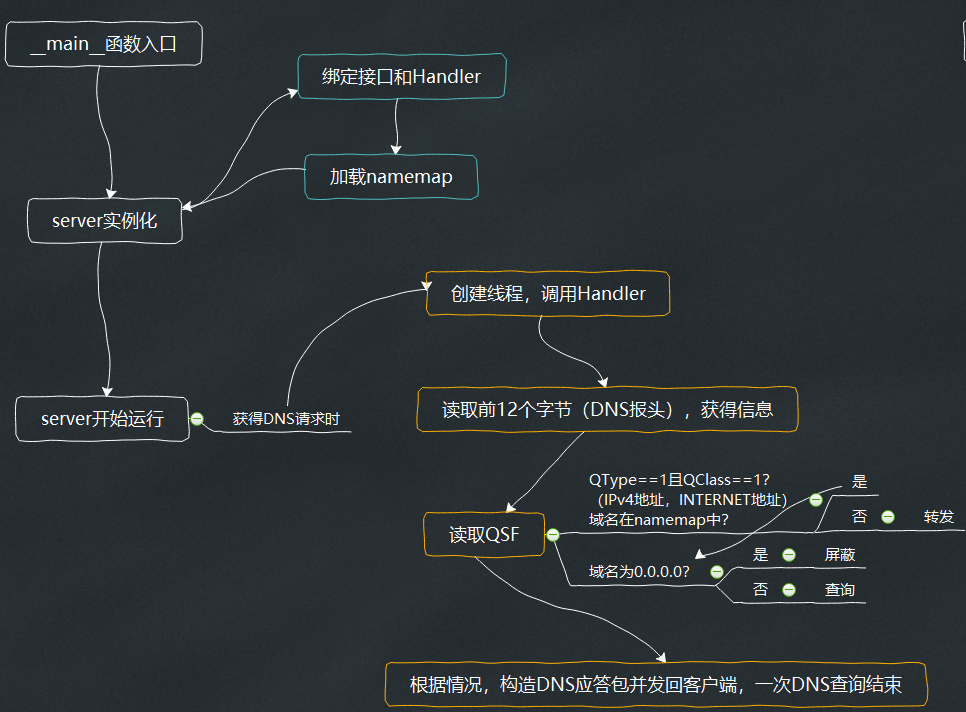
根据输入的询问表信息，构造DNS应答包的QSF字节

DNS RRS：

根据输入的资源表信息，构造DNS应答包的RRS字节



1. 软件流程图：



1. 测试用例以及运行结果：

将DNS服务器设置为127.0.0.1，在程序内设置转发目标DNS服务器为114.114.114.114，在python命令行环境下运行程序。

打开命令行窗口。分别测试屏蔽、查询和转发功能。

1. 屏蔽功能：选择屏蔽列表中如下网址进行测试：

youlove.3322.net

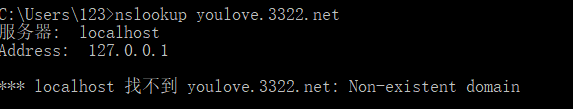
test0

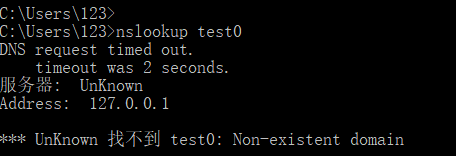
zhendeshi.tiebiechang-tiebiechangde.zuichang255zifu.domain-name.com

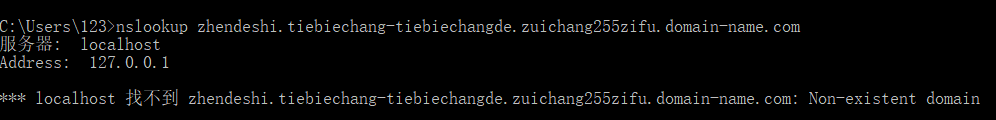
[www.51bug.com](http://www.51bug.com)

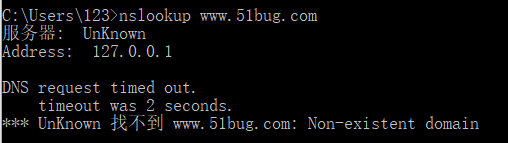
运行结果如下：

客户端：







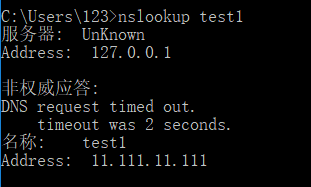


结果表明，DNS屏蔽功能有效。

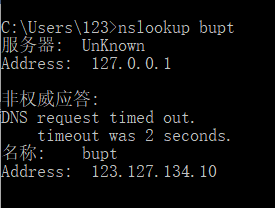
1. DNS服务器功能

选择如下在DNS中存在的地址进行测试：

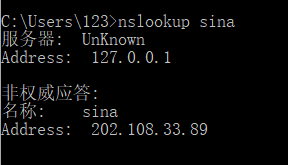
test1



Bupt



Sina



结果表明DNS查询功能有效。

1. DNS中继功能：

选择如下在DNS中不存在的地址：

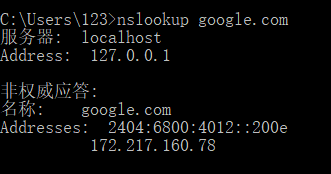
Baidu.com

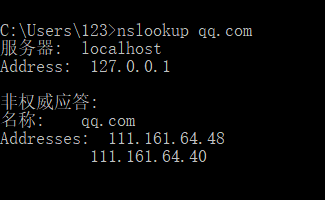
Google.com

qq.com

测试结果如下：

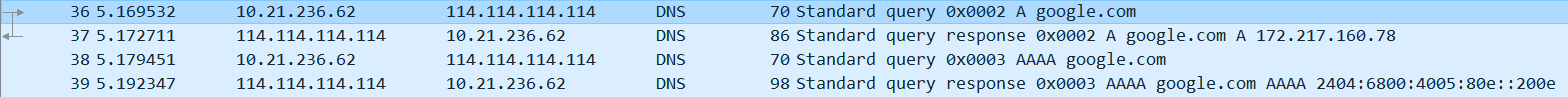


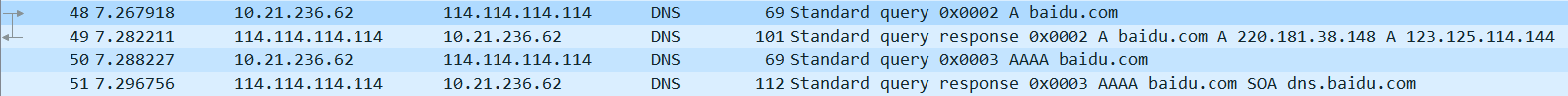




使用WireShark抓包表明，服务器转发并收到了应答信息，证明DNS转发功能有效。







1. 调试中遇到并解决的问题：

字符串编码问题：一开始使用struct.pack函数把数据打包成字节流，但由于QSF和RRS的长度是变化的，而pack函数需要输入格式字符串fmt，所以不便使用；解决方法：对字符串编码时，使用encode、chr、ord、pack等多种函数混合。

多线程问题：一开始使用的是UDPServer，但UDPServer是单线程处理，一旦出问题则服务器无法运行。而且处理速度也很慢，经常导致超时；解决方法：使用继承多线程处理模式的ThreadingUDPServer。这个方式同时解决了消息ID问题，因为是针对每个包建立线程操作的。

1. 课程设计工作总结：

在之前的学习中，我已经认识和使用过了TCP协议，但这次我使用的是UDP协议。在开发和测试过程中，我们发现，UDP协议是一个无差错控制的，不可靠的协议，经常出现丢包、超时等问题。我使用了多线程的模式对数据进行隔离发送，既使程序的可靠性和性能都得到了提高。

DNS中继服务器的设计中，对报文的处理占了极大的内容。因此，我选择了较为灵活的Python语言进行这次课设。由于Python的灵活性，在初期程序设计中数据结构产生了一些混乱。这使我认识到，在开发前期要重视软件架构的设计。好的前期设计可以大大节省开发的工作量。