

**课程设计报告**



**题目：计算机网络课程设计（DNS Relay Server）**

**实验报告**

**学号-姓名： 2017211357 付彬**

**2017211356 韩昊轩**

**2017211355 王牧峰**

**学 院： 计算机学院**

**2019年 5月**

1. **系统的功能设计**

**设计一个DNS中继服务器程序，可以读入“IP - Domain Name”（dnsrelay.txt）对照表，当客户端查询到对应的IP地址是，用域名检索该对照表，分别完成如下三种功能：**

* + - 1. **普通检索结果：在对照表中有存在对应的IPV4地址，向Client返回它。例如：test1应对应返回11.111.11.111。**
      2. **屏蔽：在对照表中IP地址为0.0.0.0，需要进行屏蔽，即不良网站拦截功能。**
      3. **中继：在表中未检测到该域名，需要向Internet DNS Server进行查询，并将结果接受返回到Client。**
         * **多个计算机的客户端同时查询，需要进行相应的ID转换。**

1. **协议分析**

**【报文格式与分块】**

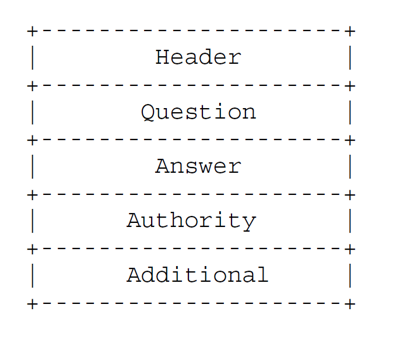


图 1 报文格式

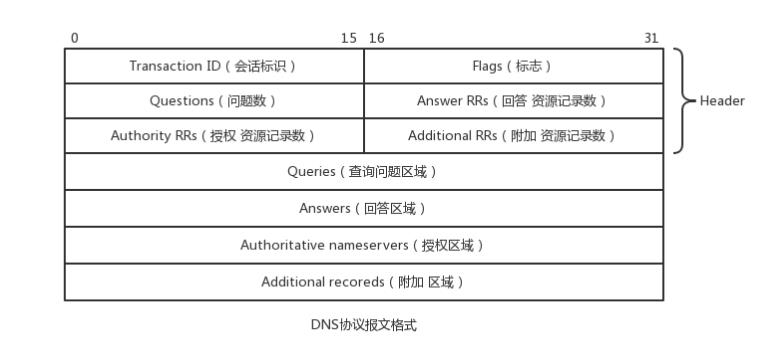


图 2 报文分块

* 1. **Header**

**【Header格式】**

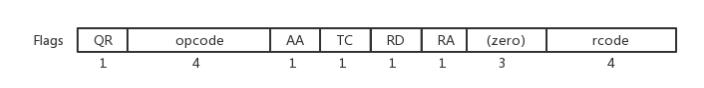


图 3 Header格式

**【关于Header-ID】**

* 若数据报从客户端发出，我们需要将ID记录下来。

**【关于Header-Flags】**

|  |  |
| --- | --- |
| QR/1 bit | QR==0时为查询报，QR==1时为响应报 |
| OPCODE/4 bits | **0为标准查询，1 为反响查询，2 为服务器状态请求** |
| AA/1 bit | **通过AA可以确定是否有权威答案（Authoritative Answer）** |
| TC/1 bit | **可截断的（Truncated）** |
| RD/1 bit | **期望递归（Recursion Desired）** |
| RA/1 bit | **递归可用（Recursion Available）** |
| Z/3 bits | **保留字段，全为0** |
| RCODE/4 bits | **0 为无差错，3 为名字差错** |

* 当报文是响应报时，通过AA可以确定是否有权威答案（Authoritative Answer），并且可以通过RCODE可以确定有无差错。当RCODE==0时，表示无差错，正确传输IP地址。当RCODE==3时，表示有差错，其中在实现我们的中继服务器时，**需要利用RCODE==3来进行屏蔽功能。**

**【关于Header-Quantity字段】**

* Questions、Answer RRs、Authority RRs、Additional RRs 各自表示后面的四个区域的数目。Questions表示查询问题区域节的数量，Answers表示回答区域的数量，Authoritative name servers表示授权区域的数量，Additional records表示附加区域的数量。
  1. **Question/Queries**

**【Queries格式】**

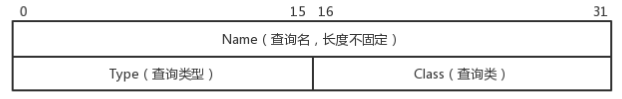


图 4 Question/Queries

**【关于Queries】**

* Name不定长，用字节填充法进行分割，并以‘\0’结尾：

**e.g. 008.cn : 03 30 30 38 02 63 6e 00**

* Type中常用的有：
  + PTR（12）：将IP地址转化为域名；
  + A（1）：IPV4；
  + AAAA（28）：IPV6;
* Class通常为0001，表示来自Internet。
  1. **RRs（包含Answer、Authority、Additional）**

**【RRs格式】**

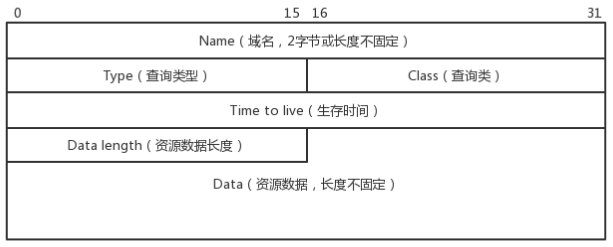


图 5 RRs

**【关于RRs】**

* 前面会重复Queries的部分；
* TTL（生存时间）：以秒为单位，表示的是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。
* DATA Length：表示数据的长度，当为IPV4时，DL是0004。
* DATA：计算IP的16进制表示。

1. **模块划分**
   1. **UDP-SOCKET**

* 使用UDP模式的socket，实现DNS中继服务器的功能。程序中仅使用一个socket，接收数据报后，判断下一步的工作，然后发送给应发送的地址。
  1. **报文处理**
* 在dnsrelay.txt中存在的IP-域名需要自己构造一个报文发送给Client端。可分为两类，一是查询到正常发送；二是需要屏蔽：
  1. 正常发送
     + 首先构造Header：从recvData中将Header提取出来，并且修改响应标志位，使其变为QR==1（响应报），AA==1（有权威），RCODE==0（正常）。
     + 进而构造Queries部分和Answer的前半部分：首先从recvData的第12byte开始搜寻第一个‘\0’（即0x00，此时不定长的Name已经完全拷贝）,并将其后面的00 01 00 01一同拷贝到Queries和Answer部分。
     + 构造Answer的TTL、DATA Length和IP部分：TTL固定为0x78，长度由于是IPV4，固一定为0x04，所以写如下字符串“string TTL\_L = {0x00,0x00,0x00,0x78,0x00,0x04};”，将其加到上一步构造的Answer后面。最终利用get\_ip( );函数将ip地址进行相应的处理，并加到构造报文的最后。至此我们构造好了正常返回的报文。
     + 注意：在实际程序书写中，为了方便相加，我们进行了从char[ ]到string再到char[ ]的转化。
  2. 屏蔽
     + 屏蔽报文构造相对简单，只需做如下更改“ recvData[2] = (char)0x85; recvData[3] = (char)0x83; ”即可。 其中RCODE为3，做到了屏蔽功能。
* Debug模式的打印：
  1. 打印报文流：利用printf( )和转义符%x

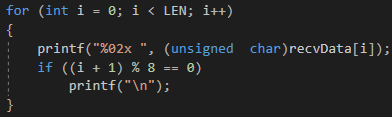


图 6 Debug报文流打印

* 1. 提取Header中Flag的信息：利用掩码，将报文流中相应的每一比特与掩码相与（&运算），判断相与之后的值，并将对应的标志位设成响应的值。关于掩码的计算，与Flag中每个标志位的比特长度有关，例如：OPCODE为四位，并且在一字节（0-7）的1、2、3、4位，故其响应的掩码应该为0x78。其他的掩码计算方式类似。

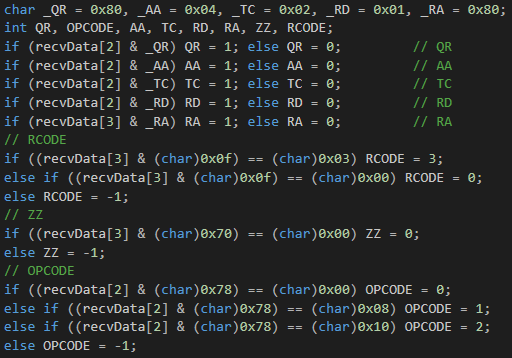


图 7 Debug Flag信息与掩码

* 1. 根据header结构体打印相关信息，并且获取发送端信息以及接收端信息：利用Get\_Header( )函数将Header其他信息捕捉并且存储，直接打印即可。发送、接收端信息直接打印对应接口。

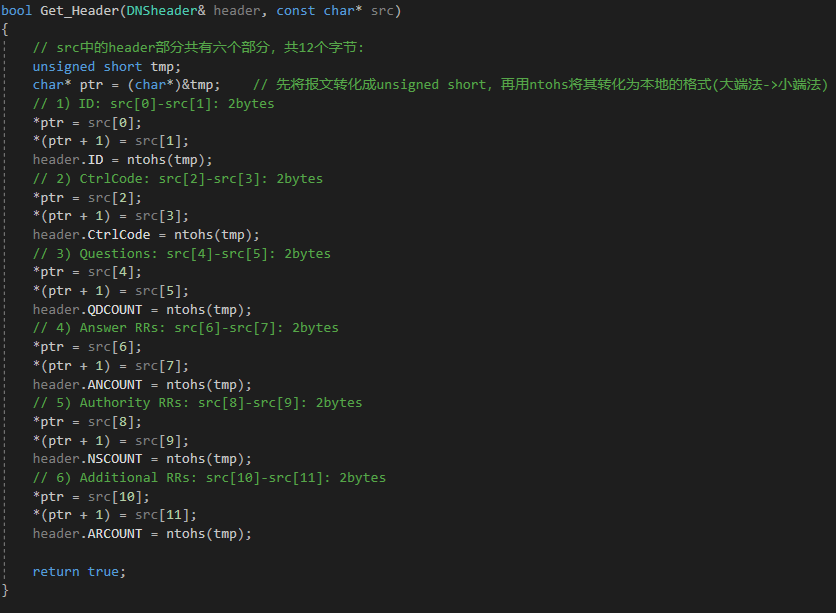


图 8 Get\_Header Function

* 1. **文件读写与IP查询**
     + 程序初始化：用于读取用户指定的调试信息输出等级、服务器地址、对照表文件。

利用int main(int argc, char\* argv[ ]){ }的两个参数，记录程序调试参数的个数和以字符数组形式存储所有程序调试参数的数组。

通过判断argc的值来确定调试参数的所有可能种类，以防止访问溢出。

argc的取值及相应判断详见代码及注释。同时，依据参数的值确定调试等级、修改默认服务器地址和默认对照表文件名。若程序调试参数超过应在的区间，则会置调试等级为-1。

将对照表文件读入全局自定义结构体类型容器LocalList存储，结构体的两个字符串成员分别存储IP地址和域名。

输出一些常规初始信息，依据调试等级选择性输出一些初始信息。

1. **软件流程图**



1. **测试用例以及运行结果**

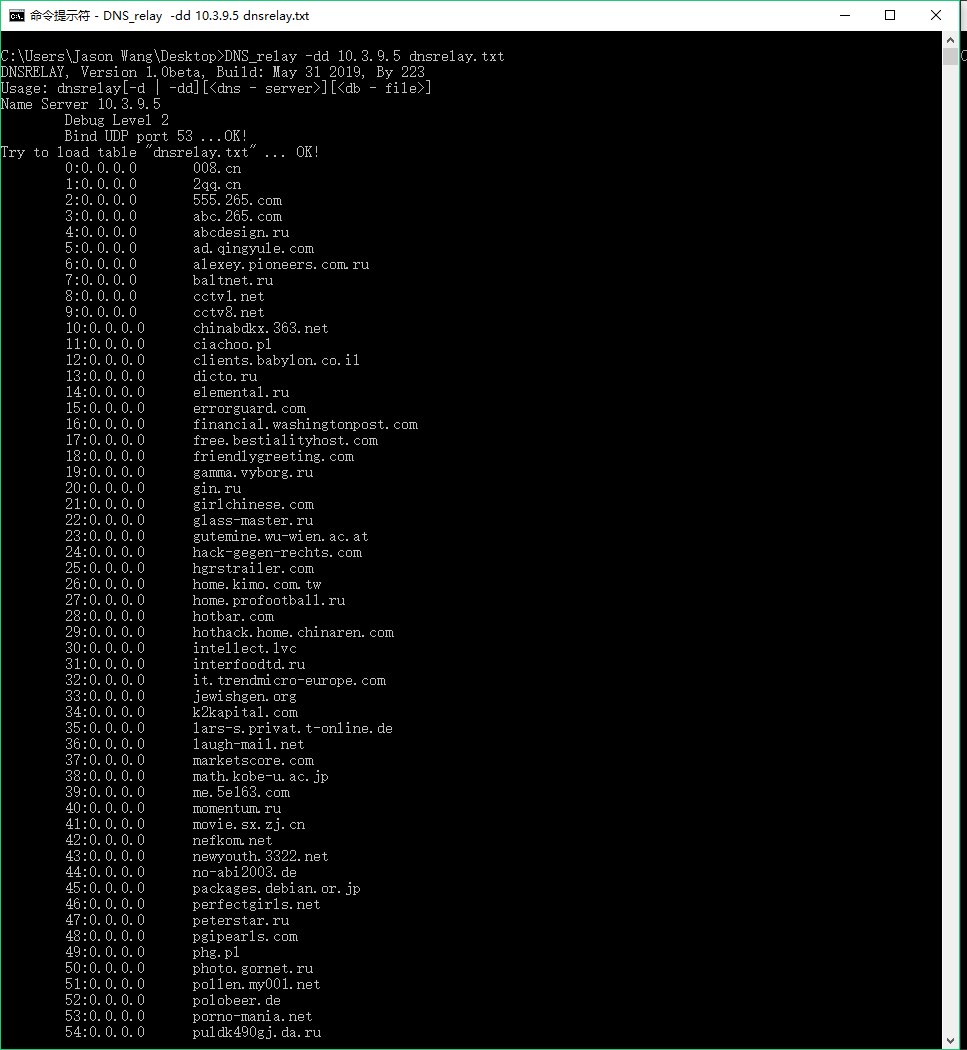
****

图 9 打开界面

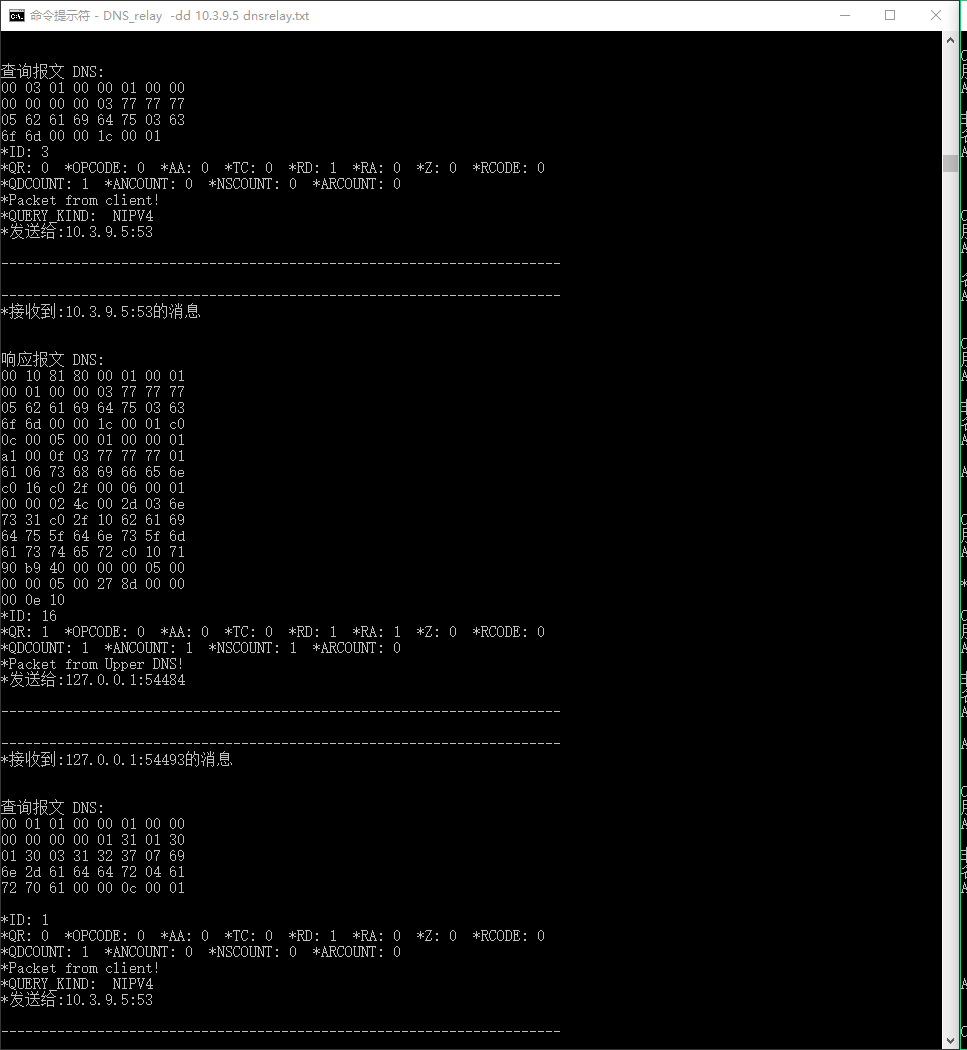


图 10 报文1

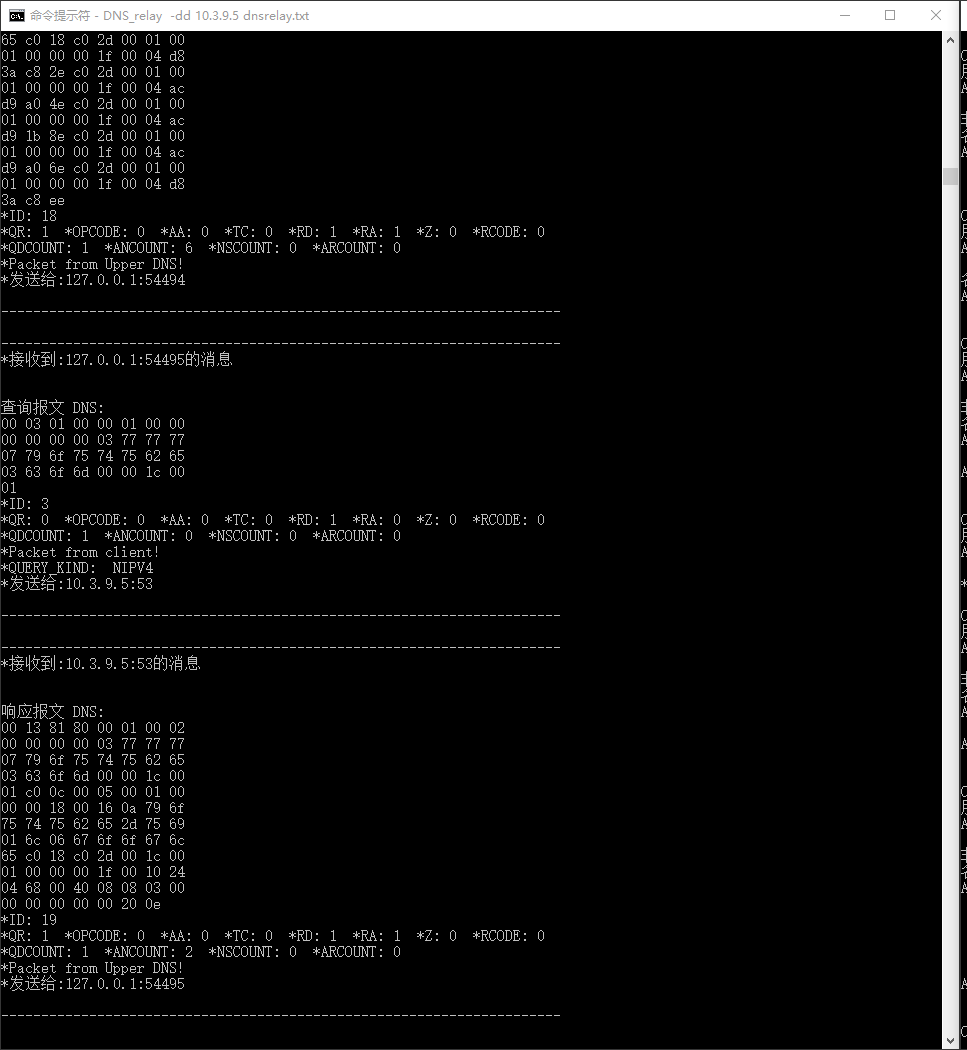


图 11 报文2

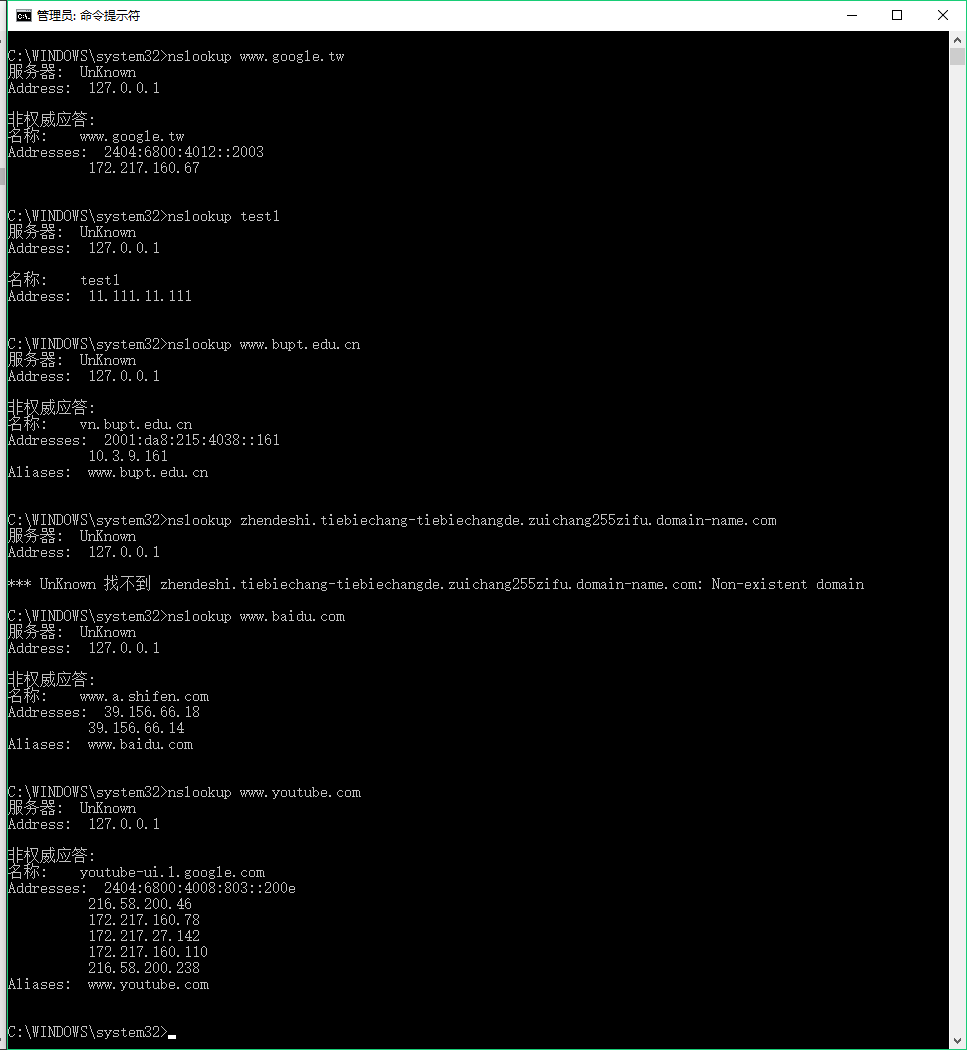


图 12 使用

1. **调试中遇到的问题与解决方案**
2. 数据报无法正常接收：发送数据报时发送了整个缓存区，应利用好recvfrom函数的返回值。
3. 中继的数据包无法收发：在socket绑定时出现问题，同时判断数据报来源是否为原DNS的条件判断书写错误。
4. 无法实现中继功能：用户的ID设置有问题，应自定义分配给需要中继的数据报独一无二的ID。
5. **课程设计工作总结**

在这次课程设计中，我们了解了RFC1034，RFC1035以及UDP-socket编程，充分地理解了DNS服务器的原理、DNS中继服务器的实现。

在程序编写过程中，我们首先进行了大量的文献查询与阅读，深入了解了DNS数据报的格式和收发规则，并以此确立了程序的流程图。

随后，我们进行了模块划分，分工合作编写代码，但是我们在编写代码时也没有停止讨论，这个函数怎么写高效，这个模块的接口是什么，我们都是经历重重讨论确定下来的，每个人都对程序有深入的理解。

在调试debug阶段，我们运用wireshark以程序的dubug模式，分析数据报的收发是否正确，格式是否合格，最终改正了若干bug，完美实现了中继服务器的功能。