# 计算机网络课程设计DNS报告

成员一：王皓睿 2018211311班2018211418

成员二：朱洁 2018211312班 2018211445

目录

[计算机网络课程设计DNS报告 1](#_Toc50673857)

[一、 系统功能设计 1](#_Toc50673858)

[（1）、设计要求 1](#_Toc50673859)

[（2）、设计要求分析 2](#_Toc50673860)

[（3）、功能及验收项目 2](#_Toc50673861)

[二、 模块划分 2](#_Toc50673862)

[三、 软件总体流程图 5](#_Toc50673863)

[四、 差错处理 5](#_Toc50673864)

[五、 数据包的结构 6](#_Toc50673865)

[六、 数据文件的格式 8](#_Toc50673866)

[七、 输出显示 8](#_Toc50673867)

[八、 输出显示：测试用例以及运行结果 10](#_Toc50673868)

[九、 调试中遇到并解决的问题 14](#_Toc50673869)

[十、 小组成员分工及承担任务比例 14](#_Toc50673870)

[十一、 程序的优点、不足及改进思路 14](#_Toc50673871)

[十二、 心得体会 15](#_Toc50673872)

[十三、 源代码 15](#_Toc50673873)

## 系统功能设计

### （1）、设计要求

①检索结果为普通IP地址，则向客户返回这个地址（即DNS服务器功能）

②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（即不良网站拦截功能）

③表中未检到该域名，则向实际的本地DNS服务器发出查询，并将结果返给客户端（即DNS中继功能）

### （2）、设计要求分析

①接受DNS请求与发送DNS请求和回应需要使用基于SOCKET的UDP通信。

②DNS服务器本地查询功能需要使用本地文件存储资源。

③对于请求的ID要进行重新分配以防重复。

④对域名查询不同的结果要进行不同的操作。

⑤考虑本机和网络不同的字节序，通信中对DNS报文进行适当地转换。

⑥收到上层DNS回应，发送回客户之外要将查询结果作为资源缓存下来以备再次请求。

⑦由于UDP的不可靠性，有可能造成等待上层DNS回应阻塞，设置超时重发。

⑧将请求存储在请求池数据结构中，开设多个线程进行请求处理，防止长时间阻塞。

⑨为避免长时间未处理的请求与缓存，设置有效时间并定时进行请求与缓存清理。

### （3）、功能及验收项目

①DNS服务器功能：查询本地储存的合法IP返回给客户端。

②不良网站拦截功能：查询结果IP为0.0.0.0，返回报错信息。

③DNS中继功能

④数据存储功能：

⑤数据文件更新功能：数据文件将在内存中进行更新。

⑥输出显示功能：调试信息级别分为三类

（0：无调试信息输出；1：简单调试信息输出；2：冗长调试信息输出）

⑦多客户端功能：利用多线程实现多客户端可同时查询服务器。

⑧差错处理：TTL缓存功能，超时重发

⑨查询功能：A类型、CNAME类型DNS报文查询。

## 模块划分

**（1）主模块（主线程）：**

主要负责实现以下功能：

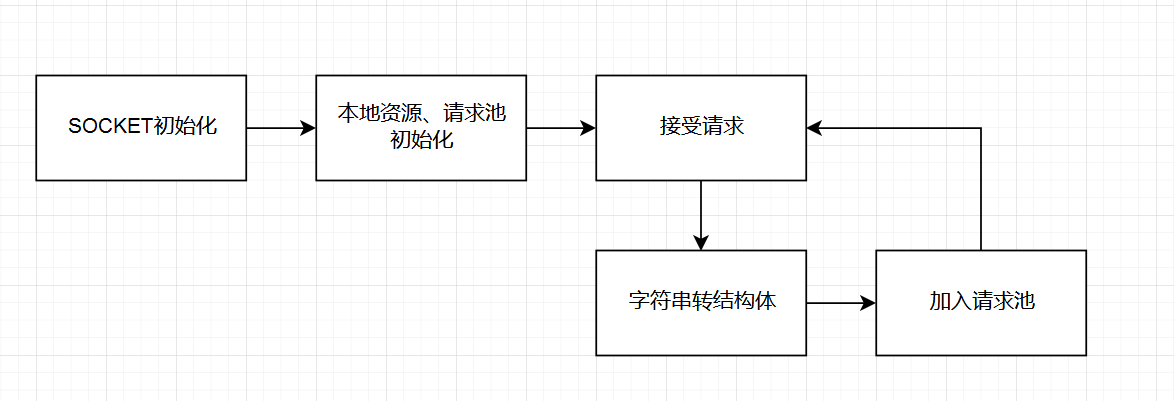
①SOCKET通信初始化；

②本地资源、请求池数据结构初始化；

③接受请求；

④将DNS报文字符串解析为便于操作的DNS包结构体；

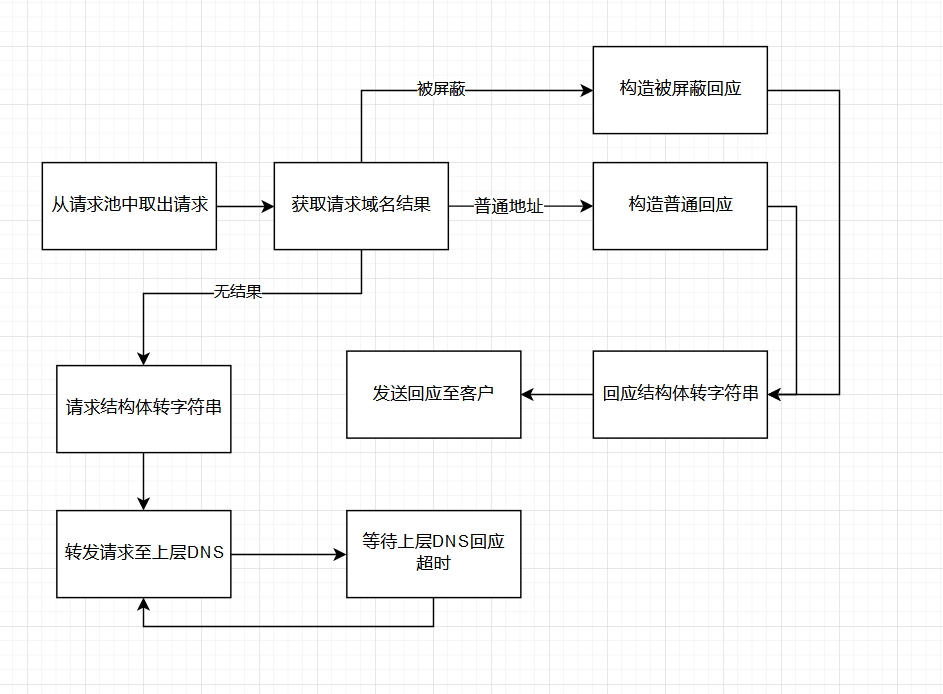
⑤将请求放入请求池供请求处理模块处理。



**②请求处理模块（请求处理线程）：**

主要负责实现以下功能：

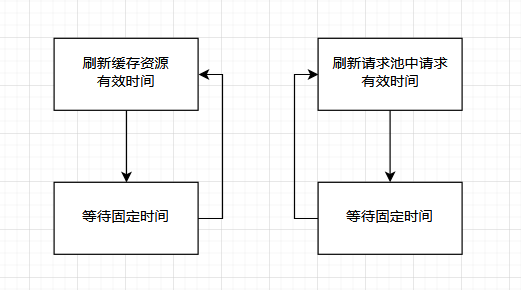
①从请求池中取出请求进行处理（包括在本地资源匹配到域名后发送回应、未匹配到域名发送请求至上层DNS、收到上层DNS回应转发给客户、增加缓存资源）。



**③请求与资源缓存管理模块（更新线程）：**

主要负责实现以下功能：

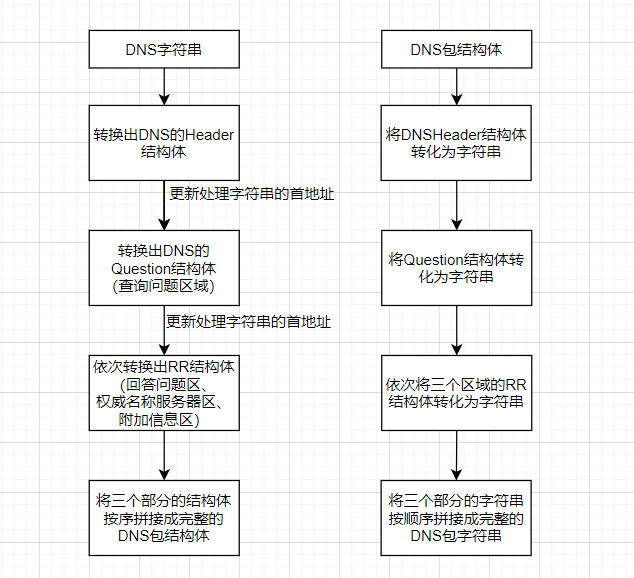
①定时更新请求池中请求与资源缓存的剩余有效时间。



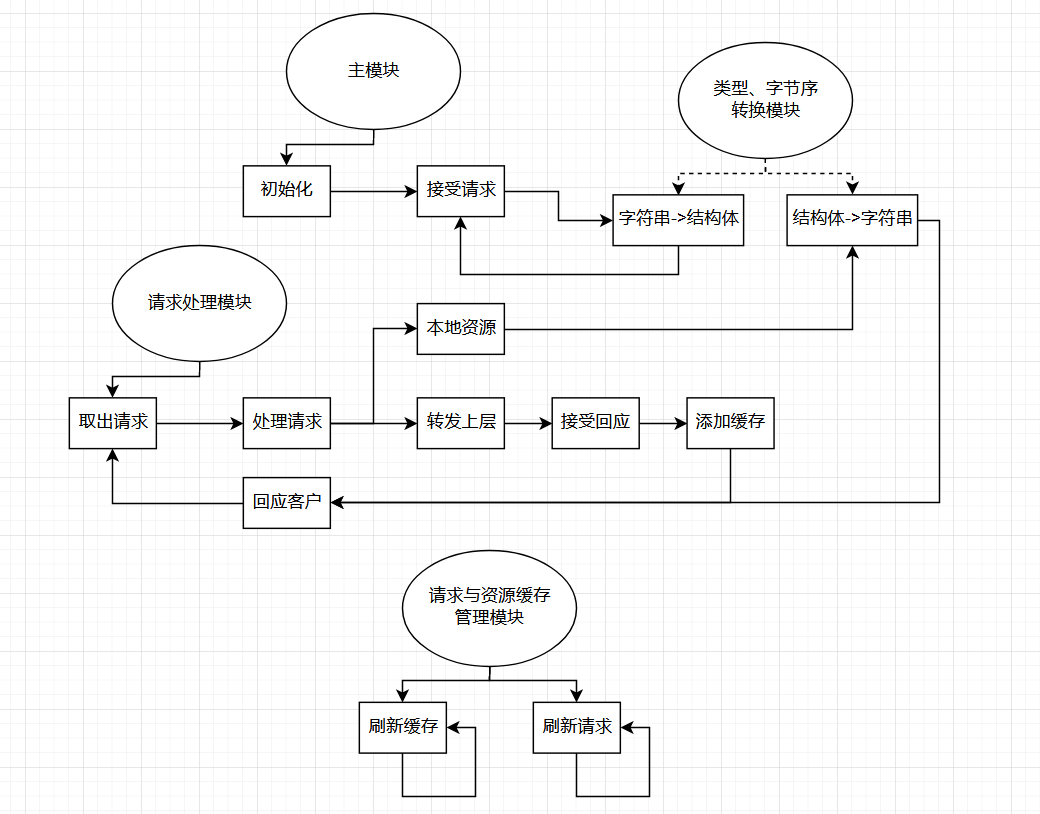
**④类型字节序转换模块：**

主要负责实现以下功能：

①将DNS报文在结构体和字符串、主机字节序和网络字节序之间进行转换。



## 软件总体流程图



## 差错处理

①超时重发：

在本地未找到资源记录进行上层DNS请求时，设置超时重发机制，在等待上层dns回应超过一定时间后进行上层DNS请求报文重新发送，避免由于UDP通信的不可靠性造成的请求长时间未收到回复，有效提高DNS系统的可靠性。

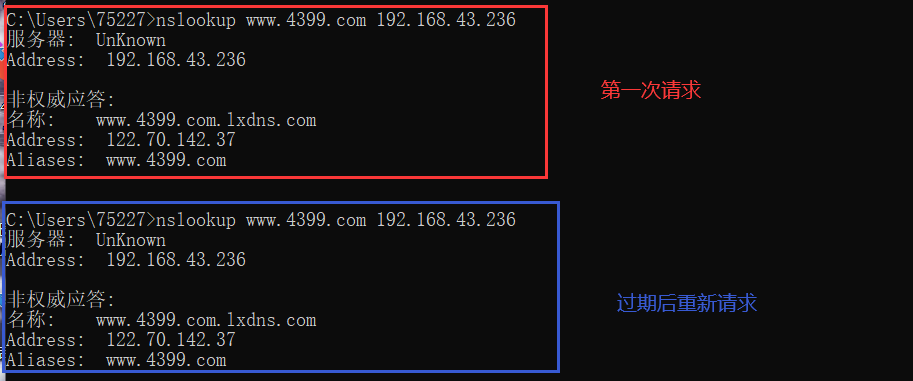
②TTL缓存：

客户端请求查询本地不存在的地址，本地会向上层DNS服务器请求，得到回应后，本地会将该地址缓存在本地，客户端再请求查询该地址时本地就可从缓存中找到，不用再向上层DNS服务器请求（如果该缓存地址没过期）。缓存的地址都是有TTL的，当某条缓存地址的TTL<=0后，该过期的缓存地址会从本地删除，客户端再请求这个地址，本地就需重新向上层DNS服务器请求。

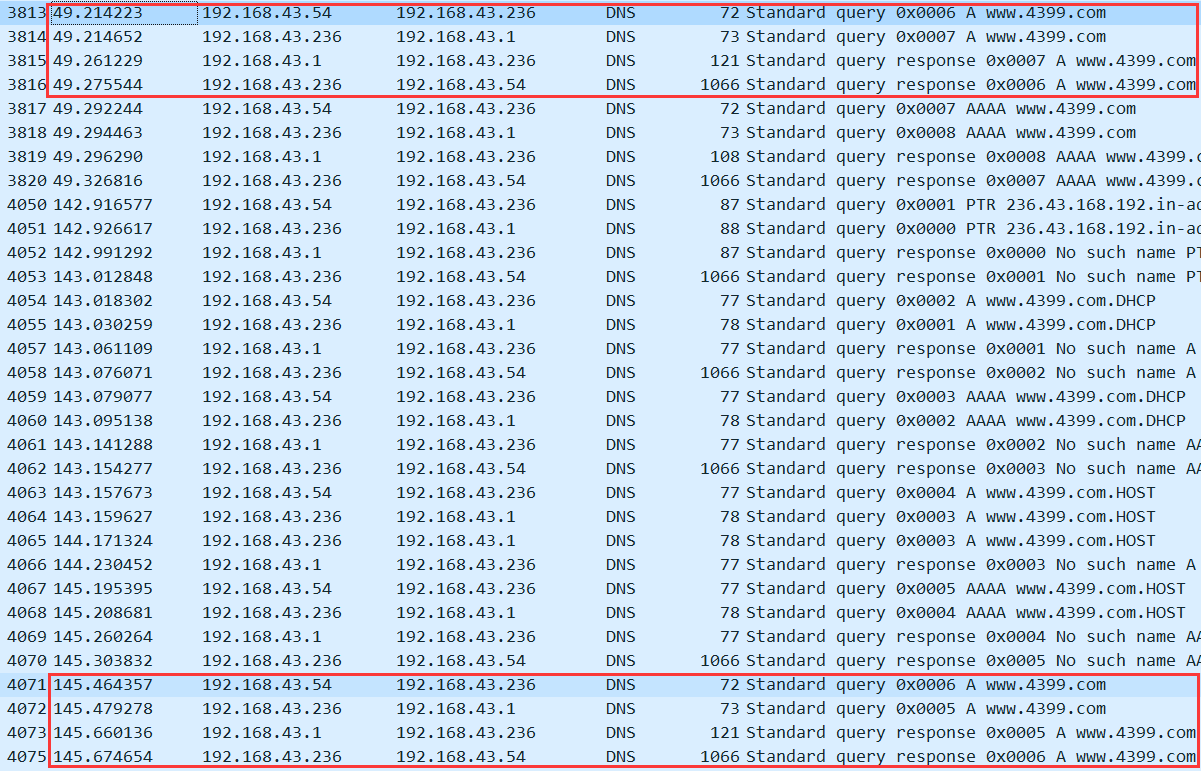
测试如下：

第一次请求，客户端请求本地不存在的地址，本地向上层DNS服务器请求后缓存到本地；

第二次请求，缓存过期后客户端重新请求。

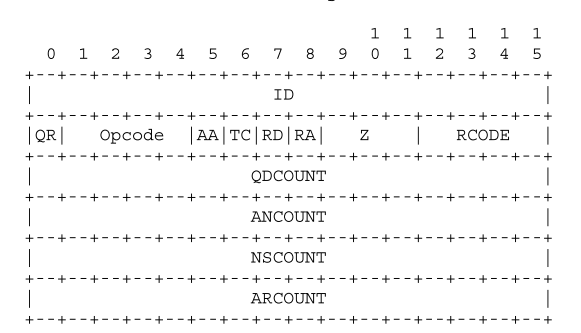


wireshark抓包结果如下图所示：



## 数据包的结构

①DNS头部区域结构体（DNSHeader）：头部的规定结构如下



设计DNSHeader包含如下数据：

UINT16 id; //16位,该值由发出DNS请求的程序生成，DNS服务器在响应时会使用该ID，这样便于请求程序区分不同的DNS响应。

//位域排序类似于小段方式

unsigned char rd : 1;//1位，期望使用递归解析, 如果该位为0，表示使用迭代查询方式

unsigned char tc : 1;//1位，被截断的报文:当响应总长度超512字节时，只返回前512个字节

unsigned char aa : 1; //1位，权威答案,表示响应的服务器是否是权威DNS服务器。

unsigned char opCode : 4; //4位，0:标准查询 ,指示请求的类型，有请求发起者设定，响应消息中复用该值。 1:反向查询 , 2:服务器状态请求

unsigned char qr : 1; //1位，0表示查询请求报文，1表示响应报文

unsigned char rCode : 4; //4位，响应码(Response coded)，仅用于响应报文

unsigned char z : 3; //3位，必须为0，保留字段

unsigned char ra : 1; //1位，递归可用(Recursion Available),该值在响应消息中被设置或被清除，以表明服务器是否支持递归查询。

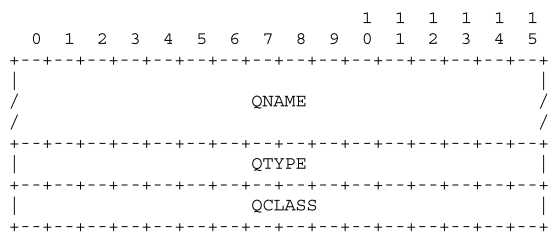
UINT16 qdCount; //question section的问题个数，16位

UINT16 anCount; //answer section的RR(Resource Record)个数，16位

UINT16 nsCount; //authority records section的RR个数，16位

UINT16 arCount; //additional records section的RR个数，16位

②查询问题区域结构体（Question）：规定结构如下：



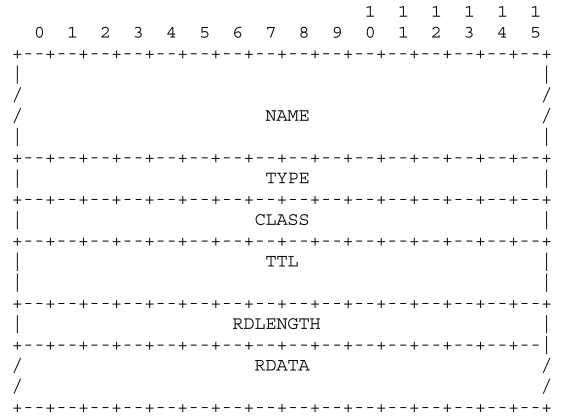
设计Question包含如下数据：

char\* qName; //字节数不定，以0x00作为结束符。表示查询的主机名。主机名被"."号分割成了多段标签。在QNAME中，每段标签前面加一个数字，表示接下来标签的长度。

UINT16 qType; //占2个字节。表示RR类型

UINT16 qClass; //占2个字节。表示RR分类

③查询问题区域、权威名称服务器区域、附加信息区域结构体（RR），此三个区域的结构格式相同，规定结构如下：



设计RR包含如下数据：

char\* name;

UINT16 type; //占2个字节。表示RR的类型，如A、CNAME、NS等

UINT16 rclass; //占2个字节。表示RR的分类

UINT32 ttl; //占4个字节。表示RR生命周期，即RR缓存时长，单位是秒

UINT16 rdLength; //占2个字节。指定RDATA字段的字节数

char\* rData;

## 数据文件的格式

本地文件资源格式：

（1）文件中存储形式：点分制IP（字符串类型） 域名（字符串类型）

（2）内存中存储形式：

struct Host

{

int type;//资源类型 ADDR\_NORMAL / ADDR\_ERROR

uint32\_t ipAddress;//IP地址

char\* domainName;//域名

struct Host\* nextHostPtr;

}HOST;

## 输出显示

缓存资源格式：

struct Cached

{

int ttl; //剩余有效时间

uint32\_t ipAddress;//IP地址

char\* domainName;//域名

char\* cName;//别名

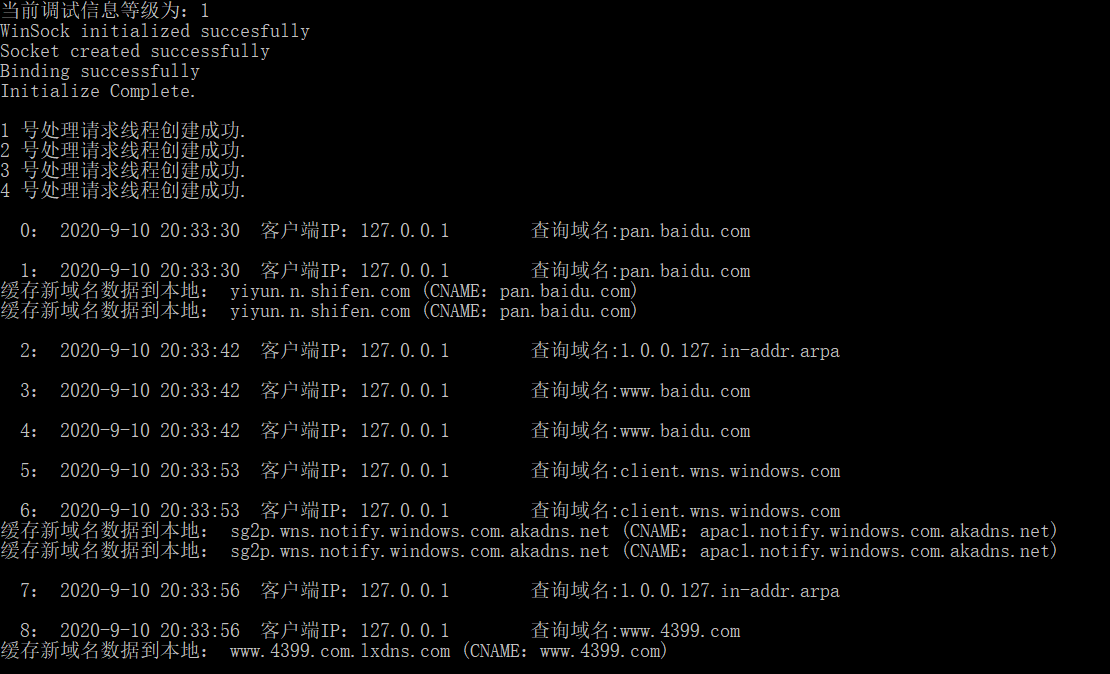
struct Cached\* nextCachedPtr;

}CACHED;

系统共设置了三种调试信息输出级别：

（1）调试信息级别为0：无调试信息输出；

（2）调试信息级别1（仅输出时间坐标，序号，客户端IP地址，查询的域名，新缓存的域名及CNAME信息），输出如下图：



（3）调试信息级别为2：

包含调试信息级别1所包含的所有内容，除此之外还包含一下信息：

①客户端请求DNS数据包的TYPE和CLASS；

②接收到DNS数据包时，输出以下信息：

从哪接收到的数据包：客户端/上层DNS服务器，及他们的客户端的IP和端口号（若是上层DNS服务器，还需输出ID转换信息）、

DNS数据包的大小及具体的十六进制内容、

DNS数据包Header部分的分析内容（ID、FLAG、Question数、Answer RR数、Authority RR数、Additional RR数）;

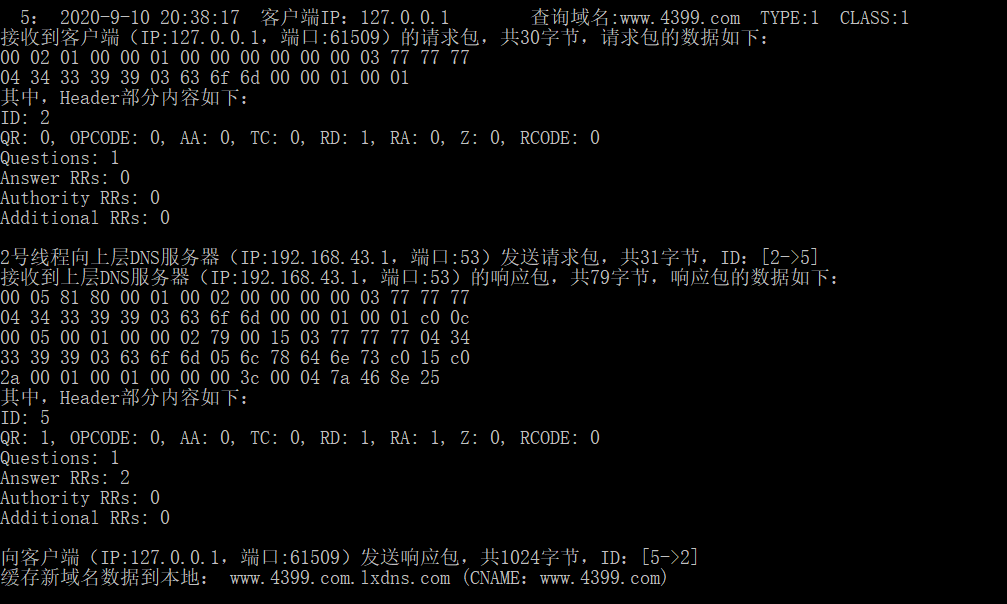
③发送DNS数据包时，输出以下信息：

向哪发送数据包（客户端/上层DNS服务器，及他们的客户端的IP和端口号）、

DNS数据包的大小、

ID转换信息。

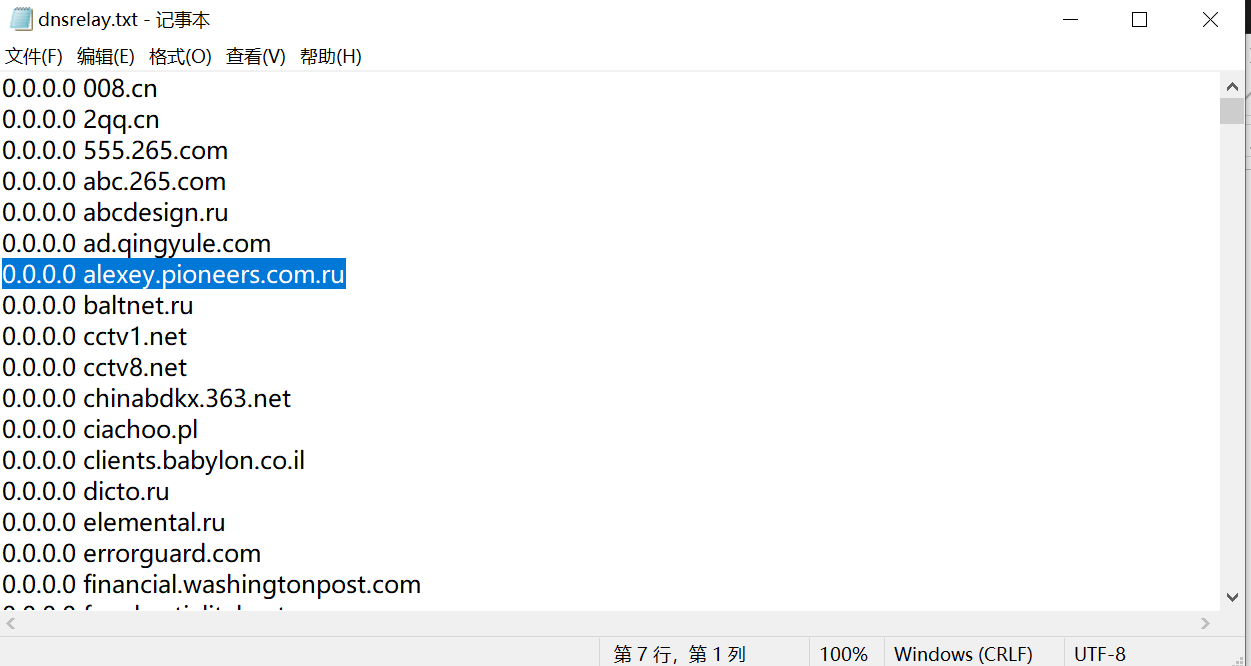
输出样例如下图：

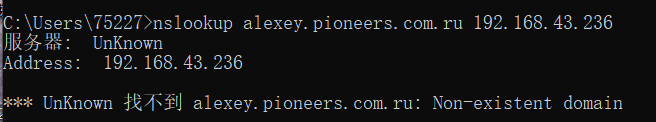


## 输出显示：测试用例以及运行结果

（1）测试本地文件中储存的无效地址（拦截功能）：

如域名alexey.pioneers.com.ru的IP是0.0.0.0，为无效地址，命令行窗口测试结果如下图：

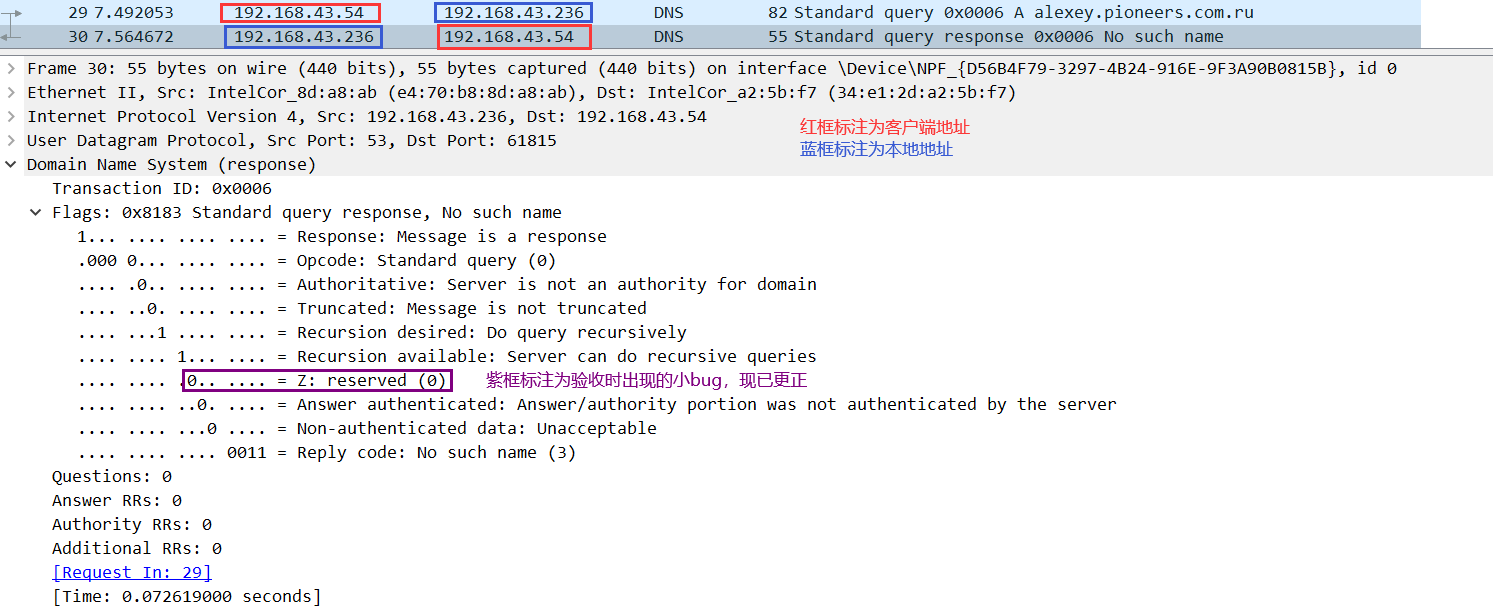




wireshark测试结果如下：

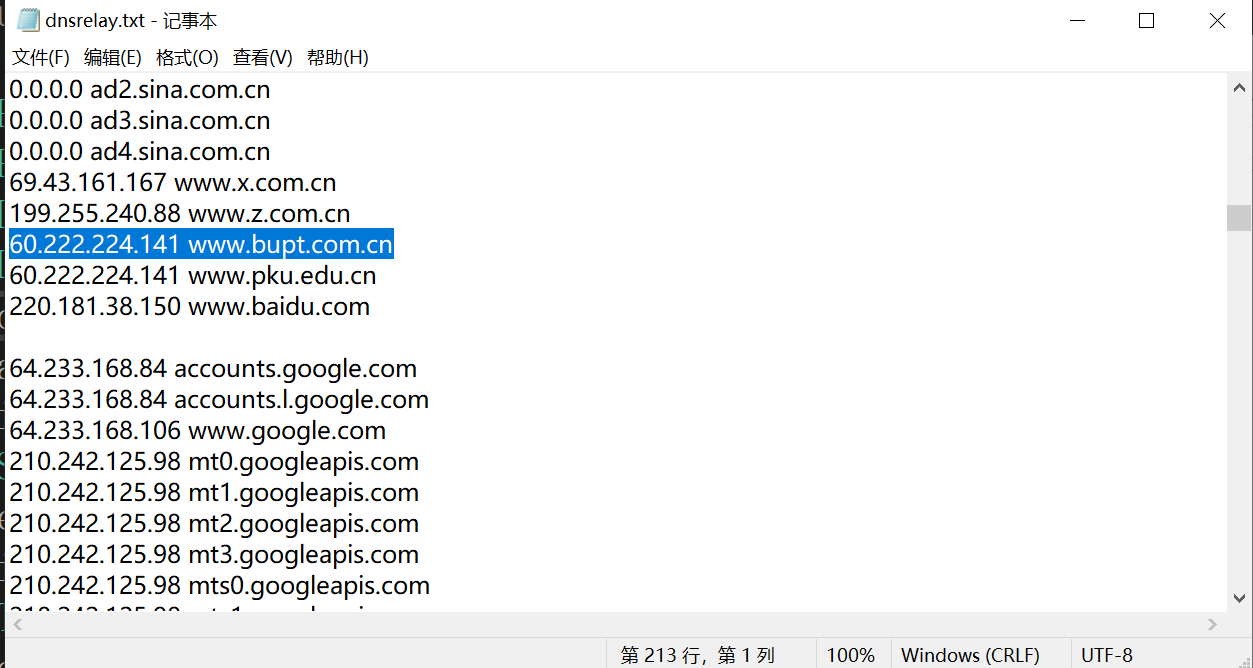
客户端（192.168.43.54）向本机（192.168.43.256）查询，本机在本地资源找到地址，并返回给客户端信息。收发2个包。

**（验收时z标志端出现了小bug，现已更正）**



（2）测试本地文件中储存的有效地址（DNS服务器功能）：

如域名www.bupt.com.cn的IP是60.222.224.141，为有效地址，命令行窗口测试结果如下图：

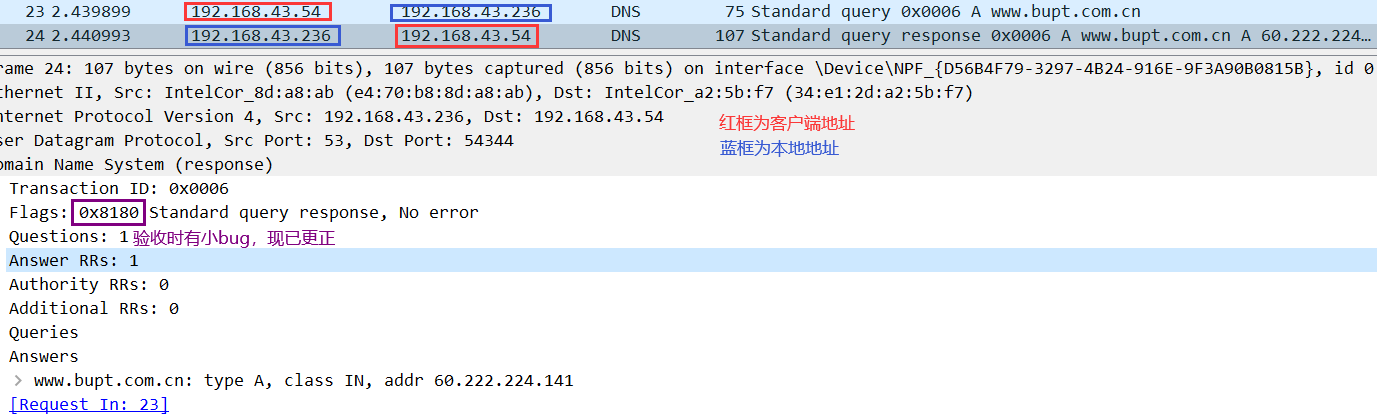




wireshark测试结果如下：

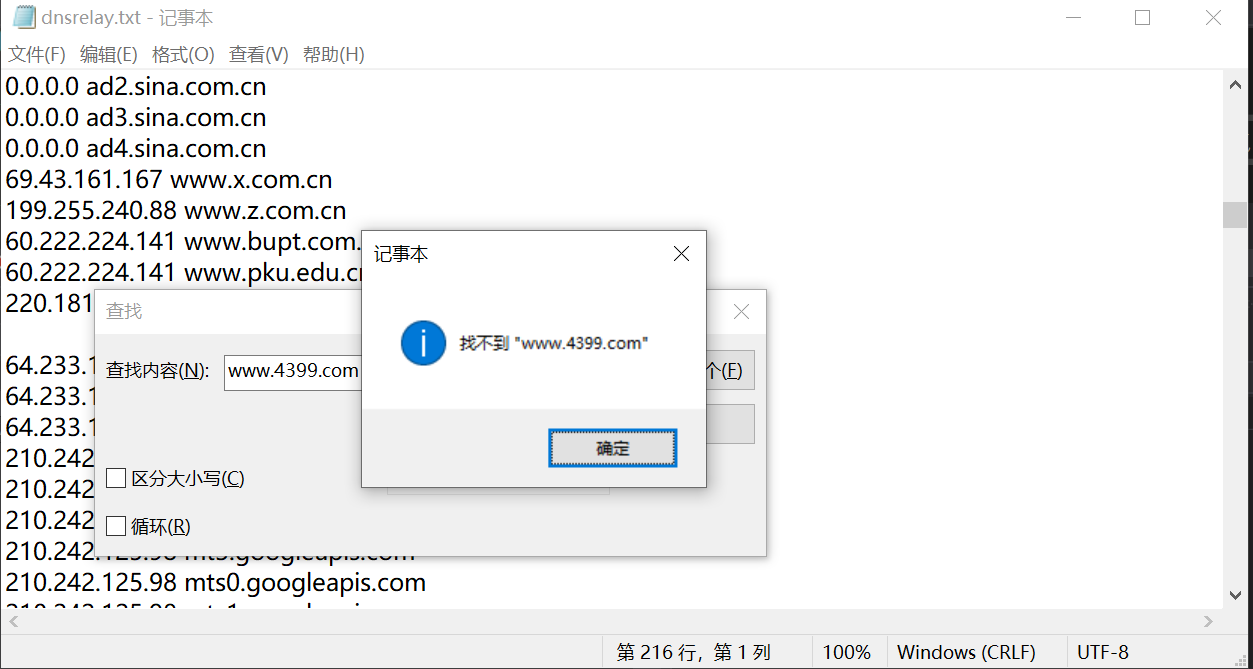
客户端（192.168.43.54）向本机（192.168.43.256）查询，本机在本地资源找到地址，并返回给客户端信息。收发2个包。

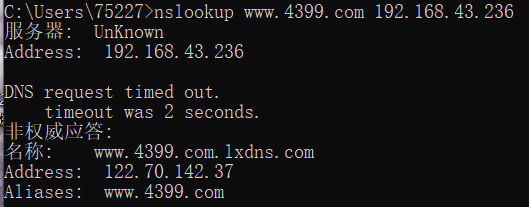
**（验收时z标志端出现了小bug，现已更正）**



（3）测试本地文件中没有保存的地址（中继功能）：

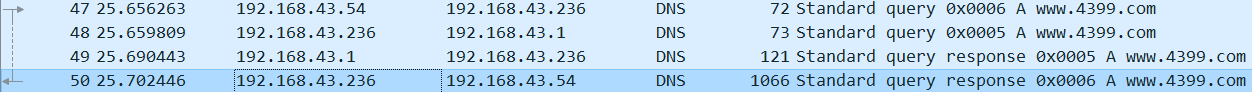
如域名[www.4399.com的IP](http://www.4399.com的IP)在本地资源中没有记录，测试结果如下图：





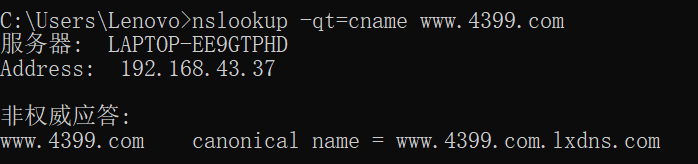
wireshark测试结果如下：

客户端（192.168.43.54）向本机（192.168.43.256）查询，本机在本地资源找不到地址，向上层DNS服务器（）发送请求，上层DNS服务器回复给本机，本机将信息转发给客户端。收发4个包。



（4）测试本地文件中不存在的CNAME：

如域名[www.4399.com](http://www.4399.com)的在本地资源中没有记录，测试结果如下图：

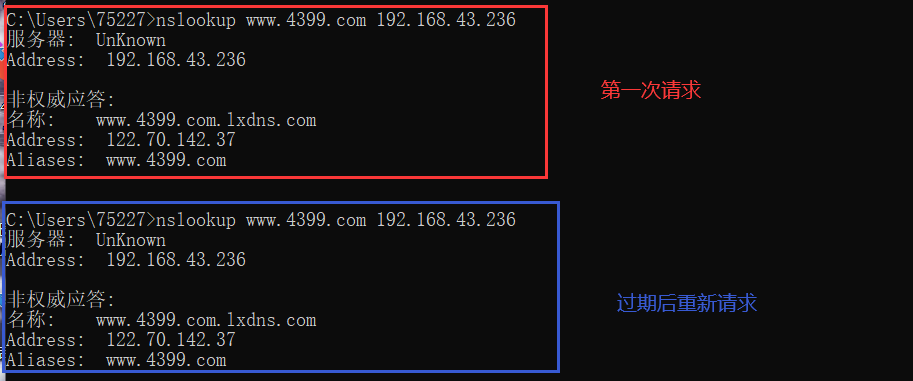


（5）测试缓存本地不存在资源后再次访问，接着在等待一段时间（缓存有效期已过）再次请求：

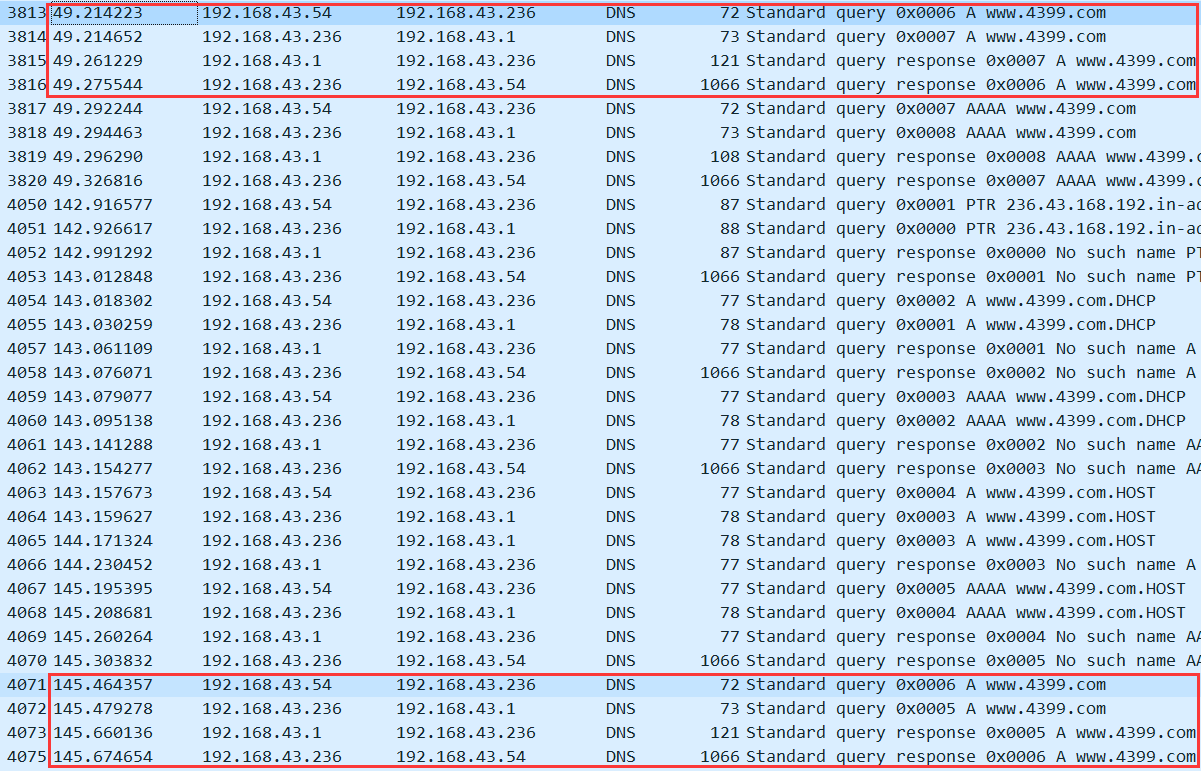
客户端命令行结果如下：

第一次请求，客户端请求本地不存在的地址，本地向上层DNS服务器请求后缓存到本地；

第二次请求，缓存过期后客户端重新请求。



wireshark抓包结果如下图所示：



## 调试中遇到并解决的问题

在调试特定输入dns包操作时，可能会调试到非预期的DNS报文，这是由于在不断网运行程序时会有其他来源的dns包被接受，因此要注意断点设置和断点条件设置，从而对预期输入的DNS报文进行流程分析调试。

在用nslookup进行本地资源测试时出现ip地址翻转的情况，这是因为在读入本地资源文件时忽略IN\_ADDR类型的s\_addr成员是网络字节序，造成构造DNS报文时ip地址是字节序是相反的，要在读入资源时用ntohl转换s\_addr成员的字节序。

在最初实际运行时偶尔会出现内存访问错误的中断出现，而初看附近代码也都有判空条件设定，最终发现这是由于请求池或缓存列表这两个全局变量同时被不同线程进行操作导致的，不同于本地资源列表的只读操作，请求池和缓存列表都会被经常读写，这就造成了操作冲突。最初由于实验对编程语言的限制，无法使用C++的mutex互斥量解决多线程冲突的问题，因此就定义全局变量进行简易的互斥量代替，但仍然会出现问题，最终在添加延时等操作后实现了类mutex上锁解锁操作的互斥量解决了冲突问题。

## 小组成员分工及承担任务比例

王皓睿负责

编写接受DNS报文、处理DNS报文请求、发送回应报文、系统请求池及缓存的管理、部分初始化。

代码占比约50%。

朱洁负责

编写DNS报文的各个部分在结构体与字符串之间的转换、输出显示、部分初始化。

代码占比约50%。

## 程序的优点、不足及改进思路

优点：

1. 对A类地址请求的全面应答
2. 缓存功能提高程序效率
3. 请求、缓存有效期机制，资源利用率最大化
4. 接受请求----四线程处理请求池请求机制，实现多客户端
5. 超时重发机制避免由于UDP通信导致的不可靠性

不足及改进思路：

（1）程序稳定性有待提高，运行一段时间后会关闭。经过调试发现，程序关闭的原因有以下几点：

①程序不断申请堆区内存，使用完后没有及时释放内存，且指针的指向也改为别的地址，之前使用过的内存就无法被删除。运行时间一长，堆区内容不够分配，系统就将结束。

②删除了一些已经释放过的空间。由于使用了多线程，就需要对多个线程共有的数据进行保护，同一时间只能由一个线程对共有数据进行读写操作。在c++中有mutex类可以对共有数据上锁和解锁，但由于c中没有mutex类，只能自己写一个简易版的锁，功能没有那么强大，最终可能导致几个线程同时占有某项共有数据，出现一个线程要释放某项数据空间可该数据空间被别的线程释放了的情况，系统出错。

改进思路：

①某个在堆区分配的数据在使用之后，及时释放。释放结构体指针的时候要注意包含关系和释放的顺序，因为一个结构体里可能还包含着其他其他类型的指针，不注意顺序释放会导致某些空间没有被释放及一些差错。

②利用全局变量设计一个类似C++Mutex的互斥量解决不同线程对同一对象进行修改读写造成的冲突

（2）本地服务器构造的DNS包在申请完空间后，使用之前没有先全初始化为0，导致Header部分的z字段的值不为0（申请完空间后系统自动分配成了1）。

改进思路： 在申请完DNS包的空间之后就对空间的数据全初始化为0。使用memset函数进行0值初始化操作。不只是DNS包的空间，其他用到在堆区动态分配空间的数据在使用之前都进行一边初始化操作以防万一。

（3）cache过期显示问题

Cache在运行时会出现超过有效期后仍然在缓存中存储的问题

改进思路:在进行cache刷新时，不仅要将链表结点中cache的内容释放掉，同时也要将该列表结点释放，实现对缓存的完全清除。

（4）cname问题

在查询cname时出现先前查询cname并保存在cache中后在有效期内再次访问却仍然要向上层DNS请求的问题

改进思路：将上层返回的Cname回应报文进行严格存储，使之在缓存中可以正确被识别并取出使用，对于资源的类型和内容要加以修改使之可以被正常调用，被用于构建回应报文发送至客户端。

## 心得体会

实验初，通过对SOCKET通信与RFC1035文件的学习，形成了对于DNS系统在结构、功能上的理解，对于实验的功能实现有了初步的规划；而后在对DNS系统的各项功能进行具体实现时，对网络通信的格式、字节序等等细节有了全面的了解，对于网络传输的内容有了直观的认识；在针对问题进行调试时，更加熟练地运用wireshark进行端口监听、报文捕获，近距离解除原始的网络报文，针对报文数据进行全面的分析，更深层次地理解了计算机网络通信的各种要素。

## 源代码

源代码过长，以附件形式提交。