

目录

[一、 系统的功能设计 4](#_Toc73627165)

[功能实现 4](#_Toc73627166)

[DNS报文分析 4](#_Toc73627167)

[不良网站拦截功能 4](#_Toc73627168)

[服务器功能 4](#_Toc73627169)

[中继功能 4](#_Toc73627170)

[高并发多线程处理功能 4](#_Toc73627171)

[DNS ID转换功能 5](#_Toc73627172)

[超时处理功能 5](#_Toc73627173)

[命令行用户界面 5](#_Toc73627174)

[开发环境 5](#_Toc73627175)

[二、 模块划分 6](#_Toc73627176)

[main 6](#_Toc73627177)

[config 6](#_Toc73627178)

[dns\_server 6](#_Toc73627179)

[dns\_client 7](#_Toc73627180)

[dns\_parse 7](#_Toc73627181)

[thread\_pool 7](#_Toc73627182)

[db 7](#_Toc73627183)

[dns\_id\_map 7](#_Toc73627184)

[timer 8](#_Toc73627185)

[rbtree 8](#_Toc73627186)

[cache 8](#_Toc73627187)

[三、 软件流程图 8](#_Toc73627188)

[程序整体处理流程 9](#_Toc73627189)

[DNS报文的接收进行响应的处理流程图 10](#_Toc73627190)

[线程池进行多线程调度的处理流程图 11](#_Toc73627191)

[计时器模块处理流程图 12](#_Toc73627192)

[ID映射模块处理流程图 12](#_Toc73627193)

[红黑树模块的处理流程图 13](#_Toc73627194)

[红黑树模块插入操作的处理流程图 14](#_Toc73627195)

[红黑树模块删除操作的处理流程图 15](#_Toc73627196)

[cache模块的整体运行流程图 17](#_Toc73627197)

[cache模块插入处理流程图 17](#_Toc73627198)

[cache模块查找处理流程图 18](#_Toc73627199)

[四、 测试用例以及运行结果 18](#_Toc73627200)

[命令行界面 19](#_Toc73627201)

[不良网站拦截 21](#_Toc73627202)

[服务器功能 22](#_Toc73627203)

[中继功能 24](#_Toc73627204)

[DNS ID转换功能 25](#_Toc73627205)

[超时处理功能 26](#_Toc73627206)

[五、 调试中遇到并解决的问题 26](#_Toc73627207)

[六、 心得体会 27](#_Toc73627208)

1. 系统的功能设计

## 功能实现

### DNS报文分析

DNS即域名系统，是用于进行“域名-IP地址”转换的一个分布式数据库。其报文结构由报文头部、问题字段、回答字段、权威字段、附加字段这5部分组成。其中报文头部固定12字节，由DNS ID、查询/返回标志QR，返回码RCODE等内容组成。其余字段均遵循一定规则进行组织，包含名字、类型、类等字段。

对于域名等名字形式的资源，DNS报文采取两种表示方法：一，“标签长度-标签内容”，以0x00作为结尾标志；二，依据偏移量指向其他字段的压缩表示。

而本程序实现了将DNS报文由进行解析的功能，并根据其上述设计规则，组织了对应的数据结构进行存储，以便进行进一步的处理，并实现了字节流报文数据和解析后的DNS数据包相互转换的功能。

### 不良网站拦截功能

程序从本地“域名-IP地址”映射表中读取表项，获取需要屏蔽的域名内容（即对应IP地址为0.0.0.0的内容）。在服务器接收到查询报文后，首先在本地数据中进行查询。如果本地查询返回的IP地址是全零，则说明该域名是需要被屏蔽的内容，此时程序将返回特殊的数据包，将DNS报文头部标志字段里的RCODE（返回码）置为3，向询问客户端表示“名字差错”。从而对实现不良网站的拦截。

### 服务器功能

接收到客户端发来的DNS查询报文后，程序首先在本地“域名-IP地址”映射表和Cache中进行查询，如果请求的域名类型为A类请求（即ipv4地址查询）且仅查询单个表项，则当域名请求命中后，本程序根据请求报文和对应IP地址，直接向客户端返回DNS报文内容。

### 中继功能

当DNS查询报文请求的域名类型为其他类型，或者有多个查询字段时，本程序将不再充当服务器，而是将经过处理的请求报文转交给远程服务器，并将从远程服务器返回的回应报文返回给对应的请求客户端。

### 高并发多线程处理功能

本程序需要考虑允许多个客户端的并发查询，并且允许在第一个查询尚未得到答案之前就启动处理另外一个客户端查询。

考虑到多进程对性能和资源的占用，以及select等I/O多路复用函数对于DNS服务器固定端口的UDP连接处理并不方便，本程序采取了多线程的方式实现了并发处理。

从可移植性的角度考虑，本程序主要基于POSIX标准库函数pthread进行多线程编程。同时，为了减小高并发条件下线程频繁创造和销毁的开销，本程序实现了一个线程池，对于并发请求进行预线程化、管理调度和回收资源的处理。

### DNS ID转换功能

由于本程序始终工作在本机的53号端口，而且考虑到DNS查询请求可能来自多个客户端或主机，因此有可能出现来自不同客户端的ID发送碰撞的可能，在程序实现中继功能时，如果不加以处理直接按照原报文请求发送，会造成无法区分远程DNS服务器发过来的响应报文具体对应哪个客户端的问题。

因此，在执行中继功能时，程序有必要进行ID转换。为了提高查找效率，本程序将目前使用的ID作为键，对应转换前的ID及其客户端信息作为值存储在红黑树中，用于表示映射关系。每次进行中继发送请求之前，首先尝试将原ID插入映射表中，如果原ID未被使用，则可以直接以原ID作为新的ID使用，其红黑树的值设置为本身的ID及客户端信息；否则，将随机生成一个新的ID，在红黑树中反复查询新ID是否已经被占用，直至找到一个空闲的ID资源，并将其原ID和客户端信息保存至红黑树中。而当对方响应到达，或者对应超时，则将在ID映射表中删除对应表项，回收ID资源。

### 超时处理功能

为了更好的在多线程中进行超时处理，本程序实现了一个独立线程的定时器，定时器单独设置一个线程，进行定时功能，当每轮定时时间结束后，计时器将执行一个回调函数，而当多轮定时结束或达到最大重设次数后，计时器则会执行一个清理函数并退出。

而在本程序中，超时处理主要是针对中继发往远程服务器重传，当每轮超时结束后，超时重传计时器将会回调重传函数再次发送数据包，而达到最大重传次数后，则会执行清理函数，回收ID资源，并且回收内存资源。

### 命令行用户界面

为了给用户提供更多交互选项和调试信息的提供，本程序还基于linux的标准库函数getopt函数（在Windows系统下通过移植库函数实现），实现了一个命令行用户界面，能够根据用户的需求，修改调试等级、打印相关调试信息等功能。

## 开发环境

语言标准：C11

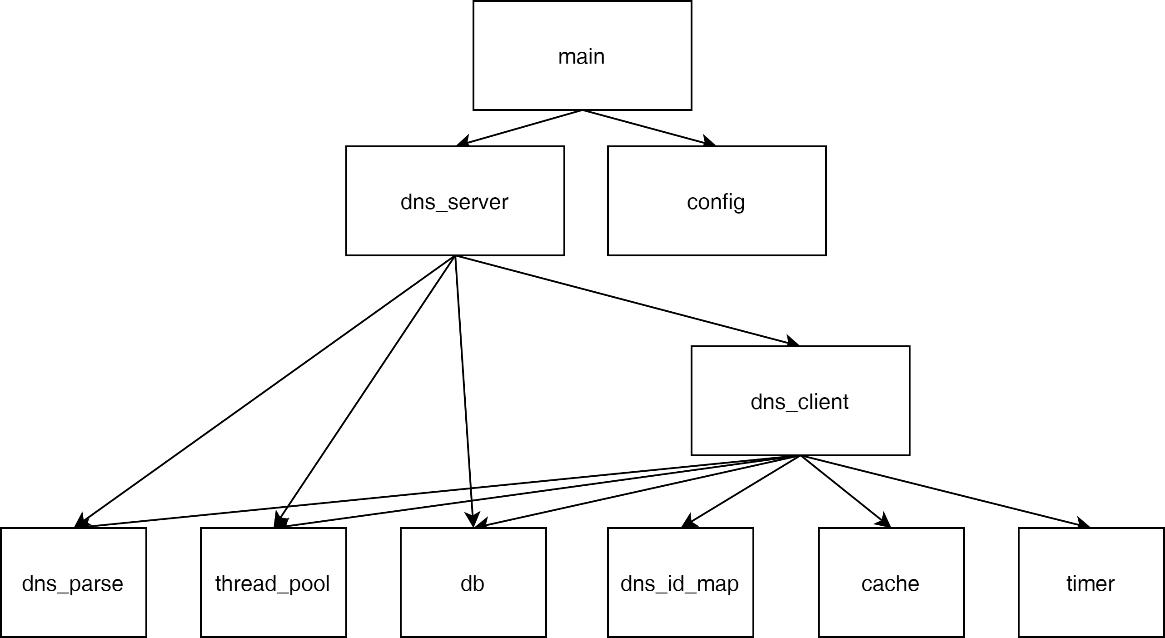
集成开发环境：Visual Studio 2017、Visual Studio 2019

操作系统：Windows10

1. 模块划分

为了更好地区分模块与模块间的功能，本程序将整个程序划分为了多个模块，分别为：初始化设置模块main、config，程序主体逻辑功能的实现模块dns\_server、dns\_client，以及为解决子问题而建立的子模块dns\_parse、dns\_id\_map、db、timer、thread\_pool、rbtree、cache。

而模块与模块间的具体调用关系如下图所示：



## main

程序的入口，主要进行对DNS服务器的启动和命令行设置的调用。

## config

负责对命令行界面进行交互的处理。初始化远程服务器地址、本地“域名-IP地址”映射表、调试信息等级。

## dns\_server

本程序的服务器端。首先初始化线程池、从程序中读入“域名-IP地址”映射表、初始化socket，然后开辟一个新线程，主线程阻塞直至接收到用户的关闭命令。

新线程进行服务器监听的循环，非阻塞式地持续监听本机的53号端口。如果接收到数据包，则首先判断DNS数据包是响应请求还是查询请求。如果是响应请求，则关闭对应的计时器，同时开辟新线程进行响应中继处理；如果是查询请求，则开辟新线程进行对查询的具体处理，而本线程总是持续监听端口，具体处理交由dns\_client模块，从而实现高并发操作。

## dns\_client

负责具体处理各种客户端事件的模块。

本模块负责对查询请求进行处理。如果请求能够在“域名-IP地址”映射表或者cache中找到，则生成对应的DNS响应，并发给对应的请求客户端；如果请求不能够在本地找到或者请求的类型是本地无法处理的类型，则程序首先将进行ID的转换，然后启动对应的超时处理计时器，发往远程服务器发送。如果响应超时，则可能是网络状况不良，本程序将进行重传。如果重传后仍旧无法收到回应，则考虑到可能是远程服务器无法实现功能，此时放弃对请求进行处理，回收ID映射表的对应资源。

本模块也负责对响应请求中继进行处理。首先停止对应的重传计时，并通过ID映射表得到原本的ID和客户端信息，删除ID映射表的对应表项以回收ID资源。再将中继的结果插入到LRU Cache中。最后将响应包返回给原本的请求客户端。

## dns\_parse

负责进行对DNS报文进行解析的模块。

本模块主要提供对DNS报文进行格式的解析，根据DNS报文的组织结构和规则，完成原始DNS报文中字节流向DNS报文头部、DNS名字表示、DNS询问字段、DNS资源记录等等字段的双向转换。

## thread\_pool

负责提供一个管理调度线程资源的线程池。

根据需要完成的任务数量，进行预线程化，同时根据当前工作繁忙状态，管理调度进行处理的线程数量，并且在线程结束后，回收对应的线程资源。

## db

负责管理本地“域名-IP地址”映射表。

通过红黑树来存储对应的“域名-IP地址”表项，从而提高资源的查询效率，提供查询、插入等访问数据的接口，同时考虑到多线程时存在共享资源线程不安全的问题，本模块通过互斥锁机制也保证了查询、插入、删除等操作的原子性。

## dns\_id\_map

负责管理ID映射表。

通过红黑树来存储DNS中继请求的ID与其原ID及客户端信息。

在发送给远程服务器之前，程序将试图将查询请求的ID插入到ID映射表中。如果ID映射表中不存在对应表项，说明该ID资源还没有被占用，则程序将会保持其原ID，并记录其相关信息；如果ID映射表中已经存在对应表项，说明ID资源已经被占用，那么ID映射表将通过伪随机方法生成新的ID并返回，以保证ID不冲突。

同样考虑在多线程状态下对共享资源的处理，本模块也通过互斥锁保证了对ID映射表的插入、查询等操作的原子性。

## timer

负责实现多线程计时器。

每个计时器单独开辟一个线程进行处理，每一轮都会进行定时检查，一旦发现定时超时，则将调用回调函数。若计时重启次数达到外部程序设定的上限，则会退出计时器，并调用对应的清理函数。

## rbtree

负责提供红黑树建树、增加节点、删除节点、释放节点、销毁树基本操作。

包含了多种红黑树基本操作辅助函数，例如：改变节点颜色，寻找叔叔节点，寻找祖父节点，左旋右旋操作，以及黑树建树、增加节点、删除节点、释放节点、销毁树基本操作。供程序调用。

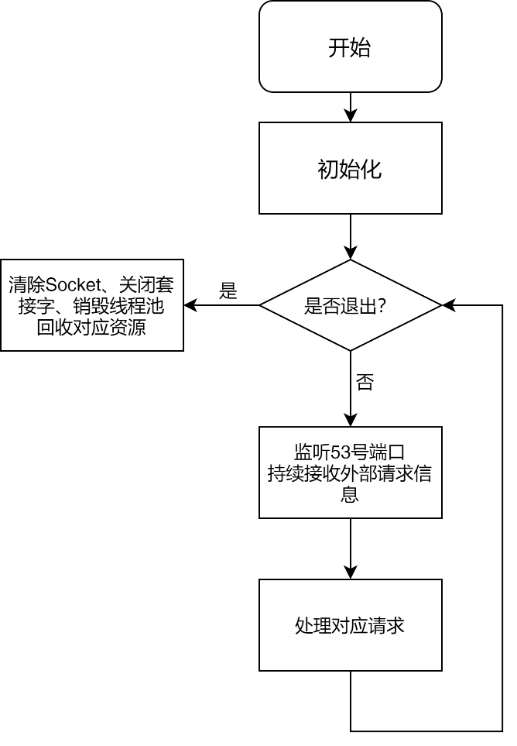
## cache

负责“域名-IP地址”映射的记忆化存储功能。

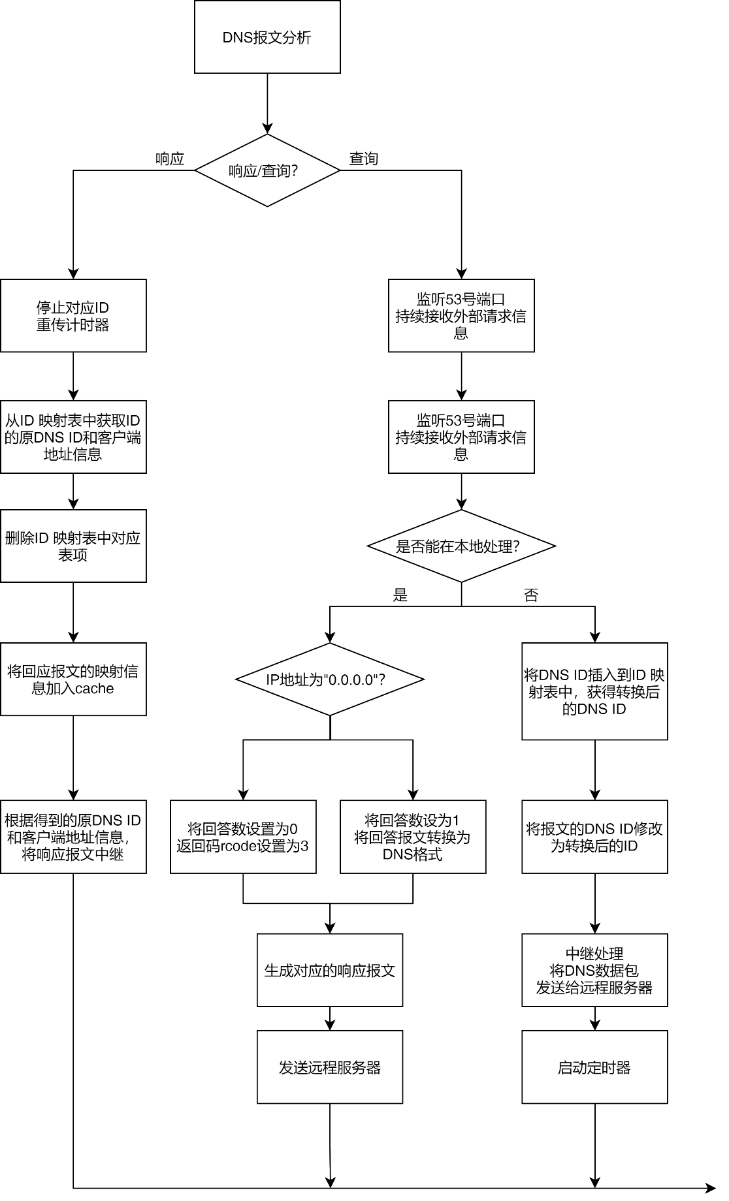
考虑到DNS的一个基本特性是使用超高速缓存，当一个名字服务器收到有关映射的信息（主机名字到IP地址）时，它会将该信息存放在高速缓存中。这样若以后遇到相同的映射请求，就能直接使用缓存中的结果而无需通过其他服务器查询。它维护一个固定大小的缓存，随当前信息与查询实时更新。

1. 软件流程图

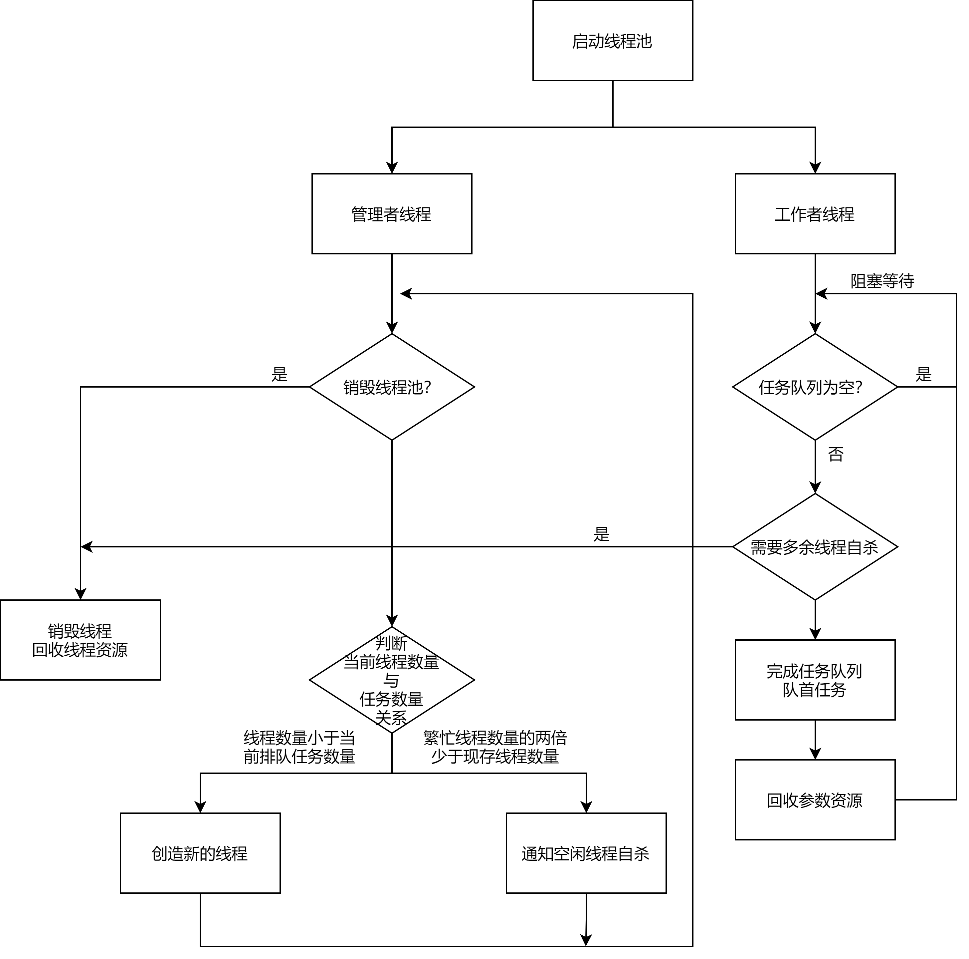
## 程序整体处理流程



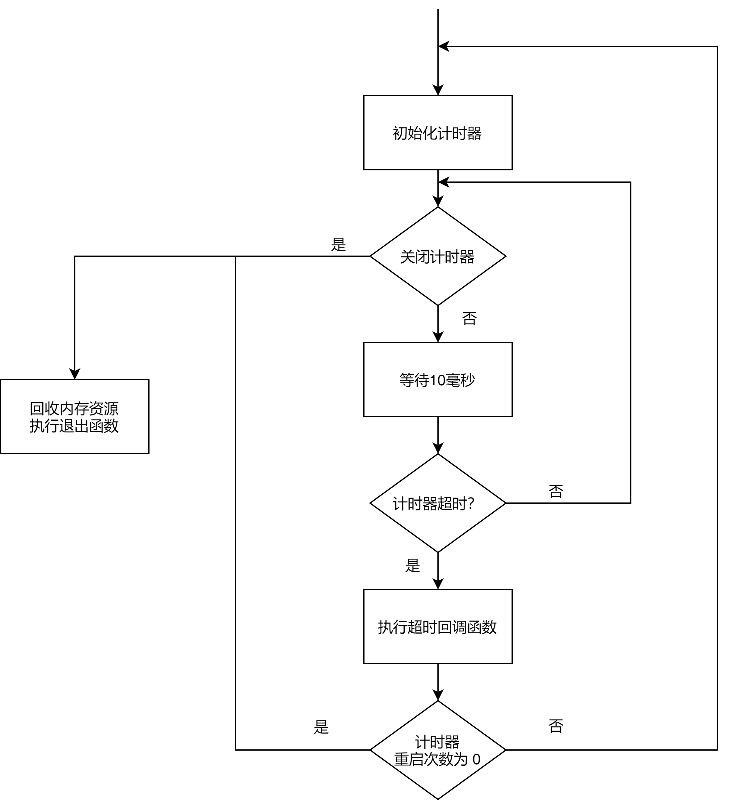
## DNS报文的接收进行响应的处理流程图



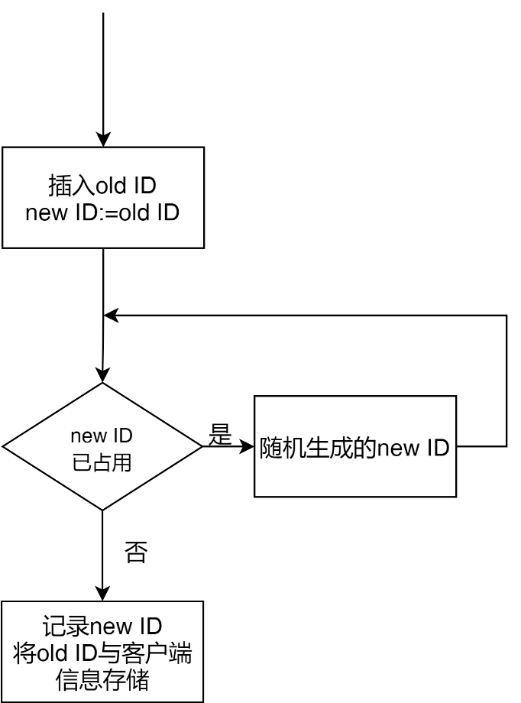
## 线程池进行多线程调度的处理流程图



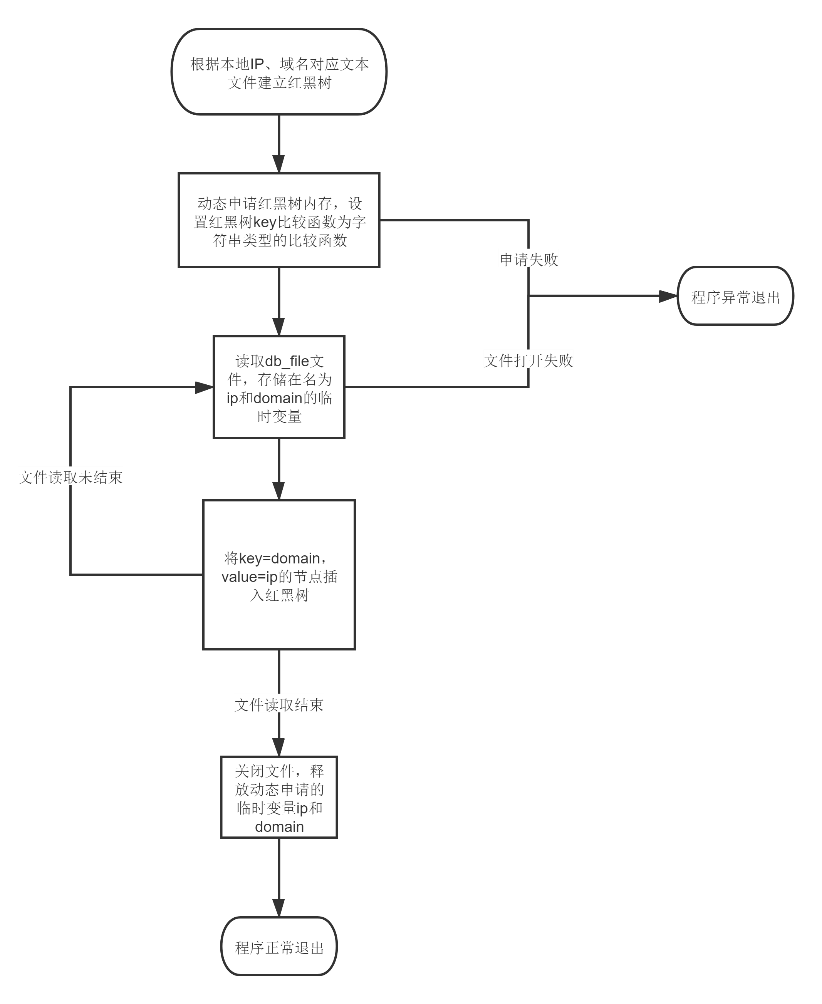
## 计时器模块处理流程图



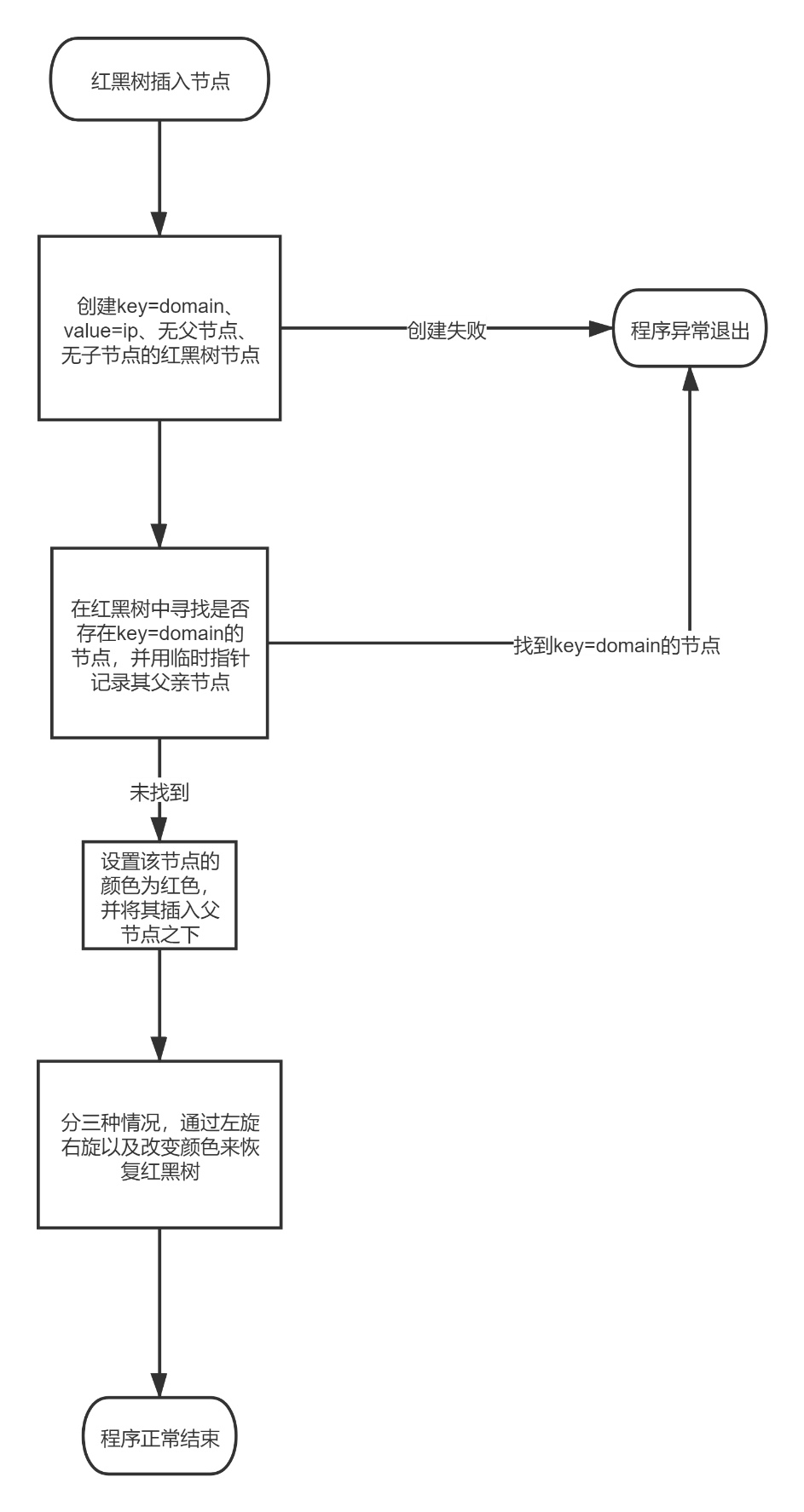
## ID映射模块处理流程图



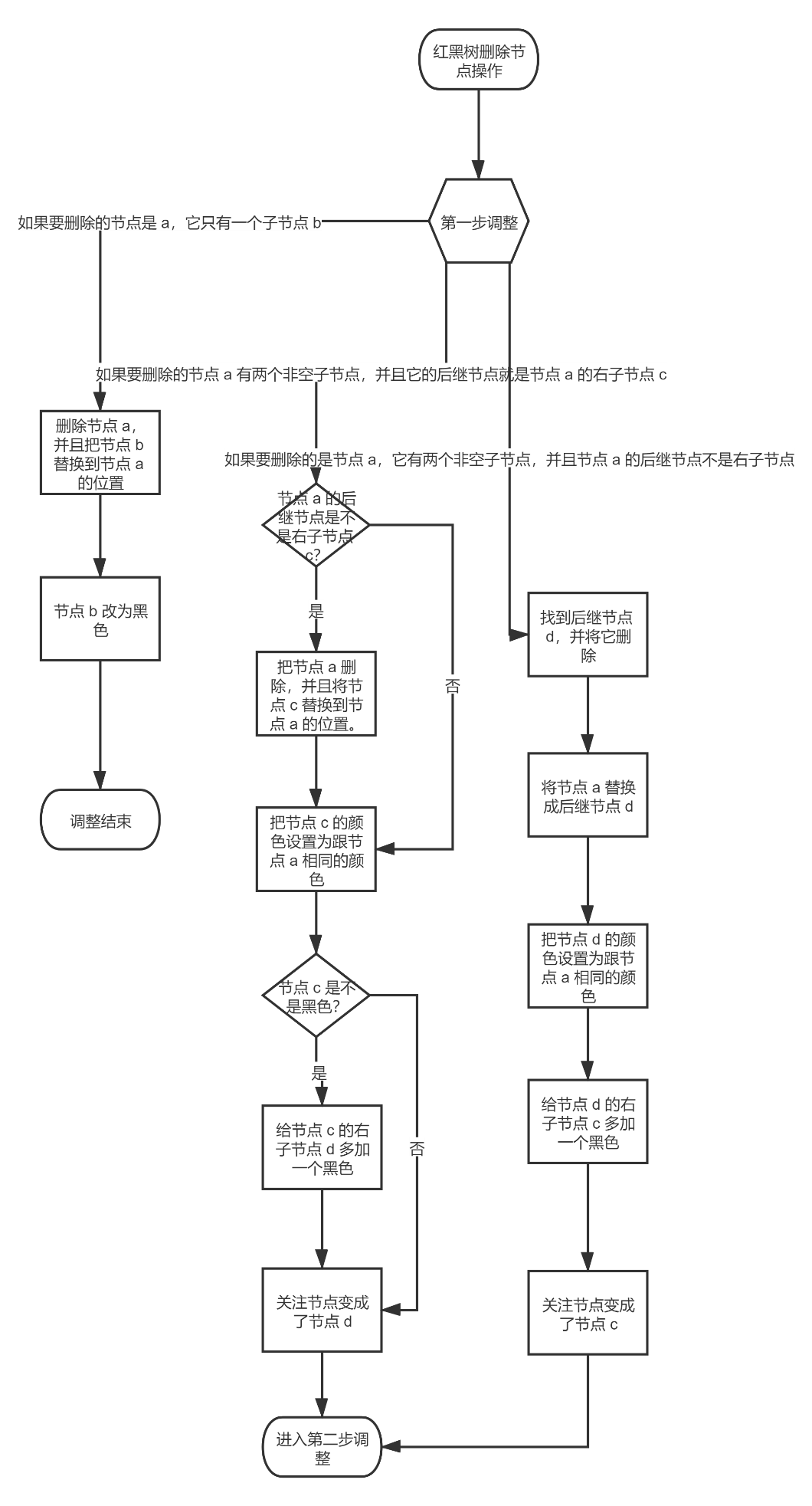
## 红黑树模块的处理流程图

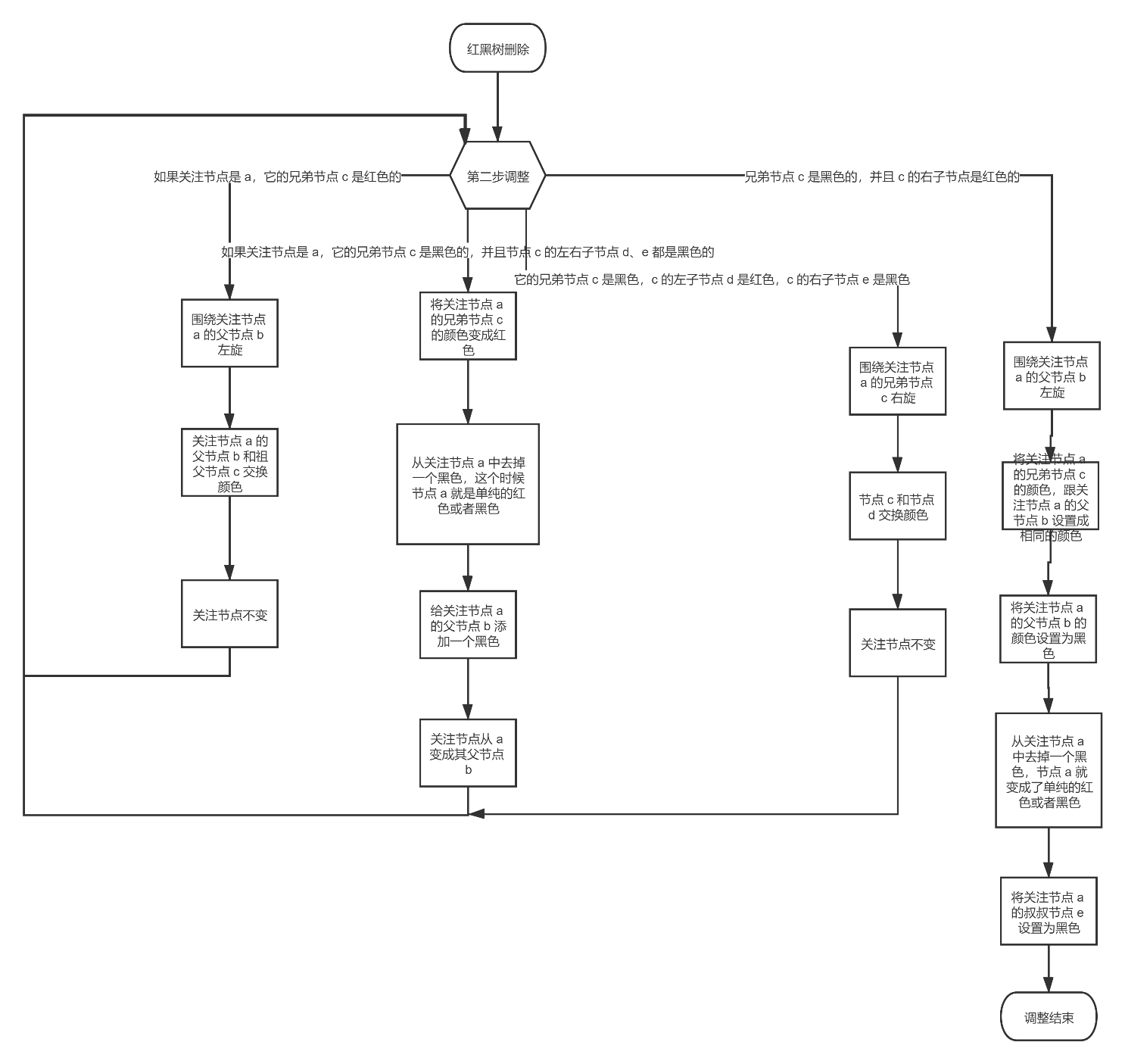


## 红黑树模块插入操作的处理流程图

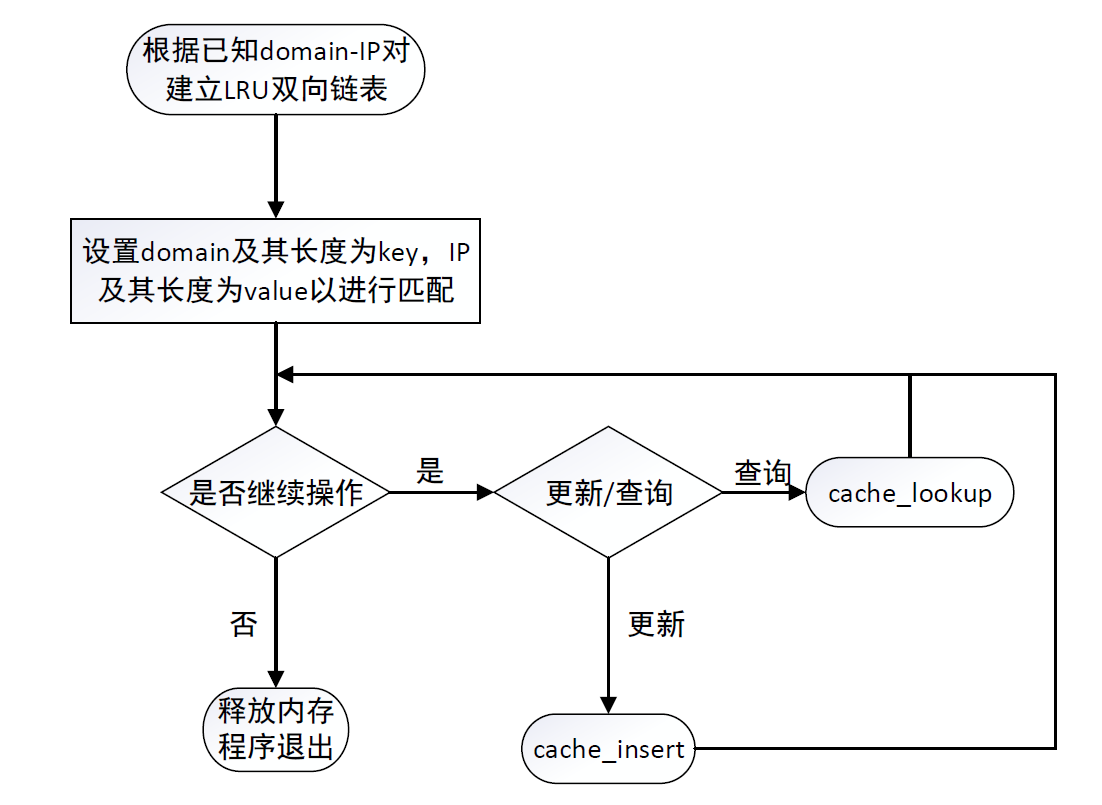


## 红黑树模块删除操作的处理流程图

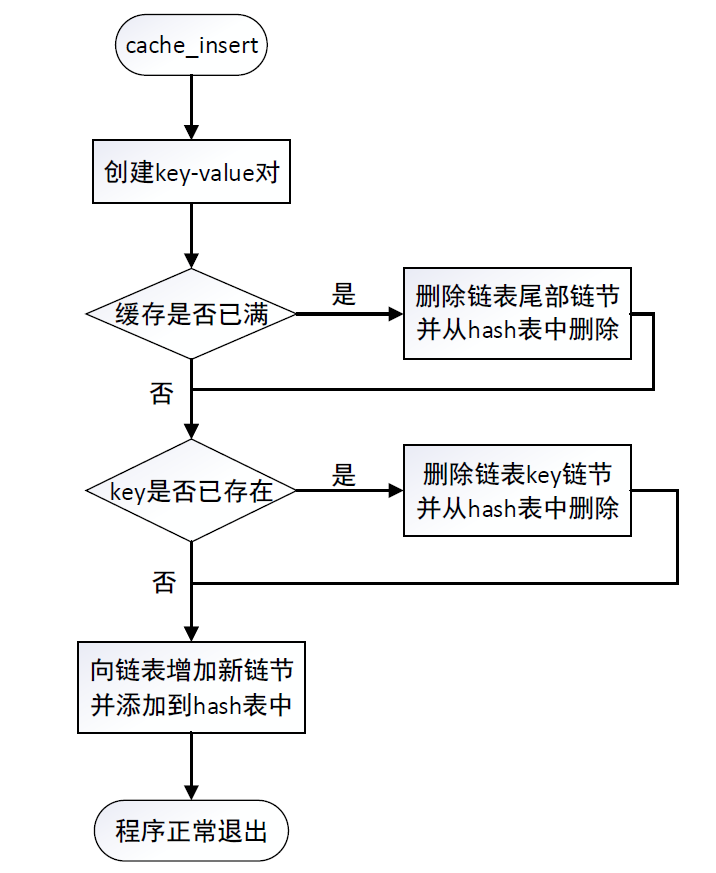




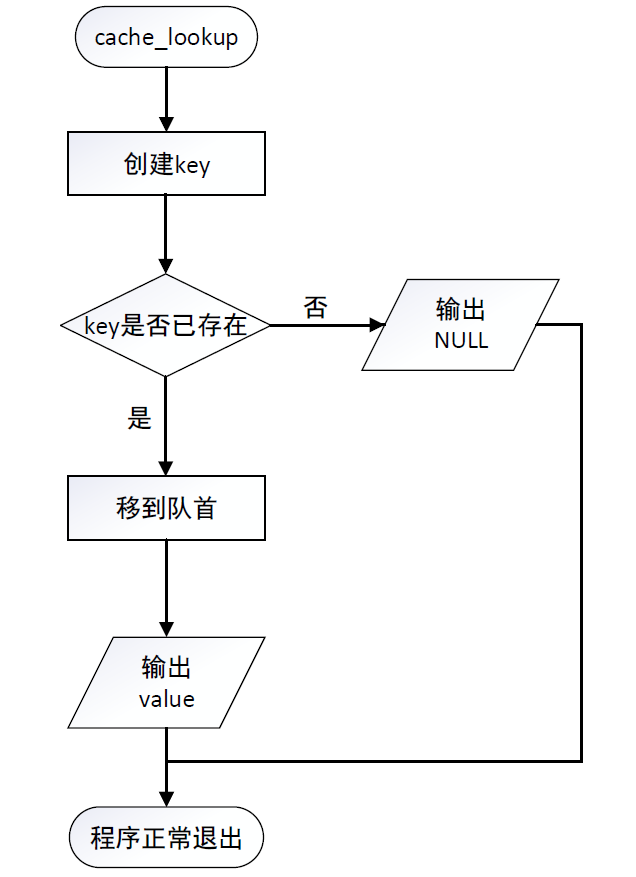
## cache模块的整体运行流程图



## cache模块插入处理流程图



## cache模块查找处理流程图



1. 测试用例以及运行结果

经过测试，本程序成功实现了不良网站拦截、服务器功能、中继功能，并且能够进行超时重传等处理机制，由于使用了多线程机制，本程序在多客户端访问、多主机访问等并发访问的情况下都有较好的表现。

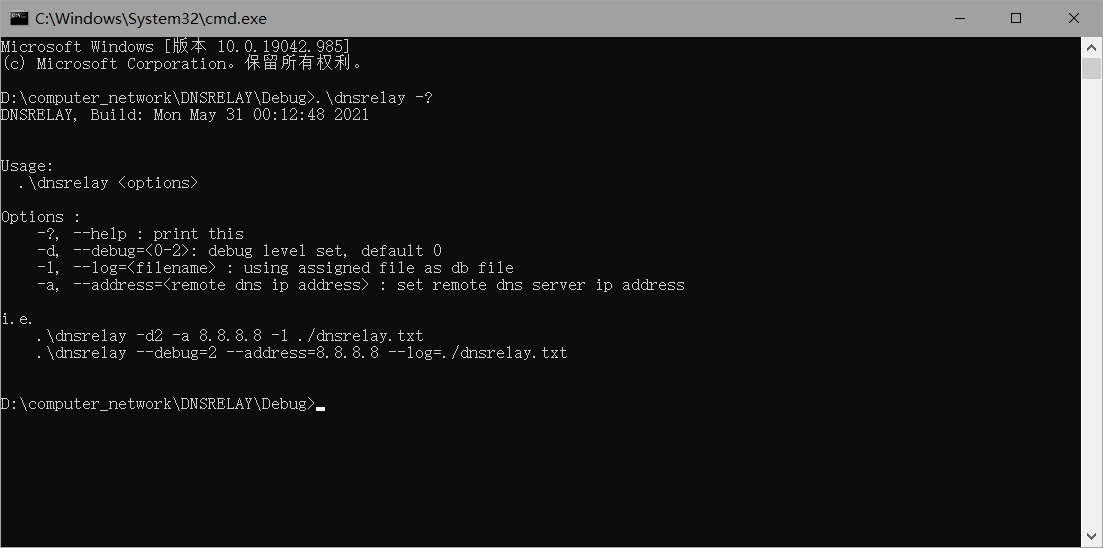
首先，我们可以测试单主机情况下的运行情况。将本机的DNS服务器地址改为回环地址“127.0.0.1”



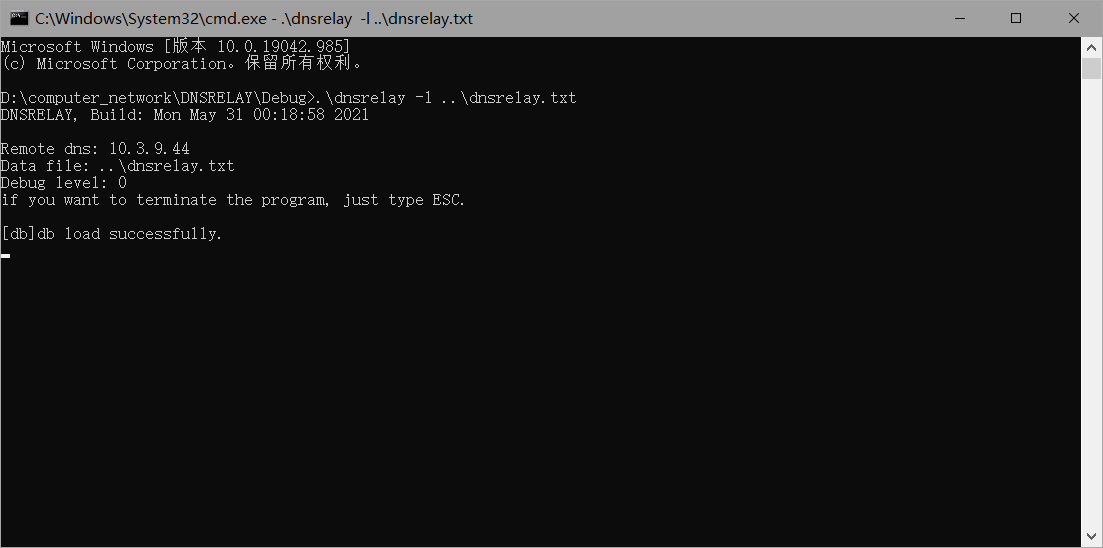
## 命令行界面

本程序实现了一个命令行交互界面，用于让用户设置远程服务器IP地址、“域名-IP地址”映射表以及debug等级。

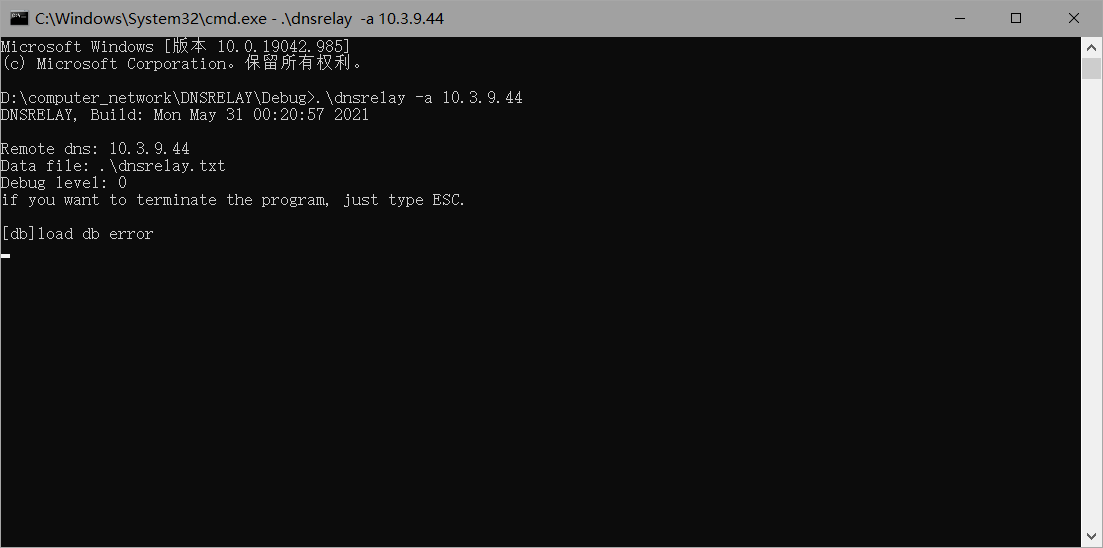
在命令行选项中输入“-?”或者“--help”，则会跳出本程序命令行交互的使用方法。



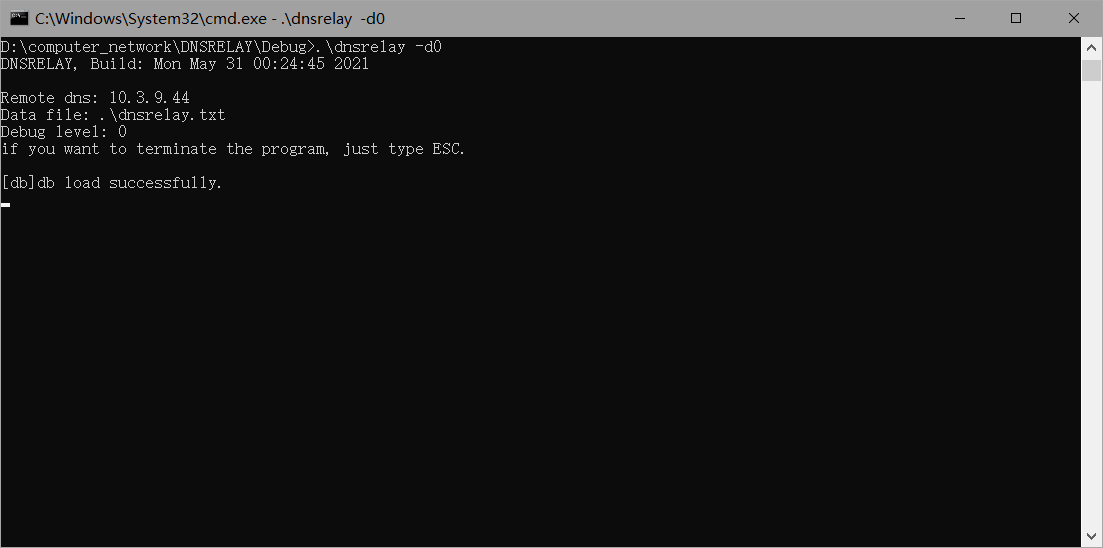
在命令行选项中输入“-l”或者“--log=”，随后输入本地“域名-IP地址”映射表的位置，则能够选择对应的文件作为本地的映射表。



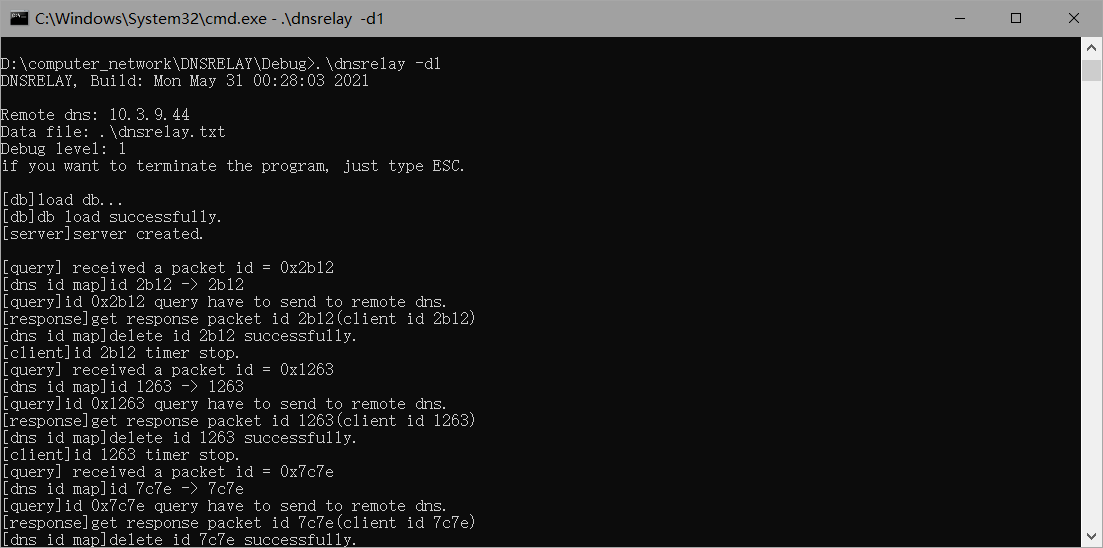
在命令行选项中输入“-a”或者“--address=”，随后输入远程服务器的IP地址，则能够选择对应的远程服务器作为中转请求的远程服务器。



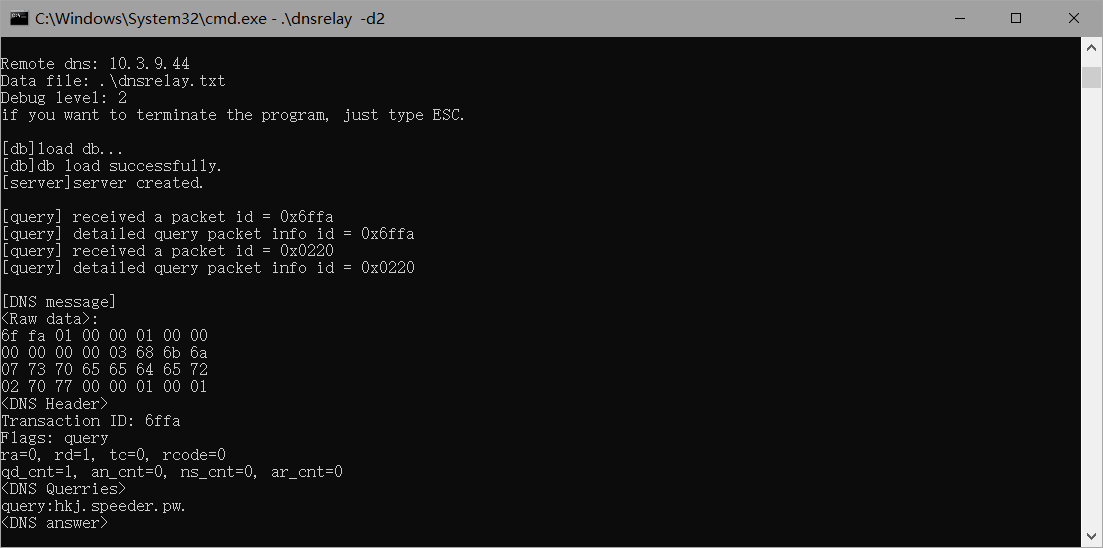
在命令行选项中输入“-d”或者“—debug=”，随后紧跟数字0-2，则能够选择对应的debug等级。在不同的debug等级下，程序将会在命令行界面上打印不同的调试信息，对于debug等级为0，则几乎不会打印程序运行过程中的各种详细信息。而对于debug等级为1，程序将会输出每次处理的信息；debug等级为2，程序将会输出每次处理的具体DNS报文解析后的结果。



debug等级为0，运行过程中不显示任何具体调试信息



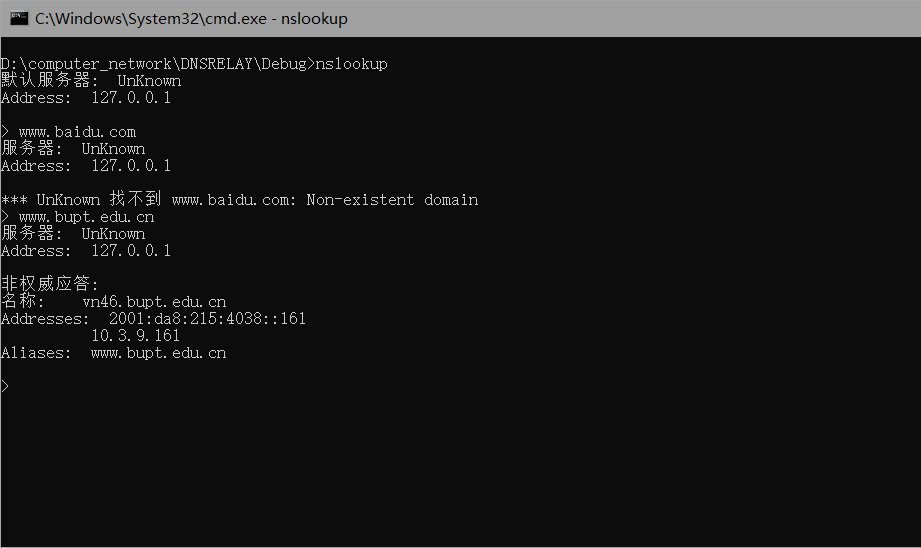
debug等级为1，会显示具体的处理信息



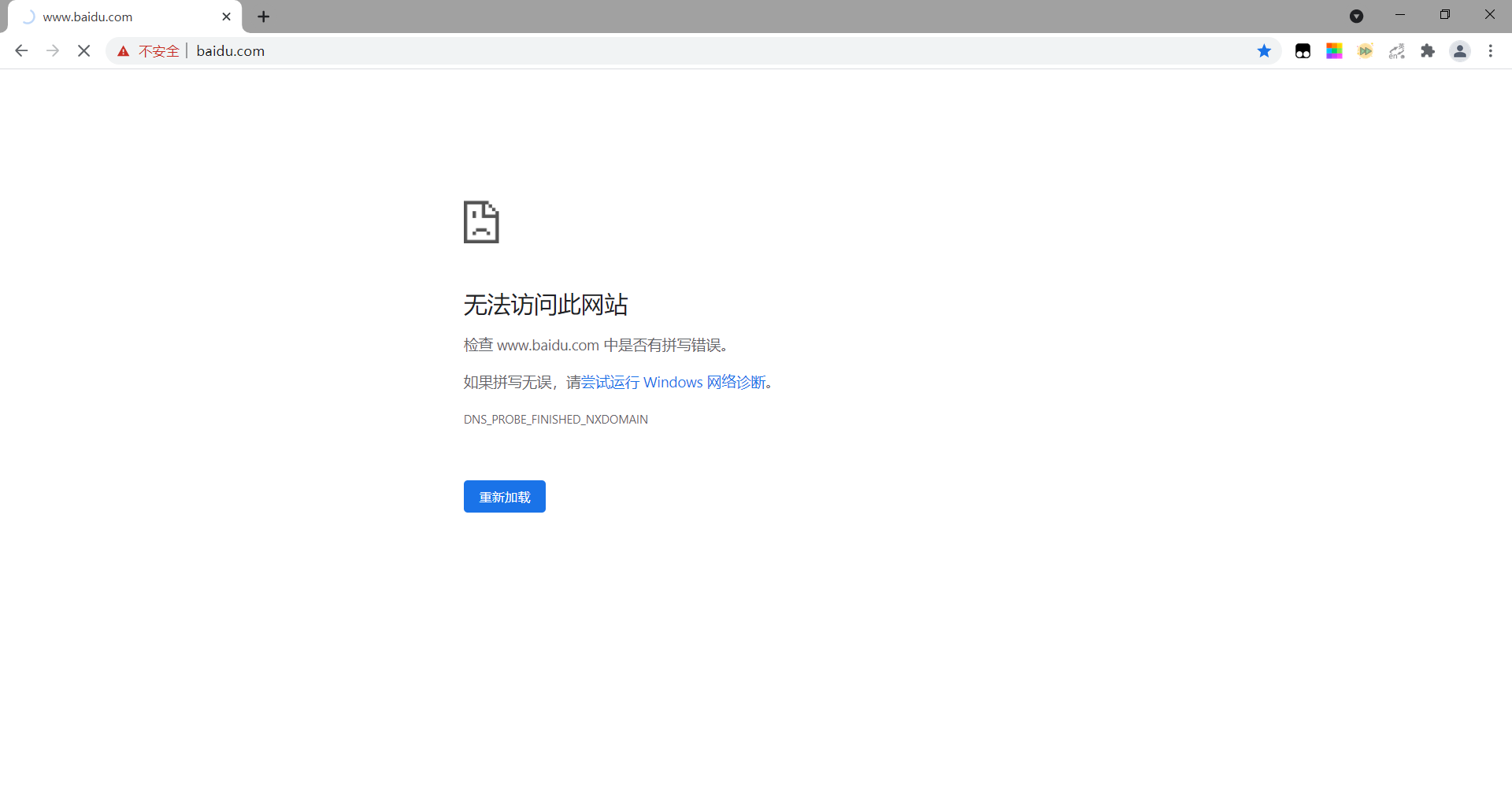
debug等级为2，会显示出处理的每条DNS数据包报文内容

## 不良网站拦截

如果需要拦截某些网站，可以在“域名-IP地址”映射表中添加表项“0.0.0.0 域名”这一字段，从而实现对某些不良网站的拦截。在本次测试中，我们首先在映射表中加入[www.baidu.com](http://www.baidu.com)作为待屏蔽的不良网站，首先输入“ipconfig/flushdns”，清除当前本地的DNS缓存，随后再打开本程序，使用nslookup进行域名查看，则此时显示“UnKnown找不到 www.bupt.edu.cn: Non-existent domain”，而对于不被屏蔽的网站，则返回正常结果。



此时登录浏览器，同样发现域名为[www.baidu.com](http://www.baidu.com)的网站由于DNS解析的问题，无法正常访问。



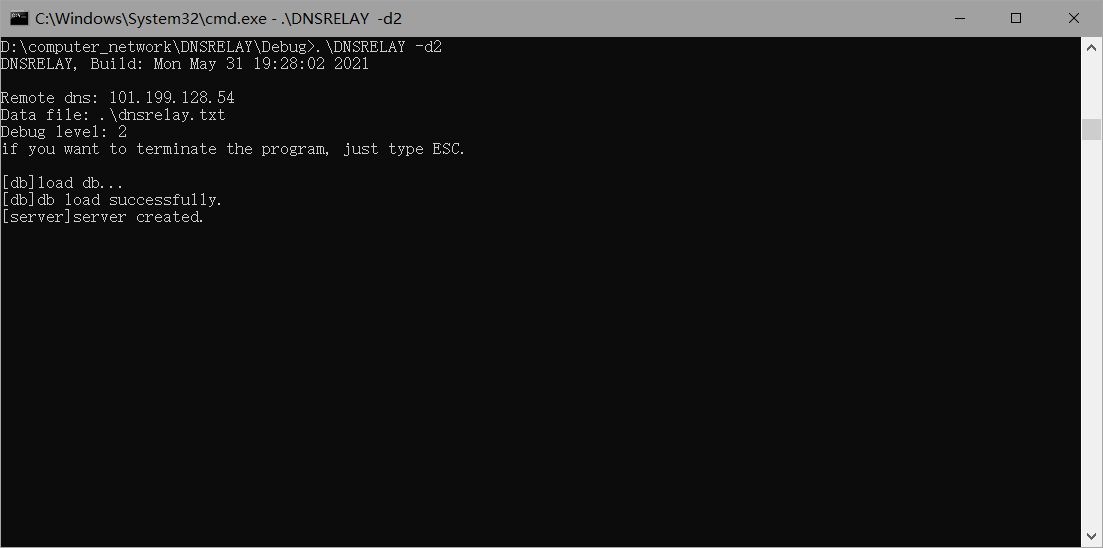
而对于正常的网站，本程序则可以正常访问。



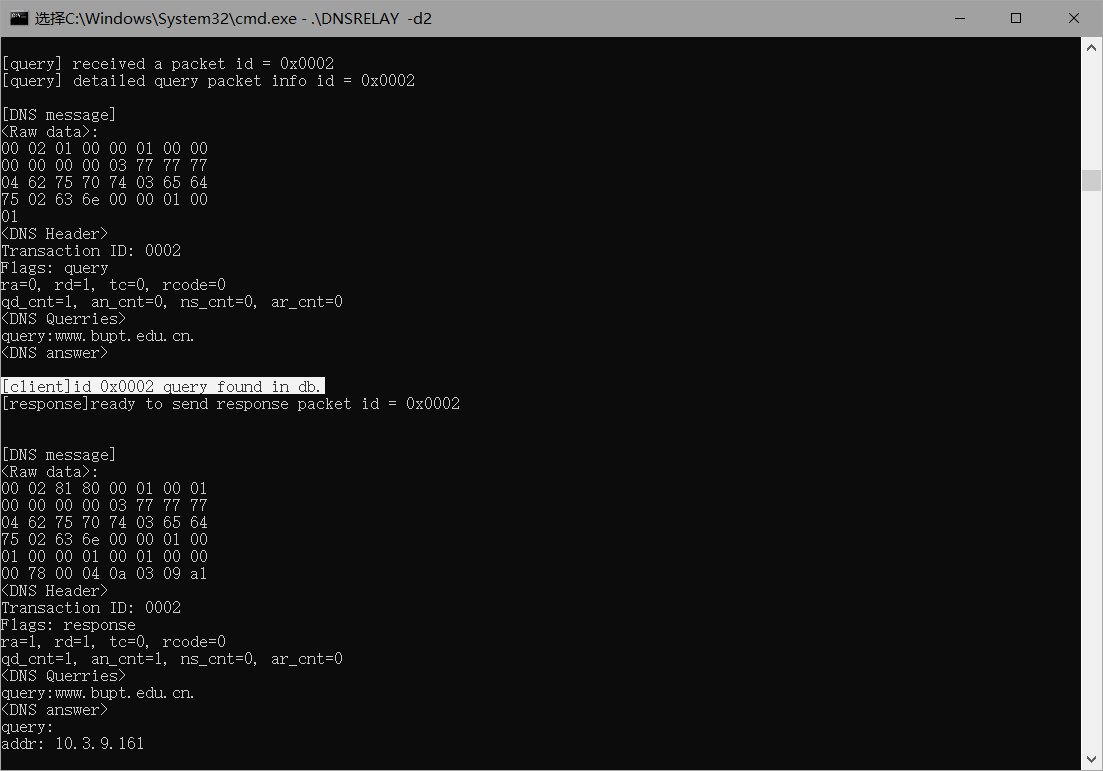
## 服务器功能

结合命令行工具nslookup与命令行界面给出的调试信息对检测结果进行确认。

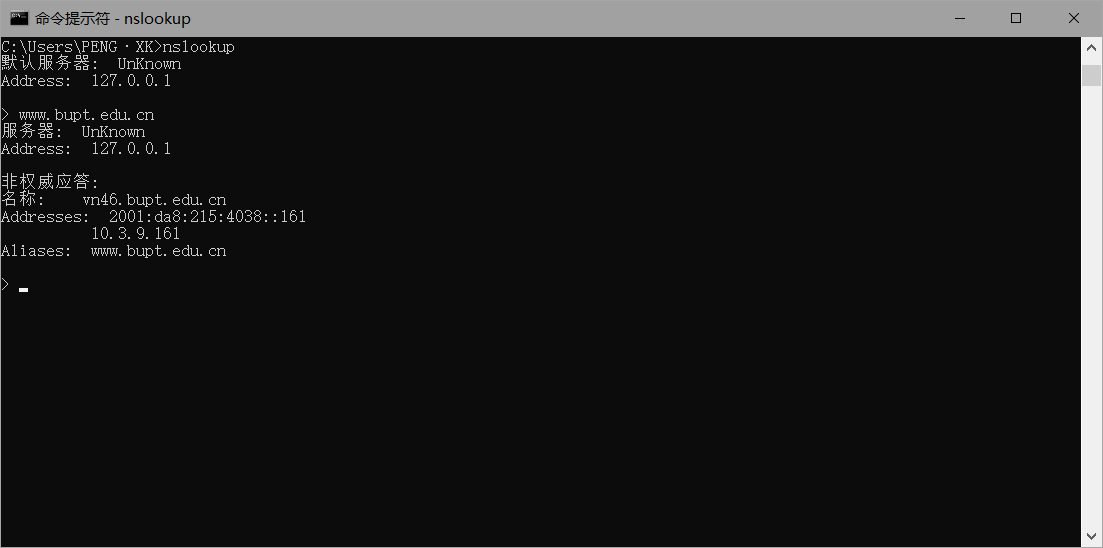
在“域名-IP地址”映射表中，写入“10.3.9.161 [www.bupt.edu.cn](http://www.bupt.edu.cn)”这一行，打开程序，将调试等级调整至2级。



此时在nslookup中输入“[www.bupt.edu.cn](http://www.bupt.edu.cn)”进行查询，最终成功在本地中找到对应表项并成功发出响应报文。不过，由于本程序并不支持对于DNS问题类型CNAME或AAAA的解析处理，因而返回的响应报文结果中仅有其ipv4的IP地址，对于别名或ipv6结果，则由本程序中继远程服务器所获知。

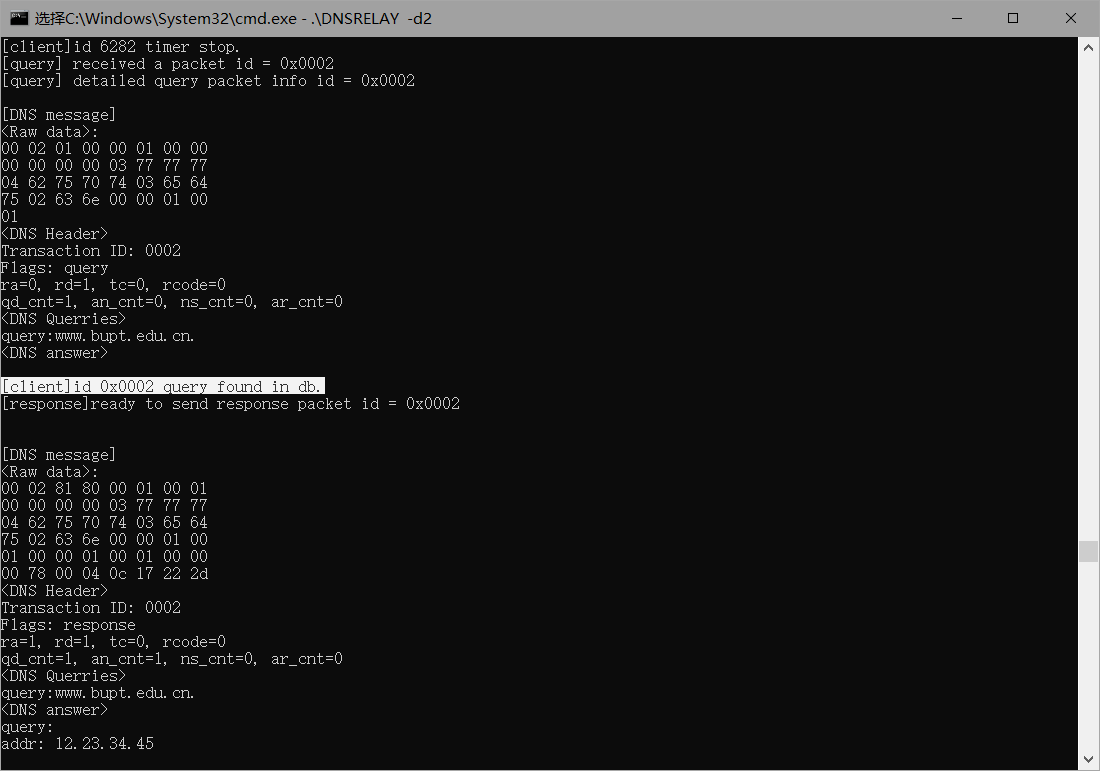


查询“[www.bupt.edu.cn](http://www.bupt.edu.cn)”时，程序在debug等级为2时的调试结果。

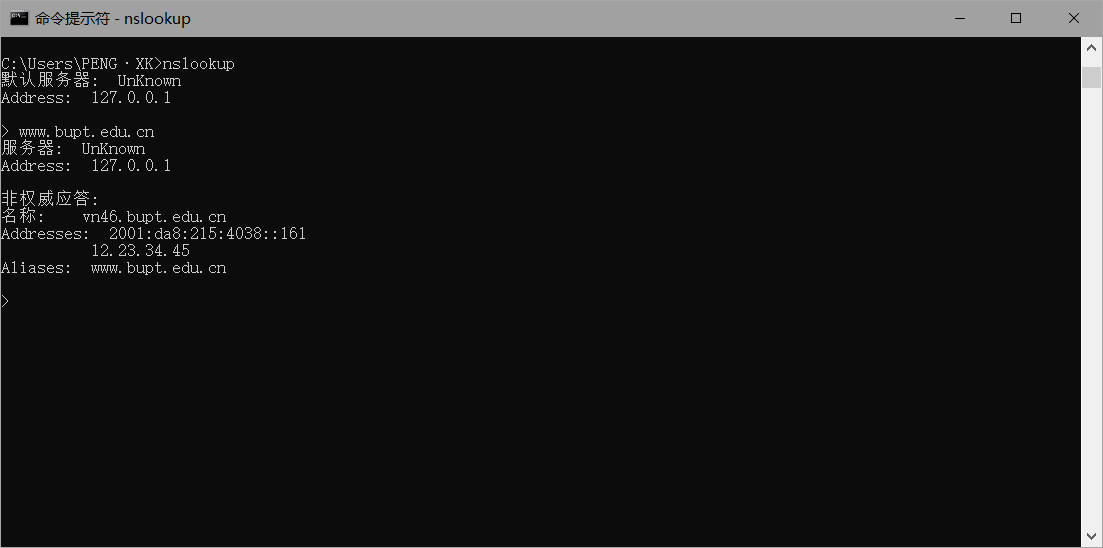


查询“[www.bupt.edu.cn](http://www.bupt.edu.cn)”时，nslookup程序返回的结果

由于本地返回的结果依照“域名-IP地址”映射表中的表项，故将IP地址改为自己设定的任意结果，nslookup的返回值也应该由本地“域名-IP地址”映射表所决定。



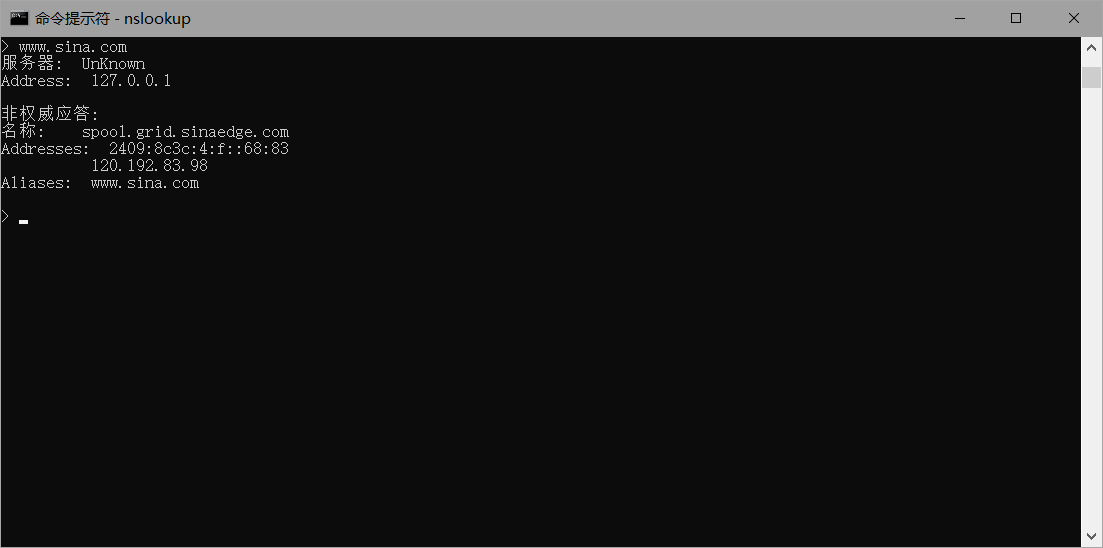
修改本地表项，查询“[www.bupt.edu.cn](http://www.bupt.edu.cn)”时，程序在debug等级为2时的调试结果。



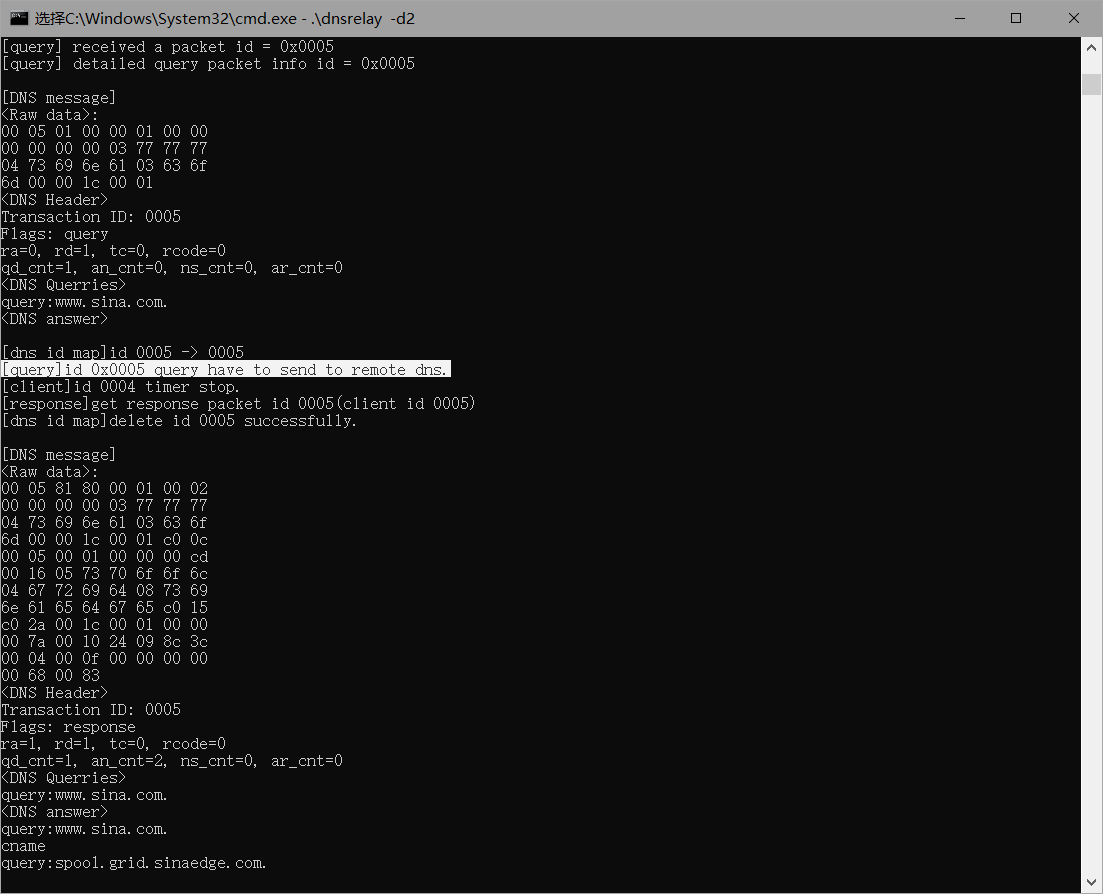
修改本地表项，查询“[www.bupt.edu.cn](http://www.bupt.edu.cn)”时，程序在debug等级为2时的调试结果。

## 中继功能

对于不符合“域名-IP地址”映射表处理要求的DNS请求，本程序将起到与远程DNS服务器之间的中继作用，我们同样可以结合nslookup的返回结果与程序的调试信息对结果进行验证。



对于不在“域名-IP地址”映射表中的域名请求，本程序仍旧能够中继返回正确的结果。



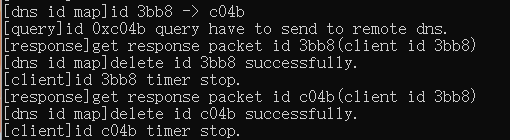
对于不在“域名-IP地址”映射表中的域名请求，程序在debug等级为2时的调试结果，具体显示出了请求DNS报文与中继的响应DNS报文。

## DNS ID转换功能

由于考虑到多个客户端同时访问本DNS服务器，本程序也能处理将同时访问的多个客户端程序相同的DNS ID进行映射，使其能够区分。而当远程DNS服务器返回，或者是定时器结束，则本程序将删除对应表项，回收ID资源。



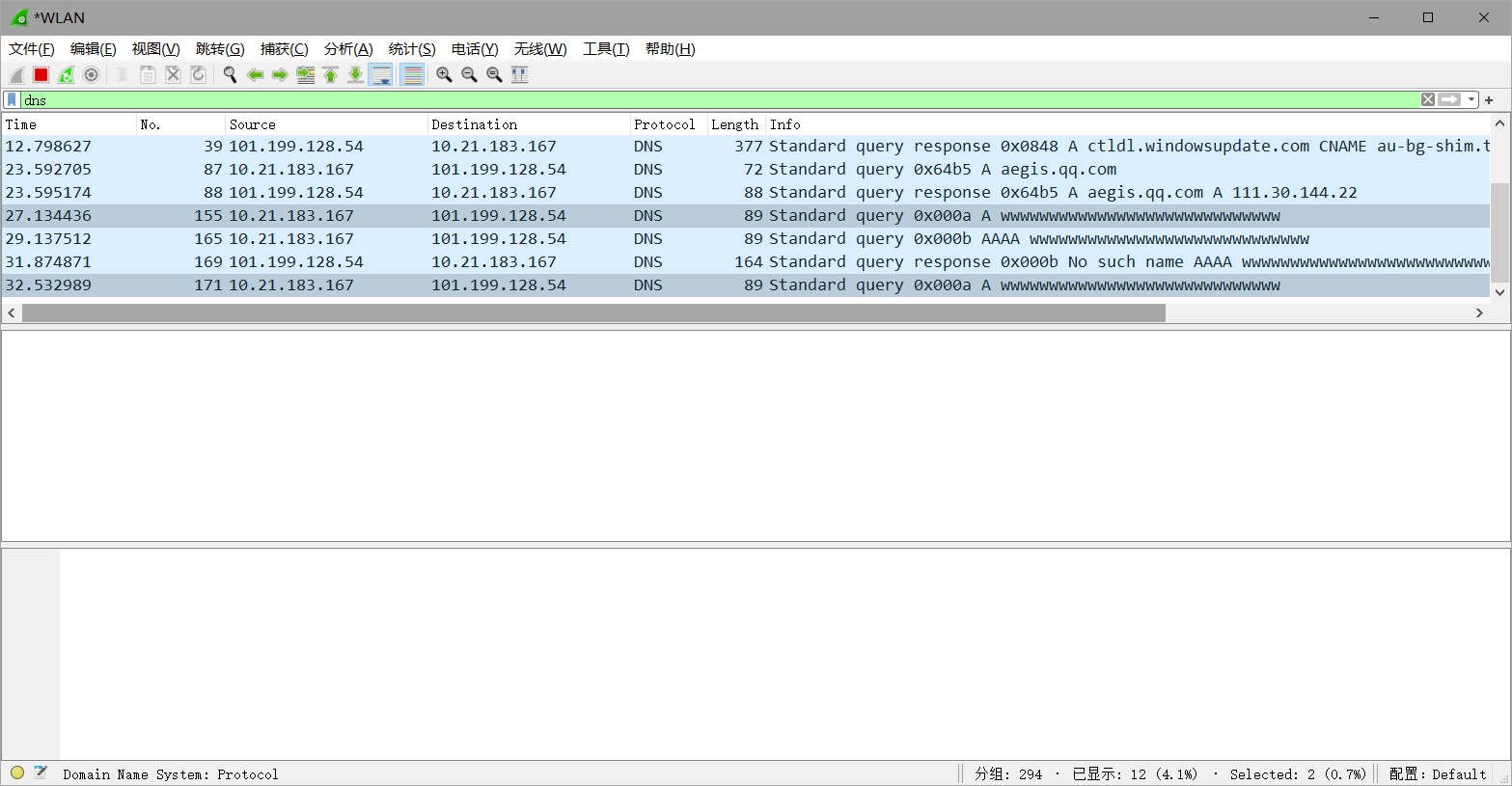
DNS ID冲突时，本程序采取随机算法生成一个新的DNS ID，保证发往远程服务器的DNS请求的DNS ID不同，使其可以被区分。



对应上述DNS冲突，对应的远程服务器返回时，程序能够正确找到其原DNS ID，并删除对应DNS ID的表项，以回收ID资源。

## 超时处理功能

为了更好地进行超时处理，本程序实现了一个独立线程的定时器，定时器单独设置一个线程，进行定时功能，当每轮定时时间结束后，计时器将进行一次超时重传，超时重传仍旧没有回应后，计时器将会执行清理程序，回收对应的ID资源并放弃处理。



通过抓包工具Wireshark，我们能够找到定时超时重发的报文

1. 调试中遇到并解决的问题

在程序的开发过程中，我们遭遇了不少的问题，其中有的问题是出现在程序设计逻辑上的，有的问题是出现在对网络编程的处理上的，还有的问题则是处于编程习惯不良好导致的疏忽问题，这些问题具体表现为以下几个问题：

* 底层数据结构及工具实现

由于本实验要求使用C语言进行编程实现，而且除了标准库和Socket库之外不允许调用其他函数库，所以很多原本可以通过C++ STL或者其他工具库实现的函数和数据结构，我们都必须自行使用C语言进行实现。这给我们的编程带来了一定的困难，尤其是对于红黑树、LRU Cache等较复杂的数据结构，我们需要花费一定的时间去理解其原理，再选择合适的物理结构进行实现。

例如，在红黑树的实现过程中，我们原本对于红黑树的销毁操作采用的是递归实现，然而在调试的过程中，我们发现递归实现对于较多表项而言会导致调用栈次数过多等问题，最后我们又将其通过非递归的方式进行实现，最终使得程序能够正常运行。

* 多线程开发问题

为了提高并发性，本程序采取了多线程的方案来处理较多，而且考虑到移植性和跨平台的特性，相较windows自带的多线程模块，我们选择了POSIX标准多线程库pthread来进行多线程开发。

在进行多线程开发的过程中，我们建立了一个管理和调度线程的线程池以提高多线程的性能。然而，在进行多线程编程的过程中，我们也遭遇了一系列问题，比如对于共享资源的访问问题，如果多线程同时操作共享资源往往会导致线程不安全，我们就不得不通过互斥锁或者信号量对共享资源进行保护，而不谨慎地使用互斥锁很有可能导致性能下降或者死锁。

如何保证多线程的线程安全，并保持线程间的线程同步，给我们的编程带来了很大的挑战。所幸，最后经过不断调试和尝试，最终能够较为稳定的运行。

* 其他问题

由于编程习惯、动态分配内存管理、疏忽等等原因，在编程过程中不免会出现一大堆让人困惑又难以定位的问题，例如文件读写没有安全检测，导致内存泄露，或者是动态分配内存时分配内存片的大小有误等等。

1. 心得体会

在本次计算机网络课程设计的实践过程中，我们小组成员相互合作、互相帮助，采用多线程和网络编程的方法，在仅使用C语言的标准库函数和Socket函数的前提下，完成了具有中继、不良网站拦截、服务器功能的DNS中继服务器。

在本次课程设计的前期工作中，我们主要是通过查阅资料、阅读文档的方式对DNS进行了解，由于RFC1035文档及一些辅助资料结构清晰、介绍翔实，随着对计算机网络课程的进一步了解和掌握，DNS的机制和原理也逐渐变得更加清晰。

课程设计真正的难点则在于程序的编程实现过程。由于我们小组对于多线程编程和网络编程相对而言比较陌生，故花费了不少时间成本和精力去进行相关入门的学习和探索，尽管如此，完成程序的过程中仍旧出现了不少困难和未曾预料的BUG。然而，当我们从对多线程的机制一窍不通，到能够自己实现一个可以运行的线程池。从网络编程时屡屡受挫到最后完成功能。这中间的辛苦和困难固然不计其数，但随之而来的成就感以及积累的经验和知识，终究是让人感到收获满满的。

总而言之，在付出了大量的时间和精力之后，我能够切身地感受到本次计算机网络课程设计为我们日后进一步理解计算机网络知识和网络编程、并发编程的技术奠定了坚实的基础，而其中收获的经验和教训，也必将在日后的学习生涯中使我们裨益良多。