

**《计算机网络课程设计实践》**

实验报告

**学 院 ： 计算机学院**

**团队成员 ： 郑浩 2018210459**

**胡天翼 2018210547**

**张林羽觐 2018210309**

目录

[一 实验目的 3](#_Toc50716303)

[二 实验内容 3](#_Toc50716304)

[三 功能设计 3](#_Toc50716305)

[1. 正常DNS服务功能： 4](#_Toc50716306)

[2. 不良网站拦截功能： 4](#_Toc50716307)

[3. DNS中继功能 4](#_Toc50716308)

[四 需求分析 4](#_Toc50716309)

[1. 多客户端并发处理能力 4](#_Toc50716310)

[2. 超时处理 4](#_Toc50716311)

[五、模块划分 5](#_Toc50716312)

[用户交互层: 6](#_Toc50716313)

[数据管理层: 6](#_Toc50716314)

[网络服务层： 7](#_Toc50716315)

[六、软件流程图 8](#_Toc50716316)

[七、测试用例及运行结果 9](#_Toc50716317)

[1. 不良网站拦截功能 9](#_Toc50716318)

[2. 服务器功能 10](#_Toc50716319)

[3. 中继功能 11](#_Toc50716320)

[4. 多客户端并发 13](#_Toc50716321)

[七、调试中遇到并解决的问题 14](#_Toc50716322)

[八、实验评价 16](#_Toc50716323)

[优点 16](#_Toc50716324)

[缺点： 17](#_Toc50716325)

[九、心得体会 18](#_Toc50716326)

## 一 实验目的

通过该实验，进一步巩固和深刻理解计算机网络体系结构特别是其中的DNS协议，了解和掌握网络编程的基本方法和基本机制，提高同学的编程能力和实践动手能力，体验网络编程在设计上各种问题和调试难度，巩固和深刻理解理论知识并利用这些知识对系统进行优化。

## 二 实验内容

设计一个DNS服务器程序，读入本地的“域名（域名）-IP地址”对照文件，并建立cache，当客户端查询域名对应的IP地址时，用域名检索该对照表以及本地cache，存在三种检索结果：

1、检索结果为特殊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、若本地列表中未检到该域名，则向外部DNS服务器发出查询请求，并将域名解析结果返回给客户端（中继功能），并将此次记录加入cache中。

## 三 功能设计

#### 1. 正常DNS服务功能：

将计算机的DNS服务器地址指向本机，当用户输入域名时，DNS服务器需要从本地存储的域名-IP对照表以及cache中进行查询，将查询到的普通IP地址返回给用户进行正常的网址访问。

#### 2. 不良网站拦截功能：

当用户想要访问某个网站输入其域名时，DNS中继服务器需要获取其查询信息，将用户想要访问的域名与中继服务器中存储的域名-IP对应表进行对比。如果查到IP为全0，向用户返回特殊的数据包，实现不良网站屏蔽功能。

#### 3. DNS中继功能

当DNS服务器本地存储的域名-IP对照表中没有存储用户想要访问的域名，向外部DNS服务器发出查询请求，并将查询到的结果返回给客户端。

## 四 需求分析

#### 1. 多客户端并发处理能力

作为DNS服务器，可能会同时收到位于不同计算机上多个客户端的查询请求。即：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。我们设计了多线程处理多个客户端的查询请求，只要有新的查询请求出现，就将其加入一个新的线程处理。

#### 2. 超时处理

由于UDP的不可靠性，考虑求助外部DNS服务器（中继）却不能得到应答或者收到迟到应答的情形。由于发出查询请求到收到应答报文之间存在一定时长，所以需要设定合适的阈值来判断是否超时。若向外部DNS服务器发送请求后超时，我们将继续重传请求，若在三次超时重传后仍得不到外部DNS的应答，我们将给客户端返回不可达信息。

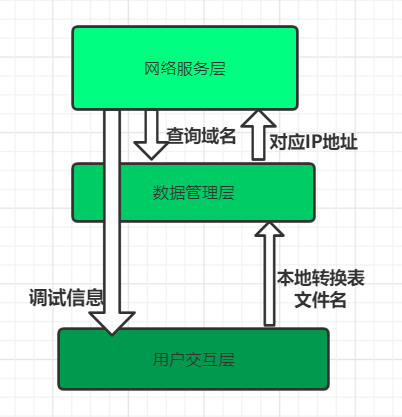
## 五、模块划分

根据团队对于DNS中继服务器的需求分析，我们将服务器分成用户交互层、网络服务层以及数据管理层的三层结构。

（1）用户交互层主要用于接收管理员输入的调试等级信息，本地域名-ip转换表文件名以及外部DNS服务器地址，并将所需日志信息打印输出到文件中反馈给管理员用户；

（2）网络服务层接收客户端发来的DNS查询请求，查询域名对应的IP地址时，用域名检索该对照表以及高速cache，进行正常DNS服务、不良网站拦截和DNS中继这三大功能。同时，我们对应用服务层的功能进行了合理地拓展，包括设计了多线程处理，并为其增添了超时机制；

（3）数据管理层实现了Hashmap和高速cache，将本地数据读入map，用网络服务层收到的域名检索该map返回对应的ip地址，并存储域名-应答报文信息到cache中，整个层次为网络服务层提供服务。



模块调用关系图

**各层次主要函数：**

#### 用户交互层:

void Process\_Parameters(int argc, char \*argv[])

作用:处理命令行输入参数，可以包含调试等级、本地对照表文件名、外部DNS服务器IP地址。

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

void Output\_Packet(char \*buf, int length)

作用：将接收到的数据包以十六进制的格式按字节进行打印展示。

#### 数据管理层:

int findDomainMap(const char\* const domain, MapHandle maphandle,IPV4\* result)

作用：在map中根据域名进行查询，得到对应的ip地址

int loadMapFromFile(const char\* const filename, MapHandle maphandle)

作用：从本地文件中读入域名-IP地址对应表，存储到map中。

void insertPacket(PacketPair\* const pairArray, size\_t arraySize, const PacketPair\* const insertElement, HANDLE Mutex)

作用：将从外部收到的DNS应答报文信息结合LRU算法存到本地cache中，方便下次直接从cache中查找。

PacketPair\* findPacket(PacketPair\* const pairArray, size\_t arraySize, const char\* const domainName)

作用:在本地cache中根据域名查找应答报文的信息。

#### 网络服务层：

int sendtoDNSServer(const UINT8\* buf, int len)

作用：将组装好的报文转发给外部DNS服务器

void recvfromDNSServer(UINT8\* buf, unsigned int\* len)

作用:接收外部DNS服务器发送的报文

unsigned char\* TransName(const unsigned char\* name, int len)

作用:将域名转换为查询名，如www.bupt.edu.cn -> 3www4bupt3edu2cn0

void resolveDomainName(const unsigned char\* name, int len, IPV4\* result, char\* answers,size\_t \*answerslen, size\_t\* pResultsCount)

作用：解析未知域名，设置并发送DNS请求报文，接收并解析收到的DNS报文，得到DNS服务器发回的IP地址。

## 六、软件流程图

## 七、测试用例及运行结果

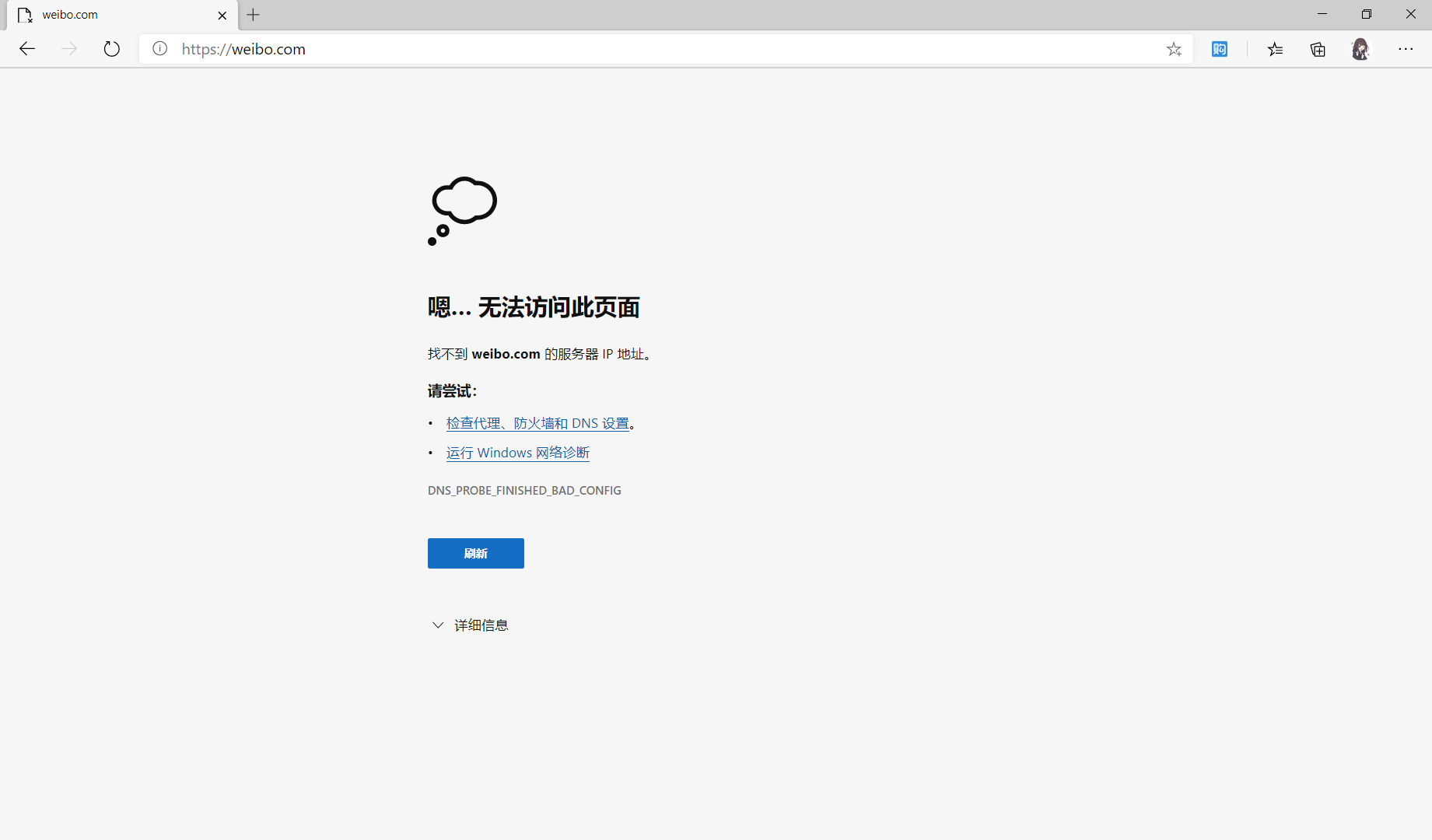
我们设计的DNS服务器经过了大量的试验及测试，在多用户访问以及单用户多访问情况下都有着较好的表现。并且，在DNS服务器功能上本服务器实现了不良网站拦截功能、服务器功能以及中继功能。为了提高服务器性能和效率，我们也在组装数据包、本地数据存储等方面也做了一些拓展与优化。接下来本部分将从以下六个方面展示DNS服务器的功能与特点（各方面中可进一步细分为多种情况）：

1. 不良网站拦截功能；
2. 服务器功能；
3. 中继功能；
4. 多客户端并发处理；
5. Cache功能模块及LRU算法机制。
6. Hashmap功能模块

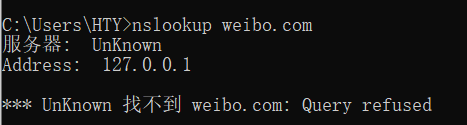
实验前，我们先将计算机本地TCP/IPv4协议配置属性中的DNS服务器设置为本地环回IP地址“127.0.0.1”

#### 1. 不良网站拦截功能

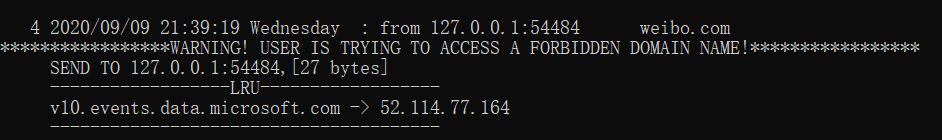
尝试将weibo.com设置为不良网站（0.0.0.0），使用浏览器结果为：



Nslookup结果为：



调试信息为：



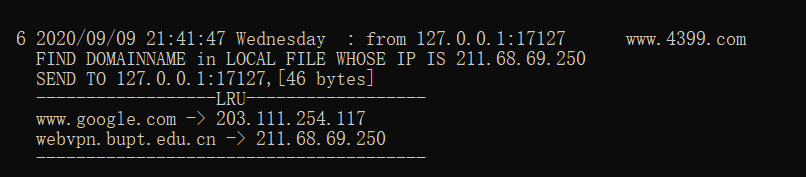
#### 2. 服务器功能

尝试在dnsrelay.txt中将[www.4399.com的IP](http://www.4399.com的IP)地址设置211.68.69.250（北邮webvpn系统）

在浏览器中访问www.4399.com



查看调试信息：



服务器在本地本地文件中找到[www.4399.com的IP](http://www.4399.com的IP)地址，并将其返回。

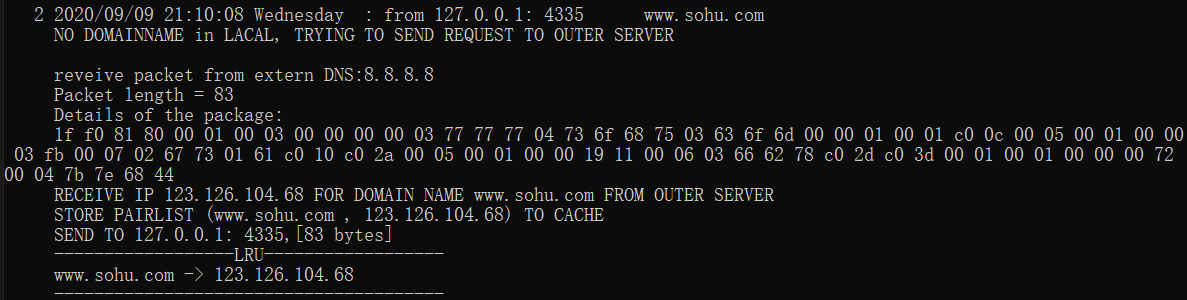
#### 3. 中继功能

访问不在dnsrelay.txt中存储的域名sohu.com

连续访问两次，两次的访问结果都正常：

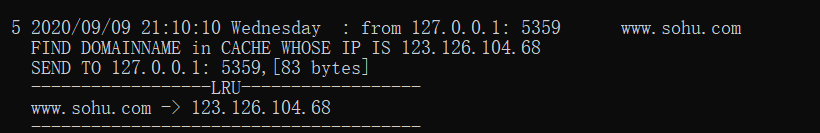


第一次访问的调试信息：

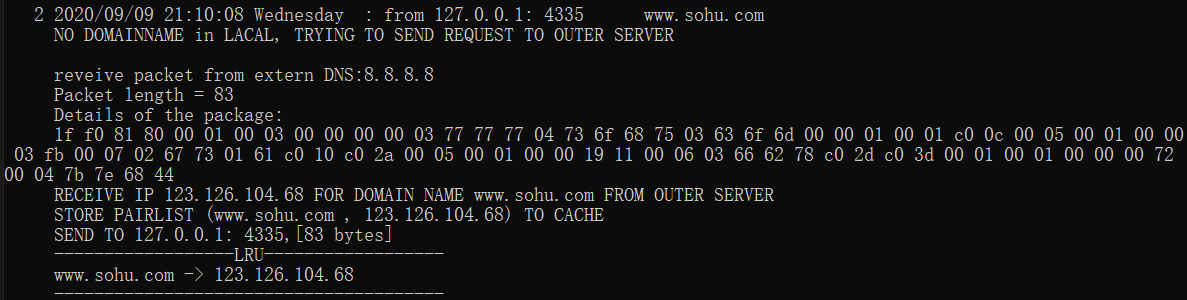


即在本地文件与cache中均未找到sohu.com，尝试向外部DNS服务器查询，得到查询结果123.126.104.68，并将其存入cache

第二次访问的调试信息：

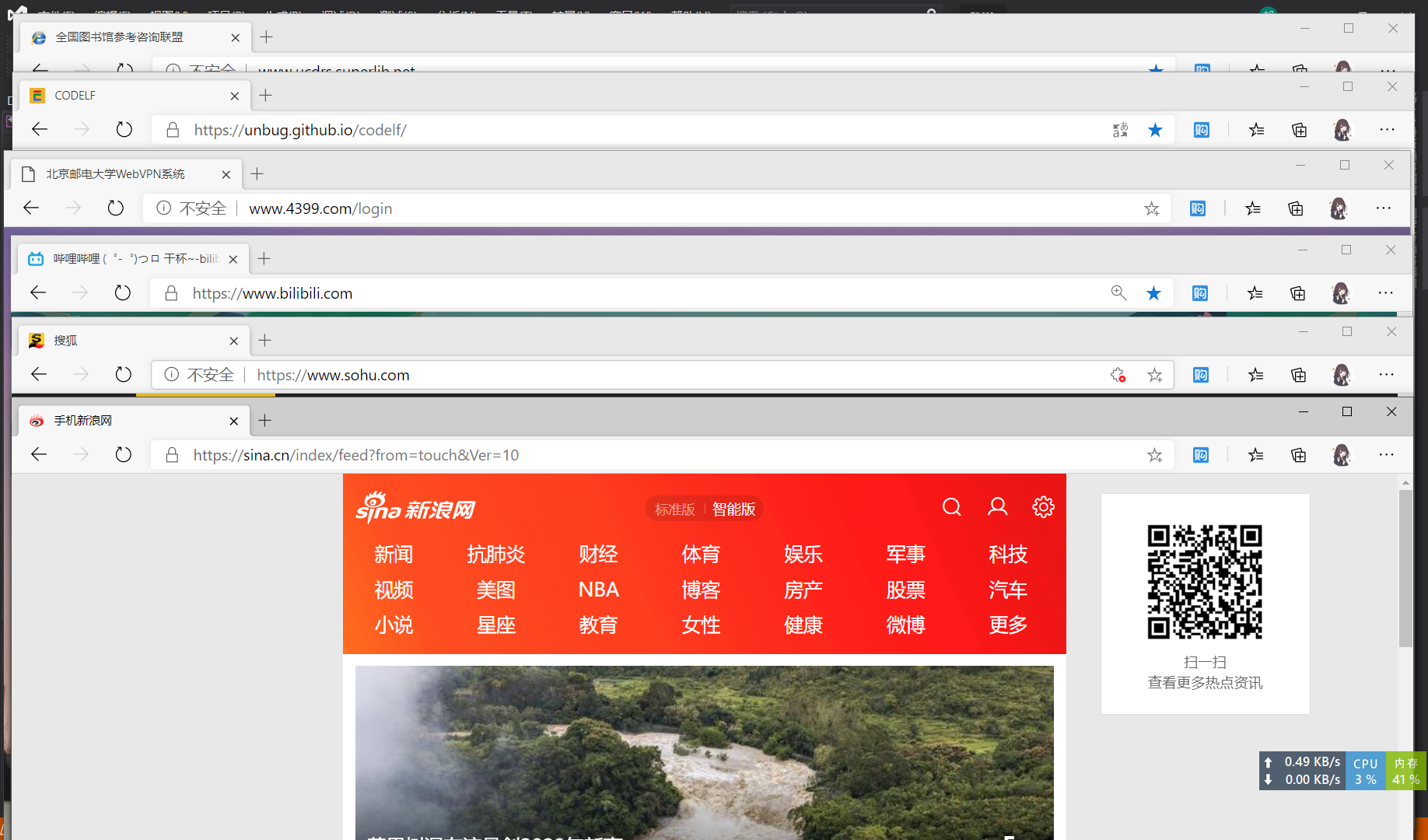


从cache中读取了sohu.com的IP地址。可以证明cache有效



即在本地文件与cache中均未找到sohu.com，尝试向外部DNS服务器查询，得到查询结果123.126.104.68

#### 4. 多客户端并发

****

****

同时打开6个窗口，DNS服务器准确并快速地回复了全部的响应。在监视窗口中，可以看到内存使用在短时间内达到很高，但之后得到释放，由此可以看出我们的程序在内存管理方便较为良好。

## 七、调试中遇到并解决的问题

**1.报文结构中的变长字段的存储问题**

报文的基本结构包含报文头部、报文问题部分、报文回答部分。在编写存储报文的数据结构时，我本想用一个有三部分结构体组成的大结构体struct {HEADER, QUESTION, ANSWER}来存储报文。但经过阅读相关资料，我了解到报文问题QUESTION部分中的QNAME字段，也就是编码过的域名是变长的。要想在结构体中存储一个变长的字段，只能在结构体中定义一个定长的指针，而后申请动态的内存空间。但申请动态的内存空间后，又无法保证此段空间与报文的其他部分相连，因而报文的结构完整性无法得到保障。

为了解决变长字段的存储问题，我放弃将所有报文结构存在一个结构体内，而考虑分段存储，即分开组装每一部分的报文，而后将数据桉结构顺序拷贝到一个buf指针所申请的一段连续内存上。为了达到这个目的，我定义了一个拷贝函数：

void copytoDNS(UINT8\*\* const dst, const UINT8\* src, int len)  
{  
 memcpy(\*dst, src, len);  
 \*dst += len;  
}

该函数可以拷贝数据，并自动地移动指针到拷贝结束地位置。

首先组装定长地报头HEADER，将其拷贝到报文上。而后拷贝不定长的编码域名，将其拷贝到报文上。最后再组装定长的QUESTION信息（查询类型和查询类），将其拷贝到报文上。这样就较好地完成了一个DNS查询报文的组装。

**2.不同类型应答报文的结构问题**

在组装拒绝报文（对于在黑名单上的域名）和组装超时报文（对于DNS请求超时）、以及对客户端返回服务器不支持的应答报文（对于ipv6请求）来说，只需要告诉客户端结果。我根据RFC1035文档分别对报文头部中FLAG字段的RCODE设置不同值来实现不同的应答信息。但是在向客户端返回这些报文时，我发现客户端接收到报文后并不能正确地识别我想要传递的信息。

通过使用wirehark抓包，我发现我向客户端传递的这些报文被标注了“结构错误”的信息。通过阅读相关资料，仔细研究报文结构，我了解到回复报文时必须要保留QUESTION部分，不论是正常应答的报文还是回答错误信息的报文。因此，在组装这些不用类型的报文时，在修改HEADER部分后，依然要原封不同的加上原先的请求报文得到QUESTION部分，这样客户端才能正确识别信息。在加上这一部分后，wirehark和客户端均可以正确识别回复的这些报文了。

**3.通过域名解析出IP地址后，无法通过这个IP地址直接访问页面**

在设计DNS服务器的高速缓存cache时，原本我们的设计方案是存下域名-IP地址键值对，在下次请求这个域名时在cache中寻找这个上次对这个域名解析出的IP地址并返回。从而避免了反复查找带来的时间浪费。但是在这样实现的过程中，我们发现有些网站通过cache返回的IP地址无法正常访问页面，且经检验，cache的设计并无问题。经过搜索资料，我了解到：

一般的网站会选择放在虚拟主机，且在主机上放置了很多个网站，而每个网站绑定1个或以上域名，虚拟主机上，例如Apache主机的配置会将对应的ip解析到对应的网站目录的，实现一台服务器上配置多个站点；一般用户在访问的时候，会产生一个http请求报文，上面的host信息可以提供给服务器，告诉服务器要访问的域名，从而实现一台主机绑定一个IP，即使有多个网站，也不会相互干扰。但使用IP访问，主机不知道用户访问的具体目录，请求便会出现错误。如果就想用ip形式访问，可以手动设置Host头域信息。

但是我们小组对于Host的实现并不了解。经过讨论，我们决定在cache中存储解析域名时返回的完整DNS应答报文，于是再次请求这个域名时同样返回完整的DNS应答报文。这样虽然增大了cache的所需空间，但也保证了cache的可靠性。

**4. 多线程在更新Cache数组和查找Cache数组中之间的竟态:查找后到返回报文之间如果Cache被修改，则有可能返回错误的报文**

添加Mutex互斥锁，在更新Cache数组和获取结果直到回复报文之后加互斥锁，确保程序运行的正确性

**5. 内存泄漏，有几个函数出口(return语句)附近，以及更新Cache数组时未释放堆上分配的内存，在长时间运行的程序中应避免内存泄漏**

使用VS的内存分析工具，截取堆上的内存快照，分析并解决代码中可能存在的内存泄漏的情况。在空闲时，程序只在堆上分配了加载map所需的内存。

**6. 无法绑定localhost的53端口**

用netstat以及tasklist分析占用53端口的进程，发现是ICS服务占用了localhost:53，打开services.msc禁用该服务

## 八、实验评价

#### 优点

1. 构筑了Hashmap结构，提升查询效率

在本地存储的域名-IP映射表的处理上，我们并没有简单地存储在结构体中，在查询时一个个地遍历。为了提升效率，我们手写了一个较为完善的Hashmap结构。在这个数据结构中查询域名对应的IP地址，能够做到O（1）的时间复杂度。在Hashmap的完善上，我们也做了诸多努力：

首先，这个Hashmap支持自定义如何给key赋值、如何给value赋值、key值比较方法等多种操作，泛用性高；

其次，我们定义了最大负载因子与桶扩容函数，根据桶的大小与桶中元素数量的大小关系决定是否需要扩容，在不浪费内存的情况下保证查询效率；

并且，在Hashmap中我们还使用了void\*来解决C中的泛型问题；

最后，我们还实现了较为良好得到封装，在外层程序中不直接使用Hashmap结构而使用一个间接访问的下标mapHandle。

在实际测试中，这个Hashmap结构也确实带来了极大的便利与高效。

2. 内存管理良好，长期运行稳定

我们在编写代码时，仔细地考虑了每个地方申请的内存是否得到了释放，严格地控制内存管理。这样可以保证服务器长期稳定的运行，哪怕是短时间内有大量数据访问，也能在访问结束后迅速释放不用的内存。测试过程如下：

****

3. 多线程间冲突管理

在使用多线程时，不同线程同时写入一块内存时会发生冲突。因此我们在设计多线程时，在可能引起冲突的地方加入了互斥锁，如果有若干进程要写入同一块内存，则只允许一个进程进行写入，而阻塞其他进程。从而保证了DNS服务器的稳定性。

4. 高速缓存的使用

由于部分网站不可以用IP地址直接访问，因此我们在设计高速缓存Cache时，并没有选择只存下域名-IP地址键值对，而是将域名查询到的报文内容全部存下，在下次再次请求这个域名时将完整报文返回。这样做虽然降低了部分效率，但保证了访问服务的稳定性。同时，在Cache的替换策略上，我们选择了LRU算法，记录Cache中每条记录的过期时间，访问或存储时刷新过期时间，由此保证Cache中总存储着最近请求的域名。

#### 缺点

在逻辑线程较少的机器上处理效率较低，伪造报文有可能无法直接访问ip。

## 九、心得体会

胡天翼：

在本次DNS中继服务器的设计实验中，我结合了《计算机网络》理论课学习的传输层、应用层通信原理的基础知识和过去学习的C语言编程知识，成功地完成了一次用高级语言对底层系统的模拟实践。实验过程中，我将理论知识应用于实践，得到了进一步的掌握与巩固。  
 在本次实验中，我们完成了一个具有黑名单、本地cache缓存功能的中继服务器 。通过这次实验练习，我对网络中DNS通信过程有了一个初步的体验。一开始，我对如何构建一个完整的通信过程毫无头绪，通过反复阅读PPT、RFC1035文档和查阅相关资料，理解实际DNS通信的过程，我终于对如何构建中继服务器有了初步的认识。在编写代码的过程中，我也因忘记了C语言的部分基础知识，导致代码编写混乱，出现未知的BUG。通过请教同学老师，反复调试程序，我最终解决了若干问题，正确地完成了功能的实现。在调试过程中，我因客户端接收到的结果与预期不符感到困惑。通过wireshark抓包，分析传递报文的具体结构与出现的问题，我最终明白报文发送代码中存在的缺陷，调整代码后最终发出了正确的报文。  
 这次实验我总共花了将近两周周的时间，虽然过程是艰辛的，但是结果确是让人回味的。最后，衷心地感谢所有给予我帮助的老师和同学们！

张林羽觐：

本次实验，加深了我对C API接口设计的理解，也理解了C++使用引用传参，重载运算符，构造函数，析构函数，带来的方便，更理解了HashMap的数据结构，插入，Rehash的过程，对于如何处理内存泄漏，线程间同步的理解也有所加深。

郑浩：

本次设计DNS中继服务器的实验中，加深了我对计算机网络体系结构的理解以及让我对socket网络编程进行了实践，最后完成了本次的DNS中继服务器课设作业，在组员的共同努力下实现了所有功能并且也保证了服务器长时间运行时的稳定性。

在课设开始之前，我已经学习过《计算机网络》，了解网络通信的基本流程，但是对于DNS中继服务器在网络中起到的作用还不是太了解，只是停留在了域名解析这一步，借助这次机会，我阅读了RFC 1035以及网络上关于DNS服务器的其他资料，这才更加深入的了解了DNS的机制及原理。同时，在本次作业调试过程中我也学会了网络编程的重要方法，结合wireshark进行抓包，分析我们设计的服务器发出的报文，同时可以与正常的DNS报文进行对比，借此对DNS报文格式更加熟悉了也方便我们调试。

这次服务器设计加入了多线程让我对较复杂多线程的程序调试有了更多的经验，在调试中线程在读本地cache以及输出时发生线程之间的冲突，利用VS查看不同线程的执行情况结合单步调试等工具，让我更快的发现问题，并和队友一起提出加入互斥锁的想法，这解决了线程竞争的问题。

虽然这次作业从学习socket编程、学习DNS服务器原理、报文结构等花费了挺长一段时间，但是最后能和队友一起实现这样的一个程序还是非常值得的。感谢在这过程中给予我指导帮助的每一位同学老师！