

计算机网络课程设计实验报告

课程设计名称	计算机网络课程设计	学 院	人工智能学院	指导教师	张雪松
班 级	班内序号	学 号	学生姓名	成绩	
2020219108	03	2020212183	杨再俨		
2020219108	02	2020212181	王焕捷		
2020219108	13	2020212259	何奕骁		
课 程 设 计 内 容	<p>一、课程设计教学目的：将计算机网络课程中所学的DNS相关知识点运用到实际生活中，通过动手实践完成了一个具备基本功能的DNS服务器，在这个过程中加强小组成员对该协议的了解，锻炼了小组成员之间的沟通和交流能力，同时巩固了我们的独立思考和动手能力。</p>				
	<p>二、基本内容：设计一个 DNS 服务器程序，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：</p> <p>(1)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）</p> <p>(2)普通 IP 地址，则向客户端返回该地址（服务器功能）</p> <p>(3)表中未检测到该域名，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）</p>				
	<p>三、高级功能：我们加入了 cache 缓存机制，配合LRU换出算法与清除长时未访问的cache数据的算法，以及字典树查询算法，提高了查询效率；我们通过条件编译的方法，保证了我们的DNS服务器程序能够Windows/Linux 跨平台，并且具有一致的性能。</p>				
	<p>四、实验方法：首先通过查询资料和阅读书籍完整掌握DNS服务器的相关知识，接着对整体框架进行设计，进而对主体框架所用到的其他函数接口进行详细设计。本实验使用C语言完成DNS服务器的设计。</p>				

一、系统功能设计

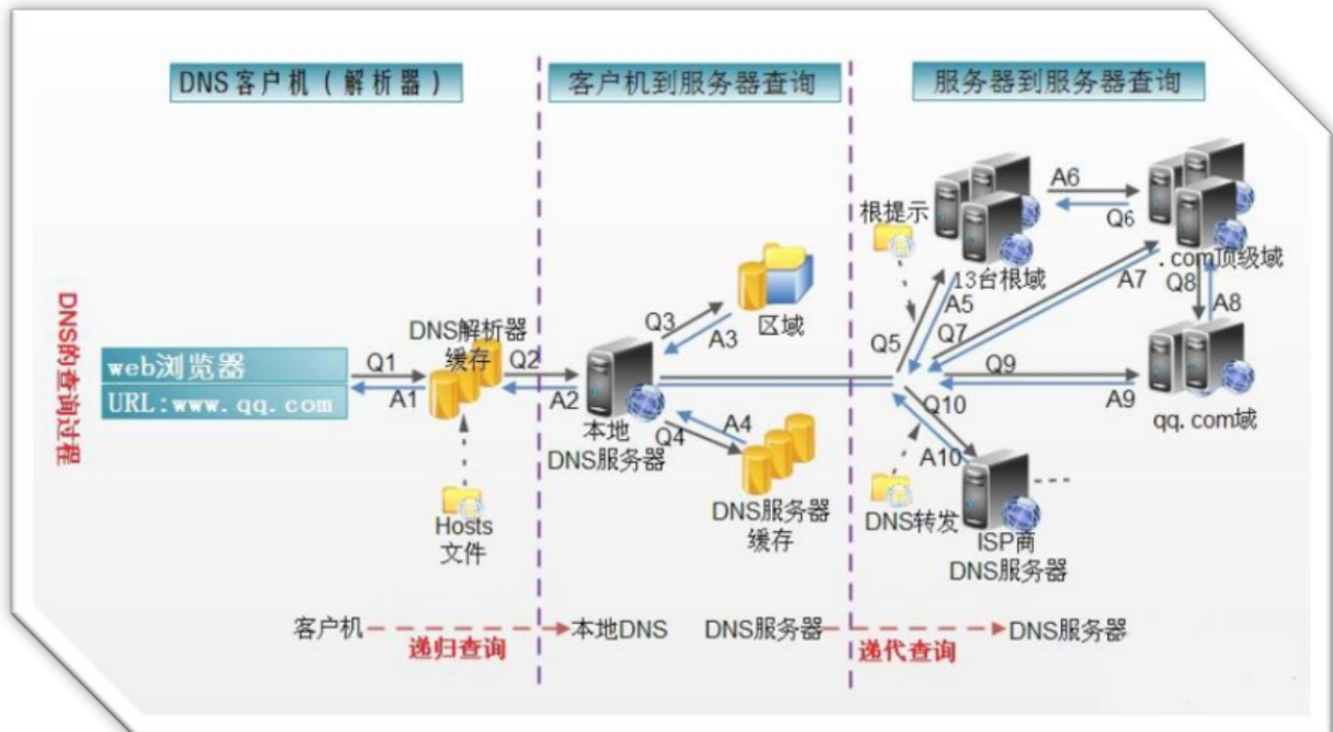
1.设计任务

设计一个 DNS 服务器程序，读入“域名-IP 地址”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，实现下列三种情况：

- 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)；
- 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)；
- 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)同时程序可支持多客户端同时查询的情况。

2.背景简述——DNS（域名系统）

域名系统（英文：Domain Name System，缩写：DNS）是互联网的一项服务。它作为将域名和 IP 地址相互映射的一个分布式数据库，能够使人更方便地访问互联网。DNS 在传输层主要使用UDP 端口 53，其采用 Client-Server 模式，具有管理域名、域名-IP地址转换等功能。当前，对于每一级域名长度的限制是 63 个字符，域名总长度则不能超过 253 个字符。DNS 大致结构与功能演示如下图所示：



3.实验环境

- i. 操作系统： Windows10 / Windows11/Linux CentOS7
- ii. 编程语言： C
- iii. 开发环境： Visual Studio 2019 集成开发环境 与 CodeBlocks集成开发环境

4.实验准备

- i. 查阅并学习了 socket 编程的详细过程与注意事项
- ii. 结合 RFC1035 文档对 DNS 报文进行了详细的分析理解
- iii. 通过shell窗口使用 nslookup 命令进行DNS解析调试

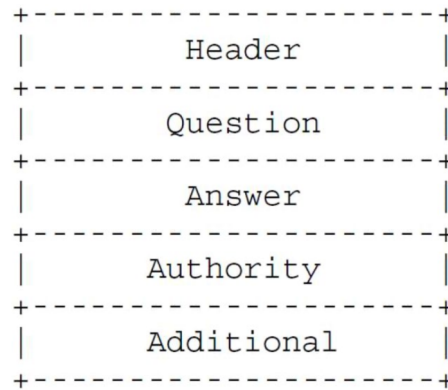
5.功能概述

本次课程设计， 我们的小组完成了 DNS 中继服务器的设计与实现。我们设计的 DNS 中继服务器采用 socket 编程， 首先满足了计算机网络课程设计报告要求之中对于基础功能的要求， 即本中继服务器支持读取本地的域名-IP 地址对照表， 并当用户输入要查询的域名后对输入的域名进行查询。当在本地文件信息或缓存中查到了该域名的 IP 地址时， 若 IP 地址为 0.0.0.0， 向客户端发送“域名不存在”的报错 DNS 报文， 实现了不良网站的拦截功能； 若 IP 地址为普通地址， 则直接向客户端发回， 实现了 DNS 服务器功能。而当本地文件中无法检索到对应域名信息时， 中继服务器将选择向外部的实际本地 DNS 服务器转发请求报文， 得到反馈后在加入自己缓存队列的同时， 也向客户端进行转发操作， 实现了 DNS 中继功能。

6.DNS 包结构的详细分析

6.1DNS的报文

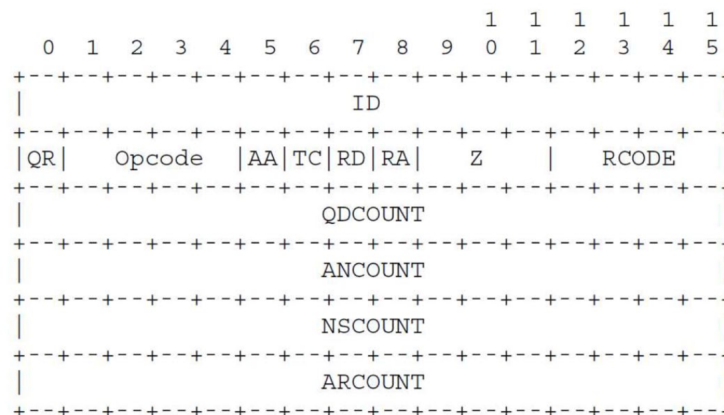
DNS 的报文构成如下图所示：



报文由报头 header 部分和数据 data 部分组成；数据字段分别为 Question：向域名服务器提出的问题个数；Answer：域名服务器回答问题的资源记录 RR 个数；Authority：指向权威服务器回答的 RR 数；Additional：附加信息答案的 RR 数。data 字段的每个子段都是由 0~n 个资源记录组成。

6.2DNS报文的报头header部分

DNS 的报头构成如下图所示：



DNS 报头共 12 字节，其不同字段的含义如下：

ID：程序分配的 16 位标识符，由客户程序设置并由服务器返回结果。客户程序通过它来确定响应与查询请求是否匹配。

QR：1 位字段，0 表示查询请求报文，1 表示响应报文。

OPCODE：4 位字段，通常为 0(标准查询)，其他值为 1(反向查询)和 2(服务器状态请求)，3——15 供将来使用。

AA：权威答案(Authoritative answer)标志

TC：被截断的报文(Truncated)标志，当响应的总长度超 512 字节时，只返回前 512 个字节。

RD：期望使用递归解析 (Recursion desired)标志，查询报文中计算机网络 课程设计报告设置，响应报文中返回，同时引导名字服务器采用递归查询方式。如果该位为 0，表示使用迭代查询方式。

RA：递归可用(Recursion Available)标志，如果名字服务器支持递归查询，则在响应中该比特置为

Z: 必须为 0, 保留将来使用RCODE: 响应码(Response coded), 仅用于响应报文: 值为 0: 没有差错条件值为 3: 名称差错。仅对来自权威名字服务器响应有意义, 预示该请求查询的域名不存在。

QDCOUNT: question section 的问题个数

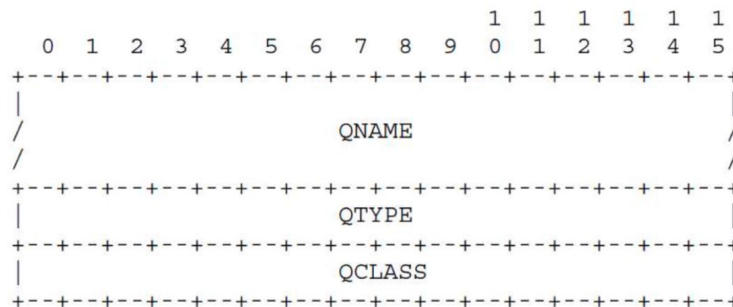
ANCOUNT: answer section 的 RR 个数

NSCOUNT: authority records section 的 RR 个数

ARCOUNT: additional records section 的 RR 个数

6.3 DNS 报文的问题请求段格式

DNS 报文的问题请求段格式如下图所示:

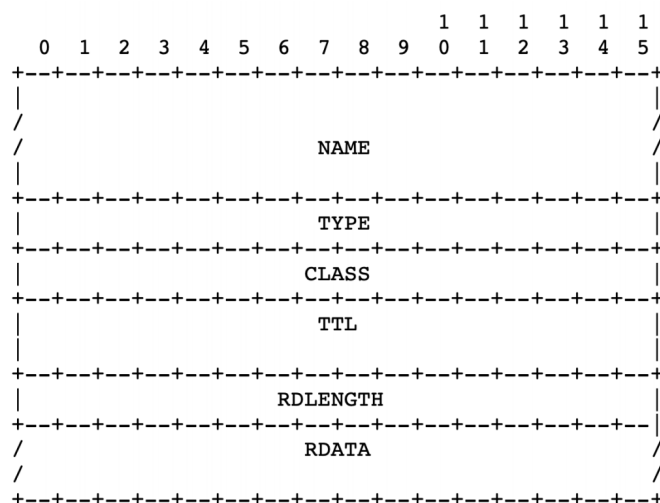


我们仔细阅读了 RFC1035 文档, 了解到 QNAME 字段以标签形式存储了要查询的域名名称。例如客户端查询 www.baidu.com 这一域名对应的 IPV4 地址时, DNS 包将该字符串按照符号“.”划分成了三个标签: www、baidu、com。在具体的封包中, 每个标签存进去的第一个字节并非其内容, 而是这个标签的字符数长

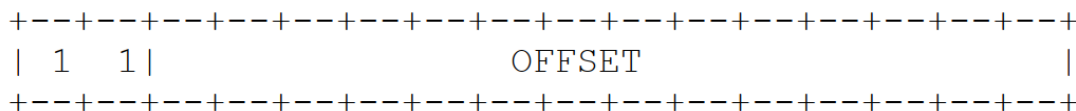
度。例如 baidu 标签转换为 DNS 报文支持的字节存储方式, 其第一个字节为 0x05, 表示该标签共长五字节, 后续五个字节紧跟着 baidu 五个字符的 ASCII 码。为了区分 QNAME 和 QTYPE 的边界, 当域名按照标签存储完毕后, 会填充一个字节的 0, 即 0x00 于最后, 标识域名内容存储结束。QTYPE 表示查询类型, 如 A (1), MX (15), AAAA (28) 等。QCLASS 在因特网中固定为 1, 表示“IN”。

6.4 DNS 报文的资源记录格式

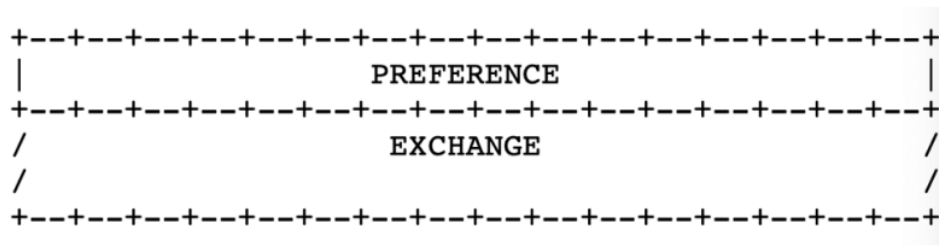
DNS 报文资源记录格式如下图所示:



同样的，我们依旧查询文档得知NAME 字段有非压缩方式和压缩方式两种存储方式实现。其中非压缩方式的结构规则和 QuestionSection 中的 QNAME 字段相同。而压缩方式的出现是考虑到较多情况下 NAME 字段和 QNAME 字段内容完全相同，因此无需再次存储详细信息。故该 NAME 字段改动为如下所示：



该字段前两位表示这是用压缩方式存储的 NAME，后 14 位为偏移量，类似一个相对指针，其表示这里的 NAME 在 DNS 报文中第一次出现时，距离 DNS 报文报头的字节偏移数。对于一问一答的类型，该偏移量为 0x0C，即报头的 12 字节偏移。采用压缩方式存储，可以使资源记录格式成为定长数据字段。后续的 TYPE、CLASS 字段与 QTYPE、QCLASS 含义相同，TTL 表示本答案的生存周期。RDLENGTH 表示资源数据的字节数，如 A 类型一定为 4 字节（32 位），而 RDATA 为具体的 RR 值。特别的，对于 MX 类型数据，其 RR 部分如下格式：



PREFERENCE 表示本答案的优先级，为 16 位整数，EXCHANGE 为具体的资源内容。

二、模块划分及软件流程图

1.模块划分

1.1 初始化模块

该模块主要功能为初始化DNS服务器的相关配置，如读取配置文件dnsrelay.txt、初始化socket套接字等。

初始化模块

复制代码

```
1 void initTable(); // 初始化dns_table表（包括cache）
2 void initialize_id_table() // 初始化id表
3 void initialize_socket(); // 初始化套接字
4 void handle_arguments(int argc, char* argv[]); // 处理用户shell输入
```

1.2 域名-IP表存储和查找模块

该模块解决如何在内存(table表)及Cache中保存url-ip映射，以及通过哪些方式对这些数据结构进行处理。

1.2.1 table表与cache的基本结构

table与cache采用的是循环双链表进行储存的，链表采用的是Linux内核中的list.h，内存与cache共用一个结点，含注释的源代码如下所示：

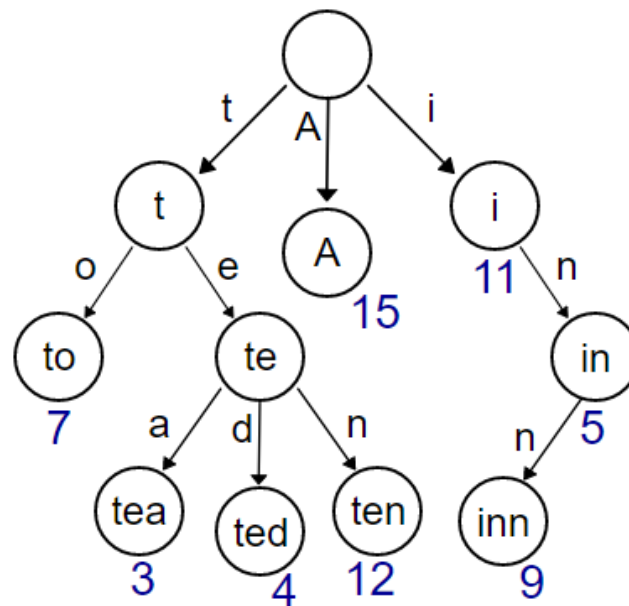

```

1 // 链表的结点
2 ▾ typedef struct Node {
3     // list.h封装好的循环双链表节点结构体
4     struct list_head list;
5     char url[200]; // url
6     char ip[50]; // ip地址
7     // 该结点的预计存活时间，到该时间将删除该结点并同时删除相应字典树中对应的结点
8     // 内存(table)与cache中的结点能存活的时间不同，cache结点的存活时间远小于table
    的
9     time_t expireTime;
10 } Node;

```

1.2.2 Trie字典树

Trie（字典树）来实现域名到对应IP和TTL记录的映射。它的基本思想是构造一棵树，从根结点出发，依次以字符串中的字符为下标在树的结点进行转移，直到字符串读完，停在的结点即为字符串所对应的索引。我们可以将索引指向Cache或table中的记录。字典树的结构如下图所示：



我们为table与cache分别创建了对应的字典树，他们的字典树结构相同，如下：

字典树结点

复制代码

```
1  typedef struct TrieNode
2  {
3      union
4      {
5          struct Node* lf; // 该Trie结点指向的链表对应结点
6          struct TrieNode* ptr[BRANCH]; // BRANCH==192 孩子结点数组
7      };
8  } TrieNode;
```

字典树的方法如下：

字典树的方法

复制代码

```
1  // 向字典树中增加结点，type为0是往table字典树中添加；为1是往cache字典树中添加
2  void addDataToTrie(char* url, int type, struct Node* lf);
3  // 在字典树中查询 type为0是在table字典树中查询；为1是在cache字典树中查询
4  // 返回查询到的对应链表中的结点（若未查到，返回NULL）
5  struct Node* findNodeInTrie(char* url, int type);
6  // 递归的销毁字典树
7  void delTrie(TrieNode* p);
8  // 在字典树中删除结点 type为0是在table字典树中删除；为1是在cache字典树中删除
9  void delNodeInTrie(char* url, int type);
```

1.2.3Cache

cache的基本结构如1.2.1中所言；我们的cache具有如下功能：

- 调用cache字典树查询cache中有无对应数据，若有，则将数据返回给查询者；
- 使用LRU算法，在cache容量达到所设置上限时能够根据LRU算法删除掉较长时间未访问的数据对应结点（同时删除其对应的cache字典树结点）；
- 设置ttl，若有结点超过其能生存的最长时间，便将其删除。

cache的方法如下：

▼ cache的方法

📄 复制代码

```
1 // 向cache中加入数据
2 void addDataToCache(char* url, char* ip);
3 // 在cache中根据URL获取数据, 若返回0表示cache中无相应数据, 应查询table
4 int getDataInCache(char* url, char** ip);
5 // 删除cache
6 void deleteCache();
```

1.2.3table

table的基本结构如1.2.1中所言；我们的table具有如下功能：

- 启动DNS服务器时，读取txt文件中的url-ip映射数据，并将这些数据加入到table中。这些数据的expireTime特殊的设为0，意为这些数据为静态数据，不能被删除；
- 向外部暴露在table与cache中获取数据的方法，先在cache的字典树中寻找，若cache的字典树中有数据则将结果返回；若cache中没有而table中有，则将数据更新到cache中并返回结果；否则，返回特殊结果，以提示table与cache中无所需数据；
- 除存有静态的txt中读入的数据外，还将存有每次从外部服务器得到的数据，并调用cache的方法将这些数据存储到cache中（table与cache类似于多级cache的关系）；
- 为了防止table占用内存过大或者域名ip变动，table也具有一个ttl。一定时间后table会进行刷新操作，将除了txt中读入的数据以外，动态写入数据中超时的数据所对应的结点以及相应字典树结点删除，以释放内存。

table的方法如下：

▼ table的方法

📄 复制代码

```
1 // 初始化“域名-IP 地址”对照表table以及cache
2 void initTable();
3 // 向table中加入数据 type为1时, 将加入的数据同步到cache中, 为2时, 意为加入txt中的数据
4 void addDataToTable(char* url, char* ip, int type);
5 // 在table与cache获取数据, 若返回0表示DNS中继服务器中无相应数据,
6 // 应向因特网DNS服务器发出查询, 返回2表示在cache中查到
7 int getData(char* url, char** ip);
8 // 清除table中长时间未访问的数据以释放内存并防止域名变动
9 void flashTable();
10 // 删除table
11 void deleteTable();
```

1.3 信息处理模块

该模块的功能是接受并处理来自本地DNS客户端以及外部DNS服务器的报文。实现DNS服务器的查询功能，对不良网站拦截功能以及中继功能。

信息处理模块

🔒 | 📄 复制代码

```
1 // 接收外部服务器报文并进行处理
2 void handle_server_message(char *buf,int length, struct sockaddr_in
  server_addr);
3 // 接收本地客户端报文并进行处理
4 void handle_client_message(char *buf,int length, struct sockaddr_in
  client_addr);
```

上述两个函数为信息处理的入口，分别处理来自本地客户端的报文和来自外部服务器的报文。

信息处理模块

🔒 | 📄 复制代码

```
1 // 将请求报文中的字符串格式的url转换成普通模式的url
2 void url_transform(char* buf, char* result);
3 // 将客户端id转换为服务器id, 进而实现中继功能
4 unsigned short id_transform(unsigned short ID, struct sockaddr_in
  client_addr,
5 int finished);
6 void print_debug_time(); // 输出debug时间
```

1.4 Windows/Linux跨平台处理模块

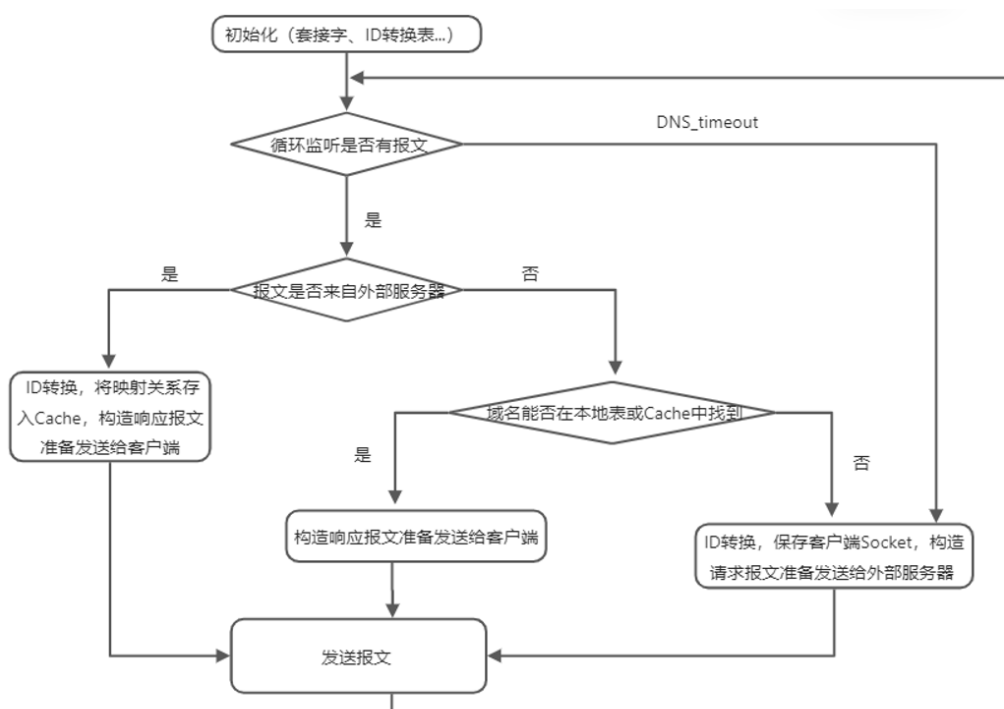
使用条件编译将Windows和Linux的头文件、宏定义、函数等统一名字。在public.h中：

```
1 // 若为windows系统
2 #ifdef _WIN32
3 #include <winsock2.h>
4 #pragma comment(lib, "Ws2_32.lib")
5 WSADATA wsaData;
6 // 若为Linux系统
7 #elif __linux__
8 #include <unistd.h>
9 #include <sys/types.h>
10 #include <sys/socket.h>
11 #include <netinet/in.h>
12 #include <netinet/ip.h>
13 #include <arpa/inet.h>
14 #define closesocket(socketfd) close(socketfd)
15 #endif
```

如果在Windows平台上编译，将保留第一个条件分支；如果在Linux平台上编译，将保留第二个条件分支。

另外，Windows系统与Linux系统的socket库中，一些结构体的简名不同，需要注意。

2.软件流程图



三、 测试用例及运行结果

1.测试步骤

(1)首先在shell窗口使用 ipconfig/all 命令查看当前PC使用的因特网DNS服务器的IP地址(如图为 192.168.3.1)

```
无线局域网适配器 WLAN:
    连接特定的 DNS 后缀 . . . . . : 
    描述 . . . . . : Realtek RTL8852AE WiFi 6 802.11ax PCIe Adapter
    物理地址. . . . . : E0-0A-F6-68-DC-76
    DHCP 已启用 . . . . . : 是
    自动配置已启用. . . . . : 是
    本地链接 IPv6 地址. . . . . : fe80::bcce:aa32:cf1b:e4f9%5(首选)
    IPv4 地址 . . . . . : 192.168.3.238(首选)
    子网掩码 . . . . . : 255.255.255.0
    获得租约的时间 . . . . . : 2022年5月18日 17:16:28
    租约过期的时间 . . . . . : 2022年5月22日 14:09:18
    默认网关 . . . . . : 192.168.3.1
    DHCP 服务器 . . . . . : 192.168.3.1
    DHCPv6 IAID . . . . . : 65014518
    DHCPv6 客户端 DUID . . . . . : 00-01-00-01-29-C4-D2-5B-88-A4-C2-ED-4D-4D
    DNS 服务器 . . . . . : 192.168.3.1
    TCPIP 上的 NetBIOS . . . . . : 已启用
```

(2)进入“网络和Internet设置” 修改当前PC的默认因特网DNS服务器的IP地址，改为自己编写的DNS服务器IP地址 127.0.0.1（本地回环地址）



(3)编译链接源码得到.exe文件，在该可执行文件的目录下打开shell终端，输入 `XXX.exe [-d | -d d] [dns-server-ipaddr] [filename]` 启动DNS服务器

@[dns-server-ipaddr]参数可填入上述查询的IP地址192.168.3.1或网上找可以使用的因特网DNS服务器的IP地址，这一步是为了能够让自己编写的服务器能在查询不到域名时求助因特网DNS服务器

@[-d | -dd]参数表示输出详细或简略运行日志，若无此参数表示不输出任何运行日志

@[filename]参数就是自定义的URL-IP映射表(包括后缀名)

(4)接着我们就可以进行调试，这里有两种方法

一种是新建一个shell窗口使用 nslookup 命令进行DNS解析调试

一种是使用Wireshark工具进行DNS抓包调试

注意：

- 1.调试结束后记得修改DNS为原来的因特网DNS服务器，否则PC没法上网了
- 2.Windows自带一些访问互联网的行为也会触发请求查询行为

其他命令：

ipconfig/displaydns: 查看当前 dns cache 的内容以确认程序执行结果的正确性（这个我还没玩明白，不知道是从服务器的shell看还是客户端的shell看，有没有可能是必须要在服务器运行的状态下才能看到）

ipconfig/flushdns: 清除 dns cache 中缓存的所有 DNS 记录

下图是成功启动DNS服务器的界面

```

PS D:\My_document\大二下文档\课程pdf\计算机网络\计网课设\自建\DNS(最终版)> .\DNS测试 -dd 192.168.3.1 dnsrelay.txt
指定DNS服务器的IP地址: 192.168.3.1
debug=2
初始化“域名-IP 地址”对照表完成。
加载配置文件成功,结果如下:
<域名: 008.cn, IP地址 : 0.0.0.0>
<域名: 2qq.cn, IP地址 : 0.0.0.0>
<域名: 555.265.com, IP地址 : 0.0.0.0>
<域名: abc.265.com, IP地址 : 0.0.0.0>
<域名: abcdesign.ru, IP地址 : 0.0.0.0>
<域名: ad.qingyule.com, IP地址 : 0.0.0.0>
<域名: alexey.pioneers.com.ru, IP地址 : 0.0.0.0>
<域名: baltnet.ru, IP地址 : 0.0.0.0>
<域名: cctv1.net, IP地址 : 0.0.0.0>
<域名: cctv8.net, IP地址 : 0.0.0.0>
<域名: chinabdkx.363.net, IP地址 : 0.0.0.0>
<域名: ciachoo.pl, IP地址 : 0.0.0.0>
<域名: clients.babylon.co.il, IP地址 : 0.0.0.0>
<域名: dicto.ru, IP地址 : 0.0.0.0>
<域名: elemental.ru, IP地址 : 0.0.0.0>
<域名: errorguard.com, IP地址 : 0.0.0.0>
<域名: financial.washingtonpost.com, IP地址 : 0.0.0.0>
<域名: free.bestialityhost.com, IP地址 : 0.0.0.0>
<域名: friendlygreeting.com, IP地址 : 0.0.0.0>
<域名: gamma.vyborg.ru, IP地址 : 0.0.0.0>

```

2.测试中继功能

在配置文件中不存在www.baidu.com对应的IP地址，故服务器进行中继并将从外部收到的响应报文转发给客户端。

客户端：

```

C:\Users\a7386>nslookup www.baidu.com
服务器: UnKnown
Address: 127.0.0.1

非权威应答:
名称: www.a.shifen.com
Addresses: 39.156.66.14
           39.156.66.18
Aliases: www.baidu.com

```

服务器端：


```

本地时间:2022/05/21 17:14:16 Saturday
#5:收到来自本地客户端的请求
完整报文内容如下:
00 02 01 00 00 01 00 00 00 00 00 03 77 77 77
05 62 61 69 64 75 03 63 6f 6d 00 00 01 00 01
报文总长度为31
#5:本地服务器查询<URL:www.baidu.com>失败, 即将向外部服务器发送请求
#5:成功向外部服务器发送请求,需要查询的域名为<URL:www.baidu.com>

本地时间:2022/05/21 17:14:16 Saturday
#5:收到来自外部服务器的响应
完整报文内容如下:
01 00 81 80 00 01 00 03 00 00 00 03 77 77 77
05 62 61 69 64 75 03 63 6f 6d 00 00 01 00 01 c0
0c 00 05 00 01 00 00 02 c2 00 0f 03 77 77 77 01
61 06 73 68 69 66 65 6e c0 16 c0 2b 00 01 00 01
00 00 00 a5 00 04 27 9c 42 0e c0 2b 00 01 00 01
00 00 00 a5 00 04 27 9c 42 12
报文总长度为90
#5:外部DNS服务器查询<URL:www.baidu.com>成功, 解析结果如下:
#5:收到的响应报文属性 TYPE: 5, CLASS: 1, TTL: 706
#5:收到的响应报文属性 TYPE: 1, CLASS: 1, TTL: 165
#5:IPV4: 39.156.66.14
#5:成功向本地客户端转发响应

```

3.测试不良网站屏蔽功能

在配置文件中我们将008.cn域名对应的IP地址屏蔽, 故尽管能在本地DNS服务器中查询到该域名对应的IP, 但是在构造响应报文进行响应时会进行屏蔽处理。

客户端:

```

C:\Users\a7386>nslookup 008.cn
服务器: UnKnown
Address: 127.0.0.1

*** UnKnown 找不到 008.cn: Non-existent domain

```

服务器端:

```

本地时间:2022/05/21 18:58:12 Saturday
#18:收到来自本地客户端的请求
完整报文内容如下:
00 02 01 00 00 01 00 00 00 00 00 03 30 30 38
02 63 6e 00 00 01 00 01
报文总长度为24
#18:从内存中查到<URL:008.cn>对应的IP, 结果如下:
<URL: 008.cn , IP: 0.0.0.0>
#18:构造并发送响应报文
#18:成功向本地客户端发送响应

```

4.测试服务器功能

在配置文件中添加了sohu对应的IP地址为61.135.181.175，当客户端请求该域名对应的IP地址时服务器可以成功进行响应。

客户端：

```
C:\Users\a7386>nslookup sohu
服务器: UnKnown
Address: 127.0.0.1

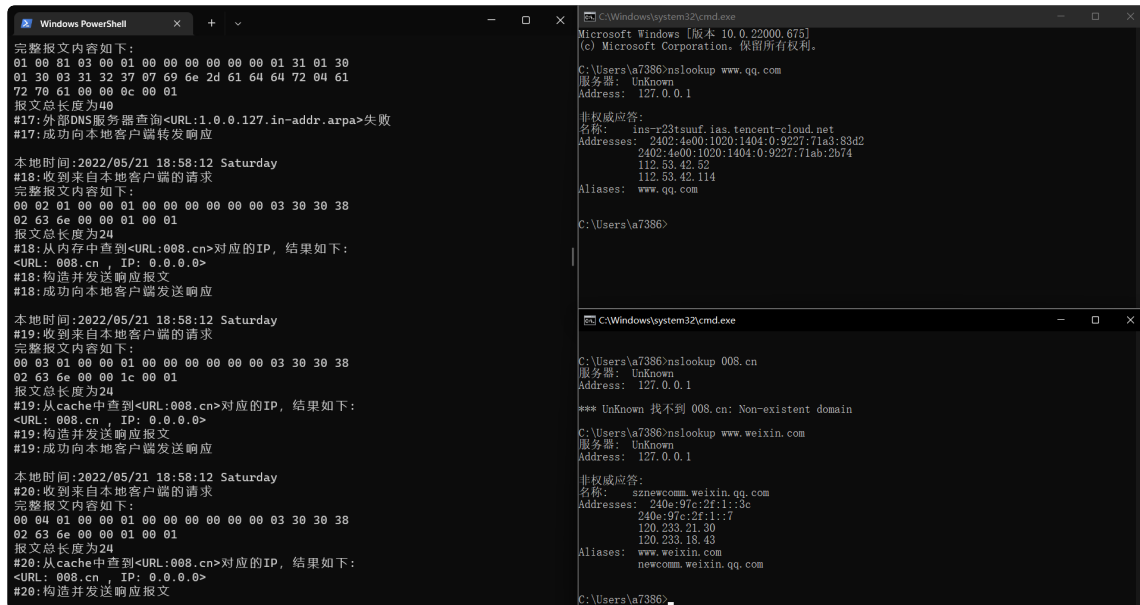
非权威应答:
名称: sohu
Address: 61.135.181.175
```

服务器端：

```
本地时间:2022/05/21 17:23:26 Saturday
#2:收到来自本地客户端的请求
完整报文内容如下:
00 02 01 00 00 01 00 00 00 00 00 00 04 73 6f 68
75 00 00 01 00 01
报文总长度为22
#2:从内存中查到<URL:sohu>对应的IP, 结果如下:
<URL: sohu , IP: 61.135.181.175>
#2:构造并发送响应报文
#2:成功向本地客户端发送响应
```

5.多用户测试

当多个客户端向服务器发送DNS请求时因为UDP协议自带的并发特性，支持多用户的操作。



```
Windows PowerShell
完整报文内容如下:
01 00 01 03 00 01 00 00 00 00 00 01 31 01 30
01 30 03 31 32 37 07 69 6e 2d 61 64 72 04 61
72 70 61 00 00 0c 00 01
报文总长度为40
#17:外部DNS服务器查询<URL:1.0.0.127.in-addr.arpa>失败
#17:成功向本地客户端转发响应

本地时间:2022/05/21 18:58:12 Saturday
#18:收到来自本地客户端的请求
完整报文内容如下:
00 02 01 00 00 01 00 00 00 00 00 00 03 30 30 38
02 63 6e 00 00 01 00 01
报文总长度为24
#18:从内存中查到<URL:008.cn>对应的IP, 结果如下:
<URL: 008.cn , IP: 0.0.0.0>
#18:构造并发送响应报文
#18:成功向本地客户端发送响应

本地时间:2022/05/21 18:58:12 Saturday
#19:收到来自本地客户端的请求
完整报文内容如下:
00 03 01 00 00 01 00 00 00 00 00 00 03 30 30 38
02 63 6e 00 00 1c 00 01
报文总长度为24
#19:从cache中查到<URL:008.cn>对应的IP, 结果如下:
<URL: 008.cn , IP: 0.0.0.0>
#19:构造并发送响应报文
#19:成功向本地客户端发送响应

本地时间:2022/05/21 18:58:12 Saturday
#20:收到来自本地客户端的请求
完整报文内容如下:
00 04 01 00 00 01 00 00 00 00 00 00 03 30 30 38
02 63 6e 00 00 01 00 01
报文总长度为24
#20:从cache中查到<URL:008.cn>对应的IP, 结果如下:
<URL: 008.cn , IP: 0.0.0.0>
#20:构造并发送响应报文

C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.22000.675]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\a7386>nslookup www.qq.com
服务器: UnKnown
Address: 127.0.0.1

非权威应答:
名称: ins-r23tsunf.ias.tencent-cloud.net
Addresses: 2402:4e00:1020:1404:0:9227:71a3:8342
2402:4e00:1020:1404:0:9227:71ab:2b74
112.53.42.52
112.53.42.114
Aliases: www.qq.com

C:\Users\a7386>

C:\Windows\system32\cmd.exe
C:\Users\a7386>nslookup 008.cn
服务器: UnKnown
Address: 127.0.0.1

*** UnKnown 找不到 008.cn: Non-existent domain

C:\Users\a7386>nslookup www.weixin.com
服务器: UnKnown
Address: 127.0.0.1

非权威应答:
名称: sznewcomm.weixin.qq.com
Addresses: 240e:97c:2f:1:1:3c
240e:97c:2f:1:1:7
120.233.21.30
120.233.18.43
Aliases: www.weixin.com
newcomm.weixin.qq.com

C:\Users\a7386>
```

6.Cache缓存测试

经过服务器转发过的url-ip会被保存在Cache中，在有效期内再次查询该url时会直接从Cache中查询得到结果。

客户端：

```
C:\Users\a7386>nslookup www.baidu.com
服务器: UnKnown
Address: 127.0.0.1

非权威应答:
名称: www.a.shifen.com
Addresses: 39.156.66.18
          39.156.66.14
Aliases: www.baidu.com

C:\Users\a7386>nslookup www.baidu.com
服务器: UnKnown
Address: 127.0.0.1

非权威应答:
名称: www.baidu.com
Address: 39.156.66.18
```

服务器端：

```
本地时间:2022/05/21 18:57:02 Saturday
#12:收到来自本地客户端的请求
完整报文内容如下:
00 02 01 00 00 01 00 00 00 00 00 00 03 77 77 77
05 62 61 69 64 75 03 63 6f 6d 00 00 01 00 01
报文总长度为31
#12:从cache中查到<URL:www.baidu.com>对应的IP, 结果如下:
<URL: www.baidu.com , IP: 39.156.66.18>
#12:构造并发送响应报文
#12:成功向本地客户端发送响应
```

7.Linux环境测试

使用的是阿里云服务器，操作系统为centos7.0。

先使用gcc编译好源文件，输出的二进制格式文件为main.out文件。

然后更改根目录的etc文件夹的resolv.conf文件，如下所示，将nameserver全部改为127.0.0.1（提前备份好），如下图所示：

```
resolv.conf x
1 options timeout:2 attempts:3 rotate single-request-reopen
2 ; generated by /usr/sbin/dhclient-script
3 nameserver 127.0.0.1
4 nameserver 127.0.0.1
5
```

保存该文件，在main.out目录下，终端输入 `./XXX.out [-d | -dd] [dns-server-ipaddr] [filename]` 启动DNS服务器

@[dns-server-ipaddr]参数可填入上述查询的IP地址192.168.3.1或网上找可以使用的因特网DNS服务器的IP地址，这一步是为了能够让自己编写的服务器能在查询不到域名时求助因特网DNS服务器

@[-d | -dd]参数表示输出详细或简略运行日志，若无此参数表示不输出任何运行日志

@[filename]参数就是自定义的URL-IP映射表(包括后缀名)

如下所示(一定要输入服务器可以使用的因特网DNS服务器):

```
宝塔终端

Last login: Sat May 21 20:59:55 2022 from 127.0.0.1

Welcome to Alibaba Cloud Elastic Compute Service !

[root@iz2zeae73xmepqyovtaalz ~]# cd /home/code/DNS
[root@iz2zeae73xmepqyovtaalz DNS]# ./main.out -dd 100.100.2.138
```

回车，服务器正常启动：

```
<域名: xyxy68.8u8.net, IP地址 : 0.0.0.0>
<域名: youlove.3322.net, IP地址 : 0.0.0.0>
<域名: zbszx.vicp.net, IP地址 : 0.0.0.0>
<域名: zelnet.ru, IP地址 : 0.0.0.0>
<域名: test0, IP地址 : 0.0.0.0>
<域名: test1, IP地址 : 11.111.11.111>
<域名: test2, IP地址 : 22.22.222.222>
<域名: sina, IP地址 : 202.108.33.89>
<域名: sohu, IP地址 : 61.135.181.175>
<域名: bupt, IP地址 : 123.127.134.10>
<域名: ad4.sina.com.cn, IP地址 : 0.0.0.0>
<域名: www.163daohang.com.cn, IP地址 : 0.0.0.0>
socket连接建立成功!
socket端口绑定成功!
```

打开一个新的终端，输入指令 `ping www.baidu.com`，结果如下：

```
[root@iz2zeae73xmepqyovtaalz ~]# ping www.baidu.com
PING www.a.shifen.com (110.242.68.4) 56(84) bytes of data.
64 bytes from 110.242.68.4 (110.242.68.4): icmp_seq=1 ttl=50 time=15.2 ms
64 bytes from 110.242.68.4 (110.242.68.4): icmp_seq=2 ttl=50 time=15.2 ms
64 bytes from 110.242.68.4 (110.242.68.4): icmp_seq=3 ttl=50 time=15.2 ms
64 bytes from 110.242.68.4 (110.242.68.4): icmp_seq=4 ttl=50 time=15.2 ms
64 bytes from 110.242.68.4 (110.242.68.4): icmp_seq=5 ttl=50 time=15.2 ms
64 bytes from 110.242.68.4 (110.242.68.4): icmp_seq=6 ttl=50 time=15.2 ms
64 bytes from 110.242.68.4 (110.242.68.4): icmp_seq=7 ttl=50 time=15.2 ms
^C
```

查看我们的DNS服务器终端，有：

```

本地时间:2022/05/21 21:05:40 Saturday
#27:收到来自本地客户端的请求
完整报文内容如下:
9a 1d 01 00 00 01 00 00 00 00 03 77 77 77
05 62 61 69 64 75 03 63 6f 6d 00 00 01 00 01
报文总长度为31
#27:本地服务器查询<URL:www.baidu.com>失败,即将向外部服务器发送请求
#27:成功向外部服务器发送请求,需要查询的域名为<URL:www.baidu.com>

本地时间:2022/05/21 21:05:40 Saturday
#27:收到来自外部服务器的响应
完整报文内容如下:
01 00 81 80 00 01 00 03 00 00 00 00 03 77 77 77
05 62 61 69 64 75 03 63 6f 6d 00 00 01 00 01 c0
0c 00 05 00 01 00 00 04 ad 00 12 03 77 77 77 01
61 06 73 68 69 66 65 6e 03 63 6f 6d 00 c0 2b 00
01 00 01 00 00 00 98 00 04 6e f2 44 04 c0 2b 00
01 00 01 00 00 00 98 00 04 6e f2 44 03
报文总长度为93
#27:外部DNS服务器查询<URL:www.baidu.com>成功,解析结果如下:
#27:收到的响应报文属性 TYPE: 5, CLASS: 1, TTL: 1197
#27:收到的响应报文属性 TYPE: 1, CLASS: 1, TTL: 152
#27:IPV4: 110.242.68.4
#27:成功向本地客户端转发响应

```

在终端中输入指令ping 008.cn(如上在Windows中测试时一样,已将该网站屏蔽),如下:

```

100 min/avg/max/mdev ~ 15.250/15.255/15.264/0.133 ms
[root@iz2zeae73xjmepqyovtaalz ~]# ping 008.cn
ping: 008.cn: Name or service not known
[root@iz2zeae73xjmepqyovtaalz ~]# 

```

可见,我们实现了不良网站屏蔽功能。

经过我们的测试,在Linux系统上所有功能与Windows上的一致,由于篇幅受限,不再一一赘述。

四、调试中遇到并解决的问题

在初始调试的过程中我们发现服务器后台经常会出现一些看不懂的域名请求,后来经过研究发现就算我们不使用命令进行调试,由于操作系统或者其他后台应用程序也会做出一些需要域名请求的行为。这样其实会导致原本理想中的日志信息和真实的日志信息出现一些不同,因此我们在输出调试信息时增加了序号,这样可以较为方便的查询到本地客户端和外部服务器的对应关系。

同时还有对于IPV6的解决,因为我们涉及的服务器只能处理IPV4的请求,而有些域名会同时对应IPV4和IPV6,所以我们增加了对IPV6的处理方式即通过判断其TYPE字段的类型确定它是合法的url且是IPV6类型的请求后将该请求中继转发给外部服务器。

五、程序改进思路和展望

5.1程序改进

在处理细节方面，尽管程序可以实现上述基本的DNS服务器的功能，但是针对某些细节如处理IPV6请求、优化存储结构等还可以做进一步的改进。并且针对输出日志的功能，还是无法达到像WireShark一样的详细和专业。所以我们的DNS程序可以顺着这个思路进行改进。

在程序功能方面，为了防止像DNS域名劫持等情况，我们的DNS程序可以改进为DoH（DNS over HTTPS）。利用安全的HTTPS协议运行DNS，主要目的是增强用户的安全性和隐私性。通过使用加密的HTTPS连接，第三方将不再影响或监视解析过程。因此，欺诈者将无法查看请求的URL并对其进行更改。而且如果使用了基于HTTPS的DNS，数据在传输过程中发生丢失时，DoH中的传输控制协议（TCP）会做出更快的反应。

5.2展望

在接下来的时间里我们还将对程序进行优化和调试，争取能够达到和互联网上通用的DNS服务器一样的水准。并实现更多的更好的功能。

六、总结和心得体会

1.项目总结

我们通过查阅书籍、网上资料等掌握了如何搭建一个DNS服务器，实现基本的代理、中继以及屏蔽功能。在这个过程中我们遇到了许多困难，从最开始的一筹莫展到后来针对每个接口细节的处理以及BUG的修改都花了非常多的时间。在最后看到我们的服务器成功的能够运行并且PC可以通过我们的服务器中继进行上网时，我们觉得一切都很值得。

整个实验过程随着我们对课程体系以及相关知识点的深入学习进行，我们在学习的过程中实践，在实践的过程中学习，面对以前从未接触过的socket编程以及DNS报文结构的分析刚开始确实让我们感到惧怕，随着不断的深入学习我们认识到自己的不足并进行了强化，最终圆满实现了整个程序的设计。

2.个人心得体会

杨再俨：在刚开始负责整个程序框架的设计时我非常的苦恼，因为以前从来没有接触过相关的设计，甚至都不知道一个服务器应该做什么，也不知道DNS服务器有什么作用等。经过了长时间的资料的查阅过后，我整理出了一些比较有用的资料（互联网上有非常多鱼龙混杂的资料导致盲目阅读浪费大量时间）与组员进行分享。进行了长时间的理解消化最终确定了项目的整体框架并着手进行代码的编写，期间花了大量时间进行BUG的修改和代码结构的调整等。最终看到我们的程序可以正常运行时我感到一切的努力都是值得的。

王焕捷：在本次计算机网络课程设计中，我们深入理解了DNS服务器的工作原理，对DNS报文有了很深刻的认识，在小组共同学习下，我们实现了本地域名解析、中继、不良网站过滤，多用户使用，Linux环境实现等功能。在小组鼎力合作之下，我们对于计算机网络的相关知识产生了浓厚的兴趣，先后共同学习了RFC1035文档、SOCKET编程以及DNS原理及报文格式。在实验的过程中，我们遇到了各种各样的困难，像SOCKET编程和C语言的指针的应用等等，但是经过了本次的课程设计实验之后，我们对计算机网络课程的理解更加深入了，虽然本学期计算机网络课程采取网课教学，但是学到了很多东西，学习并理解了计算机网络实现的原理。这个项目真的让我学到了很多东西。

何奕骁：由于我之前使用C语言的经验较少，在将算法实现的过程中遇到了很多困难，如指针非法访问等。但是在不断的DeBug过程中我逐渐的熟悉了C语言与指针的用法，也将我面向对象思维设计的“域名-IP地址”对照表与cache使用C语言完美的实现了。在做项目的过程中，我也学会了一些socket编程的基本操作与方法。与以往使用JavaScript、Java等语言的框架搭建服务器不同，通过这个项目，我对计算机网络课上所学知识点有了更深的理解。在与小组成员进行对调接口、沟通交流的过程中我提升了协作开发与人际交往的能力，在DeBug的过程中我也通过自己“踩下的坑”提升了自己编程的能力、养成了较好的编程习惯。总之，通过这个项目我学到了很多东西。