DNS 中继服务器实验报告

开发环境：
版本库：Qt 5.12.2
IDE：Qt Creator 4.8.2
编译器：Desktop Qt 5.12.2 MSCV2017 64bit
系统：Windows 10

一、功能设计

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，三种检索结果：

1) 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（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) 表中未检到该域名，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）

注：考虑多个计算机上的客户端会同时查询，需要进行消息 ID 的转换。

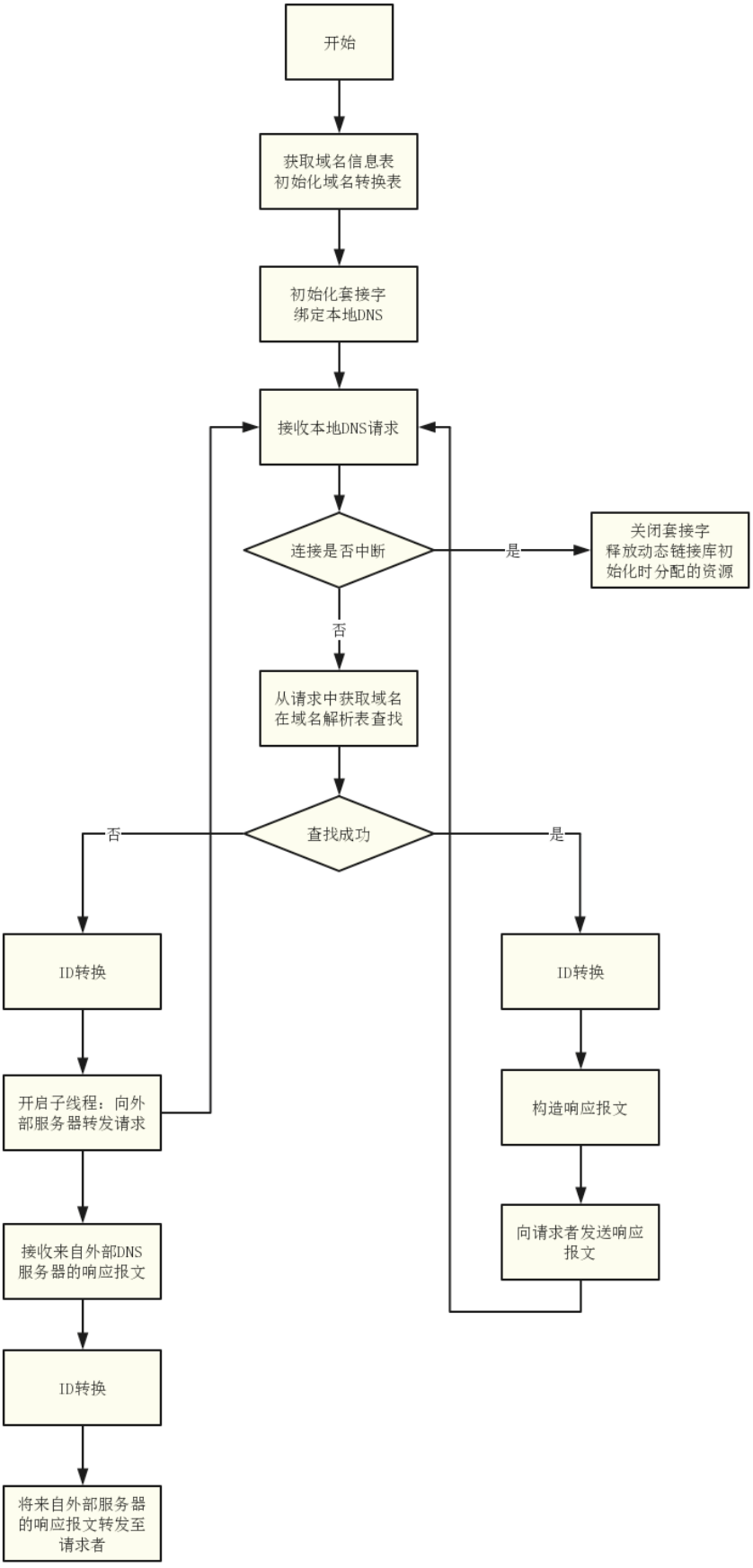
二、模块划分

该系统分为两大模块：本地解析模块和外部解析模块；

1) 本地解析模块：本模块是在该 DNS 服务器本地保存的域名解析表（曾经解析过的的域名和其对应 IP 信息）中查找从应用程序来的请求解析的域名，在这个文件中查到需要的域名后取出对应的 IP 地址，并构造 DNS 应答数据包返回给发送域名解析请求的应用程序。
2) 外部解析模块：若本地解析失败，该 DNS 服务器就调用外部解析模块，此模块将应用程序发送的 DNS 请求报文转发给外部 DNS 服务器，然后接收外部服务器返回的应答信息，并根据这个信息给予应用程序相应的 DNS 应答。每一次外部解析就是一个子线程，用以处理多台计算机同时发送请求的情况。
3) 命令行参数解析模块：解析 cmd 命令行，输出不同级别的调试信息
   如下所示：

```
if(argc==1)
{
    cout <<"【"<<QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss").toUtf8().data()<<"】"<<"<"<<seq++<<
    level=0;
}
else if(argc==4)
{
    if(strcmp(argv[1],"-d")==0)
    {
        cout <<"【"<<QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss").toUtf8().data()<<"】"<<"<"<<s
        strcpy(out_DNS, argv[2]);     //outerdns = 外部dns    ⚠ 'strcpy' is deprecated: This function or vari…
        path=QString(QLatin1String(argv[3]));
        level=1;
    }
    else
    {
        cout <<"【"<<QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss").toUtf8().data()<<"】"<<"<"<<s
        return a.exec();
    }
}
else if (argc==3)
{
    if(strcmp(argv[1],"-dd")==0)
    {
        cout <<"【"<<QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss").toUtf8().data()<<"】"<<"<"<<s
        strcpy(out_DNS, argv[2]);    ⚠ 'strcpy' is deprecated: This function or variable may be unsafe. Cons…
        level=2;
    }
    else
    {
        cout <<"【"<<QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss").toUtf8().data()<<"】"<<"<"<<s
        return a.exec();
    }
}
else
{
    cout <<"【"<<QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss").toUtf8().data()<<"】"<<"<"<<seq++<<
```

三、软件流程图


开始
获取域名信息表
初始化域名转换表
初始化套接字
绑定本地DNS
接收本地DNS请求
连接是否中断
是
关闭套接字
释放动态链接库初始化时分配的资源
否
从请求中获取域名
在域名解析表查找
查找成功
否
是
ID转换
开启子线程：向外部服务器转发请求
接收来自外部DNS服务器的响应报文
ID转换
将来自外部服务器的响应报文转发至请求者
ID转换
构造响应报文
向请求者发送响应报文

四、测试用例以及运行结果

1. 中继功能测试

查找本地 DNS 服务器中不存在的域名，经由中继功能对外部 DNS 发送请求报文，并将得到的应答报文 发送给请求方

请求查询域名 www.baidu.com 的 IP。如图所示，得到了对应的 IP 地址

```
**********收到请求者的包**********
00 03 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 1c 00 01
对应网址：www.baidu.com
在域名解析表中没有找到！
        id转换:00 03 -->00 5d

**********id转换后发给服务器的包**********
00 5d 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 1c 00 01
```

```
C:\Users\12937>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
```

2. 本地解析功能测试

查找在本地 DNS 服务器中存储的域名和对应 IP（**未被屏蔽**），本地 DNS 服务器收到请求报文，查 cache 找到域名对应的 IP，并生成应答报文，发送到请求方

请求查询域名 testel 的 IP。如图所示，得到了对应的 IP 地址 11.111.11.111

```
0.0.0.0 zhp.gdynia.pl
0.0.0.0 test0
11.111.11.111 test1
22.22.222.222 test2
202.108.33.89 www.sina.com.cn
61.135.181.175 sohu
123.127.134.10 bupt
```

```
********************收到请求者的包********************
00 03 01 00 00 01 00 00 00 00 00 00 05 74 65 73 74 31 00 00 1c 00 01
对应网址：test1
在域名解析表中找到!

//////////////////服务器功能:此网址可快速中继//////////////////

********************发送给请求者的包********************
00 03 81 80 00 01 00 01 00 00 00 00 05 74 65 73 74 31 00 00 1c 00 01 c0 0c 00 01 00 01 00 00 01 00 00 04 0b 6f 0b 6f
```

```
C:\Users\12937>nslookup test1
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:    test1
Addresses:  11.111.11.111
          11.111.11.111
```

3. **屏蔽功能测试**

   查找在本地 DNS 服务器中存储的域名和对应 IP（**该域名被屏蔽**），本地 DNS 服务器收到请求报文，查 cache 找到域名对应的 IP，并生成应答报文，发送到请求方（报文的回答数设置为“0”）

   请求查询域名 zhp.gdynia.pl 的 IP。如图所示，不显示其 IP，实现对该域名屏蔽

```
0.0.0.0 zhp.gdynia.pl
0.0.0.0 test0
11.111.11.111 test1
22.22.222.222 test2
202.108.33.89 www.sina.com.cn
61.135.181.175 sohu
123.127.134.10 bupt
```

```
********************收到请求者的包********************
00 05 01 00 00 01 00 00 00 00 00 00 03 7a 68 70 06 67 64 79 6e 69 61 02 70 6c 00 00 1c 00 01
对应网址：zhp.gdynia.pl
在域名解析表中找到!

//////////////////屏蔽功能:此网址被屏蔽//////////////////

********************发送给请求者的包********************
00 05 81 80 00 01 00 00 00 00 00 00 03 7a 68 70 06 67 64 79 6e 69 61 02 70 6c 00 00 1c 00 01 c0 0c 00 01 00 01 00 00 01
00 00 04 00 00 00 00
```

```
C:\Users\12937>nslookup zhp.gdynia.pl
服务器:  UnKnown
Address:  127.0.0.1

*** 没有 zhp.gdynia.pl 可以使用的 internal type for both IPv4 and IPv6 Addresses (A+AAAA)记录
```

4. 不合法（或者不存在）域名测试

查询一个不存在域名对应的 IP，首先本地 DNS 服务器的 cache 中查不到该域名，然后对外部 DNS 发送请求报文，外部 DNS 同样无法找到此域名对应 IP，因此没有应答报文；请求方在一定时间之后收不到应答报文，计时器超时

请求查询域名 999999999 的 IP。如图所示，请求方计时器超时

```
**********收到请求者的包**********
00 03 01 00 00 01 00 00 00 00 00 00 09 39 39 39 39 39 39 39 39 00 00 1c 00 01
对应网址：999999999
在域名解析表中没有找到！
        id转换:00 03 -->00 72

**********id转换后发给服务器的包**********
00 72 01 00 00 01 00 00 00 00 00 00 09 39 39 39 39 39 39 39 39 00 00 1c 00 01
```

```
C:\Users\12937>nslookup 999999999
服务器:  UnKnown
Address:  127.0.0.1

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
*** 请求 UnKnown 超时
```

五、调试中遇到并解决的问题

1、线程问题：

运行过程中，我们发现一旦在 DNS 请求的域名在本地 DNS 服务器的 cache 中不存在，那么从本地 DNS 中继服务器发出的请求报文 就得不到外部 DNS 的响应报文；例如测试样例中的域名“999999999”，实际并不存在，本地 DNS 也不存在。就会导致无应答报文，此时线程将会一直处于等待应答报文的状态，无法处理其他的请求报文。表现形式就是，一旦卡住所有 DNS 请求均无效。

解决方法：在中继功能的时候开启子线程，即把本地 DNS 服务器对外发送请求报

文，然后等待接受应答报文，接收到应答报文后转发给请求方的过程 封装在一个子线程中。这样即使子线程处在等待接受响应报文的状态，也不会阻塞主线程的继续运行

2、计时器超时问题：

使用中继功能时，如果本地 DNS 服务器发送请求报文后，长时间接收不到应答报文，对应子线程应该关闭。这就需要一个超时计时器。开始时我们不知道 VS 中 C 如何实现计时器功能，也就没有做超时处理；最后，我们将代码移植到 QT 中，使用 QT 计时器，在子线程运行到“DNS 发送请求报文”之后启动定时器，一段时间之后，计时器超时，关闭子线程

3、DNS 应答报文格式问题：

当本地 DNS 服务器中存在对应域名时，直接构造响应报文，将响应报文发送给请求方。响应报文格式开始时没搞明白，通过 WireShark 抓包分析，结合网上查找资料，构造了 DNS 相应报文格式如下

```
//构造DNS响应部分
char answer[16];
unsigned short Name = htons(0xc00c);//C00C(1100000000001100，12正好是头部的长度，其正好指向Queries区域的查询名字字段)
memcpy(answer, &Name, sizeof(unsigned short));
len += sizeof(unsigned short);

unsigned short TypeA = htons(0x0001);//查询类型：1，由域名获得IPv4地址
memcpy(answer+len, &TypeA, sizeof(unsigned short));
len += sizeof(unsigned short);

unsigned short ClassA = htons(0x0001);//对于Internet信息，总是IN（0x0001）
memcpy(answer+len, &ClassA, sizeof(unsigned short));
len += sizeof(unsigned short);

unsigned long timeLive = htonl(0x00000100);//生存时间(TTL)
memcpy(answer+len, &timeLive, sizeof(unsigned long));
len += sizeof(unsigned long);

unsigned short IPLen = htons(0x0004);//资源数据长度
memcpy(answer+len, &IPLen, sizeof(unsigned short));
len += sizeof(unsigned short);

unsigned long IP = (unsigned long) inet_addr(DNS_table[find].ip.c_str());//资源数据，即IP
memcpy(answer+len, &IP, sizeof(unsigned long));
len += sizeof(unsigned long);
len += iRecv;
```

4、文件路径问题：

dnsrelay.txt 文件，也就是存储域名和对应 IP 的文件路径开始时存在问题，一直无法打开文件。开始时使用传参的方式将 dnsrelay.txt 文件的路径传给函数。经常出现打开文件失败的信息，最后改为了直接把 dnsrelay.txt 文件的路径写在函数内部。解决了打开文件失败的问题

六、心得体会

通过 DNS 中继服务器实验，理解了域名和 IP 的对应关系。理解了 DNS 请求如何通过发送请求报文和接受应答报文来得到域名对应的 IP。

通过 WireShark 抓包，分析 DNS 报文格式，对 DNS 报文的内容有了深入的认识，理解了每一个字段的含义。

程序的性能尤其重要，虽然单线程也能解决 DNS 中继服务器功能，但是效率不是很乐观，打开网页卡顿明显。用多线程处理后，测试了上网功能，看视频依然十分流畅。甚至打了一把游戏也运行稳定。因此在以后的编程实验中，程序的性能应该作为一个尤其重要的指标来考虑。

DNS 主要用 UDP 报文，本次实验虽然没有触及到 UDP 的协议具体内容，但是对这种无连接报文的工作方式有了大概的认知。

七、函数作用说明

`int get_table(char *t)` 读 dnsrelay.txt 文件，存储到程序中

`void get_URL(char *recv_buf, int num)` 从请求报文中提取出网址

`int Find(char* Website, int num)` 在本地 DNS 服务器中查找域名对应 IP

`unsigned short register_new_ID (unsigned short old_ID, SOCKADDR_IN temp)` 将请求 ID 转换为新的 ID，并将信息写入 ID 转换表中，把旧 id 和套接字地址对应起来，以便查找

`void my_thread::run()` 封装了中继功能，对外部 DNS 发送请求，接受应答，转发给请求方

本地 DNS 存在的域名构造响应报文在 main()函数中