lab3 DNS中继服务器

1.功能设计

(1)不良网站拦截

- 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

(2)服务器功能

- 检索结果为普通ip地址，则向客户端返回这个地址

(3)中继功能

- 表中未检索到该域名，则向因特网DNS服务器发出查询，并将结果返回给客户端

(4)调试级别1

- 输出: **查询的域名及其对应的ip地址**
- 使用默认名字服务器
- 使用默认配置文件

(5)调试级别2

- 输出:**时间坐标**&&**序号**&&**客户端ip地址**&&**查询的域名**
- 使用指定的名字服务器
- 使用指定的配置文件

(6)调试级别3

- 输出:**数据报内容**&&**接收和发送数据包的地址及端口**&&**查询的ip地址及其域名**
- 使用指定的名字服务器
- 使用默认配置文件

(7)多用户并发查询

- 允许多个客户端的并发查询，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

  【实现方式: 向外部DNS服务器发出咨询请求后，不等待其返回，直接进入下一轮循环】

(8)外部DNS应答迟返回

- 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

  【实现方式: 每隔一定时间，对ID转换表中，还没收到外部DNS服务器回复的表项，打印超时汇报】

2.模块划分

DNS服务器主模块包含三个子模块

(1)命令行参数处理

本模块的功能是，依据输入的参数，设置标志数据，以控制消息的输出。通过设置不同的命令行参数，可以使DNS服务器显示不同程度的提示和调试信息。

(2)本地解析

本模块的功能是，在本地缓存文件中，查找从应用程序来的请求解析的域名，得到其对应的ip地址。然后构造DNS应答数据包返回给应用程序。

(3)外部DNS服务器解析

本模块的功能是，本地解析失败后，将应用程序发送的DNS请求报文转发给外部DNS服务器，然后接收应答，并根据此应答，对应用程序进行应答。

3.系统流程图

4.程序运行说明

1. 使用ipconfig/all记录下当前DNS服务器A，在程序的宏里将外部DNS服务器设置为A
2. 将DNS设置为本地主机的IP地址B(控制面板->网络共享中心->wlan->IPv4->dns设置)
3. 运行dnsrelay程序
4. 正常使用ping, ftp, IE等，名字解析工作正常
5. 局域网上的其他计算机将域名服务器指向DNS中继服务器的IP地址，ftp,IE等均能正常工作

输入格式

```
//调试级别1
DNSrelay [1]

//调试级别2
DNSrelay [2][指定外部DNS服务器][指定配置文件]

//调试级别3
DNSrelay [3][指定外部DNS服务器]
```

5.测试用例及运行结果

DNS服务器/配置文件

```
114.214.193.126 //本地IP
202.38.64.56 //外部DNS服务器

example.txt //配置文件
```

###(1)调试级别1

输出查询的域名及其对应的ip地址

使用example.txt 中的测试条目test1,test2,运行情况如下

IE名字解析

浏览器访问www.github.com工作正常

向名字服务器询问地址

cmd中输入指令，向名字服务器询问名字www.ustc.edu.cn的地址

```
nslookup www.ustc.edu.cn
```

结果如下：

不良网站拦截

浏览器访问名字为gamma.vyborg.ru的垃圾网站

查看当前dns cache的内容

实验前用 `ipconfig/flushdns` 命令清空dns缓存，访问github主页后，用 `ipconfig/displaydns` 命令查看当前dns缓存，其部分截图如下，证明运行正确

(2)调试级别2

cmd输入命令

```
nslookup test0
nslookup test1
```

命令行获得正确结果

程序输出:[时间坐标&&序号&&客户端ip地址&&查询的域名]

(3)调试级别3

浏览器访问www.github.com

程序部分输出【数据报内容&&接收和发送数据包的地址及端口&&查询的ip地址及其域名】

6.遇到的问题及心得体会

1.Windows系统下的编译环境问题

使用Devcpp集成编译环境，需要链接Ws2_32.lib库，除了在源代码中加入如下语句以外

```
#pragma  comment(lib,"ws2_32.lib")
```

还需要配置编译器，在编译时加入如下命令

```
-lwsock32
```

2.创建Socket时的阻塞与非阻塞问题

为了满足多用户并行访问需求与超时管理，要将本地DNS的Socket设置为阻塞，外部DNS的Socket设置为非阻塞。

3.报文结构分析

收到的报文缓存在buffer里，对其内容的操作和读取极易出错

4.等待时间的控制

等待外部dns回复的时间要多次尝试后设置，时间过长会导致DNS超时，得不到响应

7.源代码数据结构

###(1)全局

DNS报文首部

```
typedef struct DNSHeader
{
    unsigned short ID;
    unsigned short Flags;
    unsigned short QuestNum;
    unsigned short AnswerNum;
    unsigned short AuthorNum;
    unsigned short AdditionNum;
} DNSHDR, *pDNSHDR;
```

DNS域名解析表

```
typedef struct translate
{
    string IP;                      //IP地址
    string domain;                  //域名
} Translate;
```

ID转换表

```
//ID转换表结构
typedef struct IDChange
{
    unsigned short oldID;           //原有ID
    bool done;                      //标记是否完成解析
    SOCKADDR_IN client;             //请求者套接字地址
    int joinTime;                   //加入转换表的时刻
    char urlName[LENGTH];           //客户询问的url名字
    int offset;                     //客户发送报文的字节数
} IDTransform;
```

域名解析表/ID转换表

```
Translate DNS_table[AMOUNT];        //DNS域名解析表
IDTransform IDTransTable[AMOUNT];   //ID转换表
int IDcount = 0;                    //转换表中的条目个数
char url[LENGTH];                   //域名
```

系统时间

```
SYSTEMTIME sys;                     //系统时间
int Day, Hour, Minute, Second, Milliseconds;//保存系统时间的变量
```

功能函数

```
//函数1:获取域名解析表
int GetTable(char *tablePath);
//函数2: 获取DNS请求中的域名
void GetUrl(char *recvbuf,int recvnum);
//函数3: 判断是否在表中找到DNS请求中的域名，找到返回下标
```

```
int IsFind(char *url,int num);
//函数4：将请求ID转换为新的ID，并将信息写入ID转换表中
unsigned short RegisterNewID(unsigned short oID, SOCKADDR_IN temp, bool ifdone);
//函数5：打印本地构造回复报文的信息(调试等级1,2,3)
void DisplayInfo(unsigned short newID, int find);
//函数6:读取响应报文中url
void readurl(char* buf, char* dest);
//函数7：打印外部DNS响应报文的相关信息(调试级别1&3)
void DisplayAnswer(int m,int level,char *recvbuf);
```

(2)main函数中

外部DNS地址

```
char outerDns[16];
```

配置文件路径

```
char tablePath[100];
```

套接口

```
WSAData wsaData; //套接口
SOCKET  socketLocal,socketServer;    //本地和外部两个套接字
```

数据报的接收与发送

```
SOCKADDR_IN serverName; //外部DNS
SOCKADDR_IN clientName; //请求端
SOCKADDR_IN localName;  //本地DNS
```

缓存

```
char sendbuf[BUF_SIZE]; //发送缓存
char recvbuf[BUF_SIZE]; //接收缓存
```