**北京邮电大学课程设计报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | 计算机网络课程设计 | | **学 院** | 计算机 | **指导教师** | 蒋砚军 |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
| 2015211304 | 23 | 2015211243 | | 苏亦雯 |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | 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：  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）  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）  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） | | | | | |
| **学生**  **课程设计**  **报告**  （附页） |  | | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | **评语**:  **成绩**:  指导教师签名：  年 月 日 | | | | | |

注：评语要体现每个学生的工作情况，可以加页。

# 系统的功能设计

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：

* 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）
* 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）

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）

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

# 实验环境

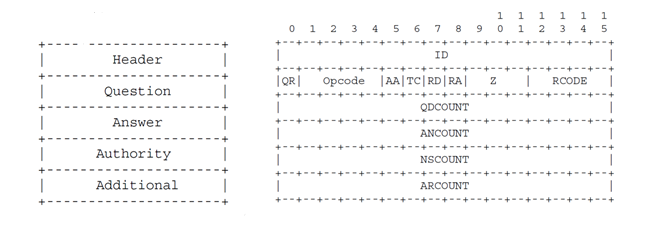
操作系统：windows 10 操作系统

编程环境：code blocks

编程语言：C语言

# 数据结构

## DNS报文说明



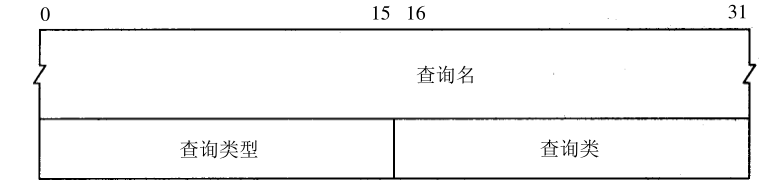
### 报头格式

* ID：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是否匹配
* QR：0表示查询报，1表示响应报。
* OPCODE：通常值为0（标准查询），其他值为1（反向查询）和2（服务器状态请求）。
* AA: 权威答案(Authoritative answer)
* TC: 截断的(Truncated)应答的总长度超512字节时，只返回前512个字节
* RD: 期望递归(Recursion desired) 查询报中设置，响应报中返回

告诉名字服务器处理递归查询。如果该位为0，且被请求的名字服务器没有一个权威回答，就返回一个能解答该查询的其他名字服务器列表，这称为迭代查询

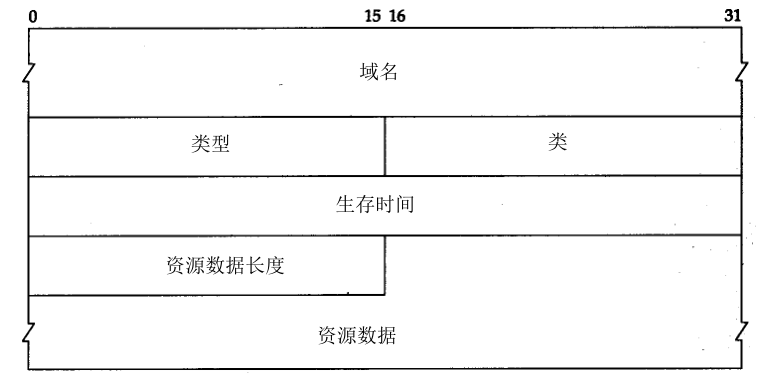
* RA：递归可用(Recursion Available)如果名字服务器支持递归查询，则在响应中该比特置为1
* Z：必须为0，保留字段
* RCODE: 响应码(Response coded)，仅用于响应报值为0(没有差错)，值为3表示名字差错。从权威名字服务器返回，表示在查询中指定域名不存在

### 查询报文中的问题部分



* 查询名：查询名是要查找的名字，它是一个或多个标识符的序列
* 查询类型：每个问题有一个查询类型，而每个响应也有一个类型
* 查询类：最常用的查询类型是A类型，表示期望获得查询名的 I P地址

### 响应报文中资源记录部分



* 域名：记录中资源数据对应的名字
* 类型：说明 R R的类型码。它的值和前面介绍的查询类型值是一样的。
* 类：通常为1，指Internet数据。
* 生存时间：客户程序保留该资源记录的秒数。资源记录通常的生存时间值为 2天。
* 资源数据长度：说明资源数据的数量。该数据的格式依赖于类型字段的值
* 资源数据：对于A类型的响应，资源数据为IP地址

## dns域名解析表结构

class translate

{

public:

string ip;

string domain;

};

translate DNStable[table\_size]; 将电脑中的dnsrelay.txt 中的ip和域名对照输入到这个表。

## 转发包的ID转换表结构

class IDtrans

{

public:

unsigned short id; //旧id

bool finish;//是否完成

SOCKADDR\_IN client; //请求者套接字地址

};

vector <IDtrans> tr; 将客户端的dns转发请求存储在这张表里，新id为旧id+1，即待外部服务器响应之后，可由此表查询到旧id，并进行接下来的操作。

# 模块划分

## 初始化阶段

在这个阶段，初始化socket，host绑定好53端口，外部dns绑定外部自定义的地址，并将本地ip域名对照表录入。

* int get\_table(char\* Path)

入口：本地dnsrelay.txt 路径

作用：将本地dnsrelay.txt 输入到ip和域名的对照表（DNStable）中

返回值：DNStable行数

## 处理请求包

在这个阶段，解析dns，将域名从收到的信息中提取出来，分析是否在ip域名对照表内，在表内则作为服务器功能自己构造响应包，若不在表内，则在将旧id存入ID转换表并换上新id后转发向外部dns。

* int get\_url(string rec,int num)

入口：dns收到的从host发来的dns信息，这段信息的长度

作用：将用户需要的域名从这段信息中提取出来，并付给全局变量url

返回值：url的长度

* string find\_url()

入口：已经解析完毕的全局变量url

作用：在ip域名对照表中查找对应ip

返回值：若存在，则返回对应ip，若不存在，则返回notexist

* int create\_response(char\* sendbuf,char\* recbuf, int rnum,string findout, int &length)

入口：发送缓冲区，接收缓冲区，接收到的字节数，查找成功的对应ip，发送缓冲区字节长度

作用：在ip域名对照表中存在url的情况下，构造响应。若findout为0.0.0.0，则要将其屏蔽，若不是则要按照响应包标准构造响应包，并将构造的包赋值给sendbuf

返回值：若屏蔽则返回0，否则返回1

* unsigned short get\_new\_id (unsigned short old\_id, SOCKADDR\_IN addr, bool ifdone)

入口: 请求包的旧id，套接字地址，是否已经完成响应

作用：建立一个新的ID转换表的表项，并完成该表项的初始化

返回值：转送包的新id

* unsigned short create\_outside(char\* recbuf, SOCKADDR\_IN clientname)

入口：发送缓冲区内得数据，套接字地址

作用：得到转送包的新的id

返回值：转送包的新id

## 接收外部响应包

在这个阶段，从外部服务器接收到响应包，将包的id与ID转换表对应，得到旧的id，将旧id替换新的id并将整个包发送回给host

## 参考实现函数

* void Display2(string findout)

入口：查找成功的对应ip

作用：打印输出时间坐标，序号，客户端IP地址，查询的域名

* string Hex2String(char \*pszHex, int nLen)

入口：接收到的包的数据，包的数据长度

作用：将其每字节转换为可打印的16进制数，并打印出来

返回值：转换结束的16进制字符串

## 超时处理

由于整个函数使用的是socket编程，在查看相关资料时发现有select()函数可以判断超时，并判断哪个socket发生了可读可写的事件。所以在每次接收开始前，都用select函数判断是否超时，若没超时则可以继续下列的操作。

## 其他辅助函数

* void split(const string& s, vector<string>& v, const string& c)

入口：要分割的字符串，字符串矢量，分割标识符

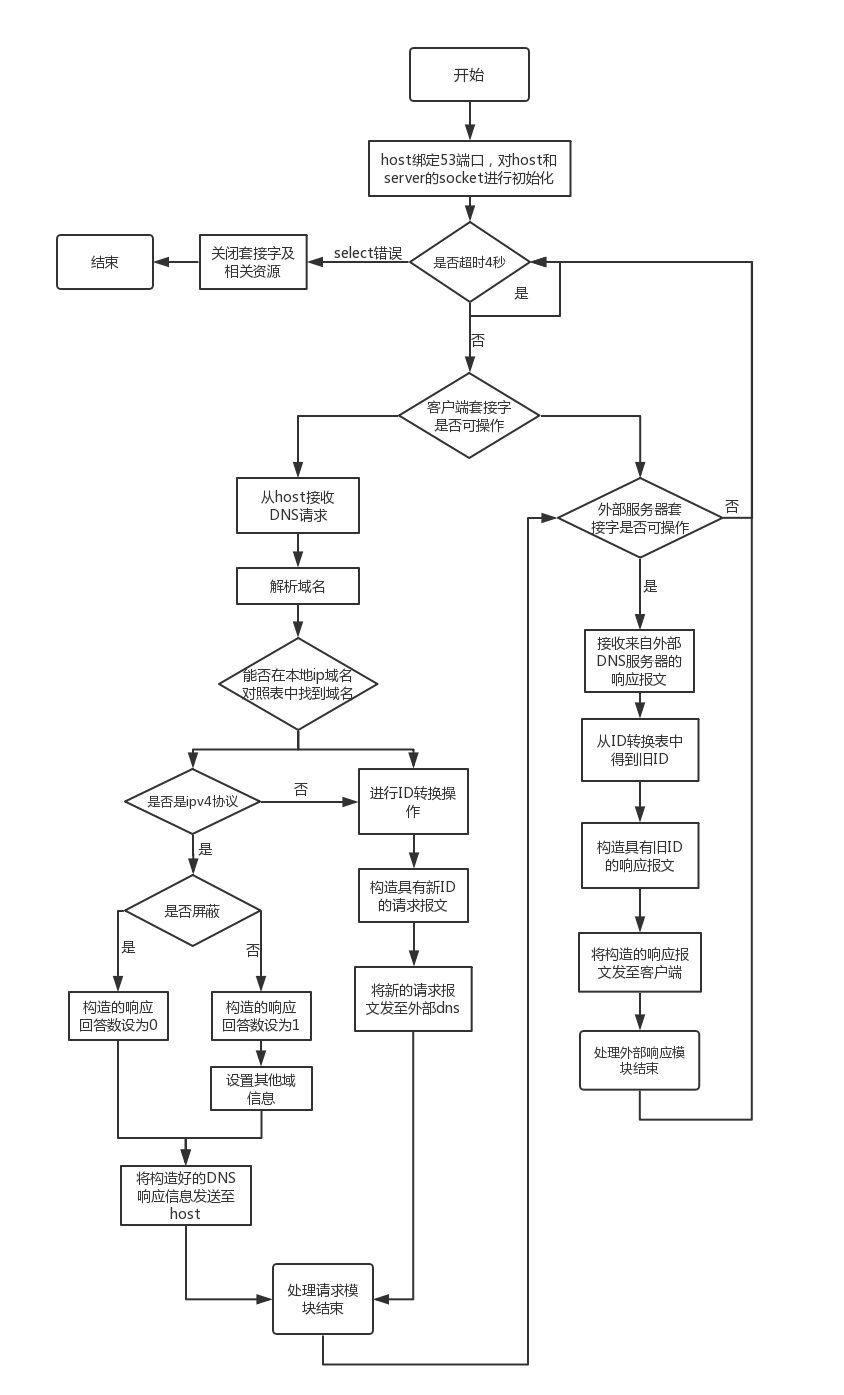
作用：以标识符为准，分割待分割字符串，将分割后的字符串依次存到字符串向量中

返回值：无

## 全局变量

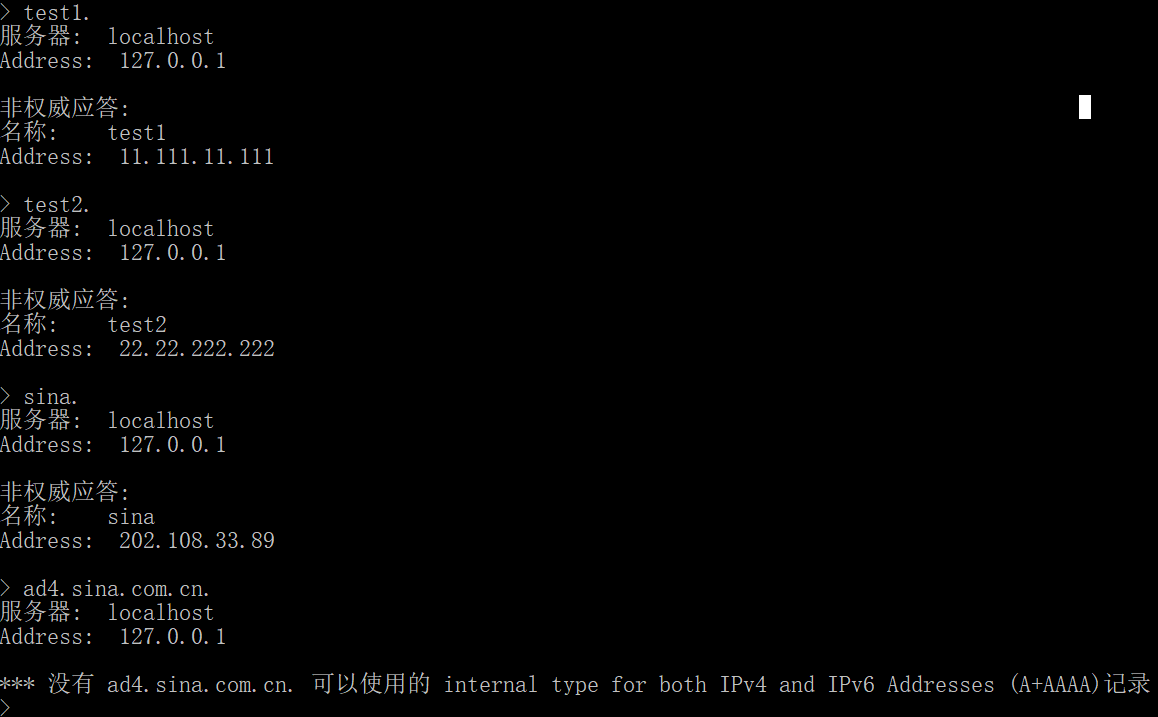
* translate DNStable[table\_size]：ip域名对照表
* vector <IDtrans> tr：新旧ID转换表
* string url：解析后的域名
* int order：收包序列号

# 软件流程图



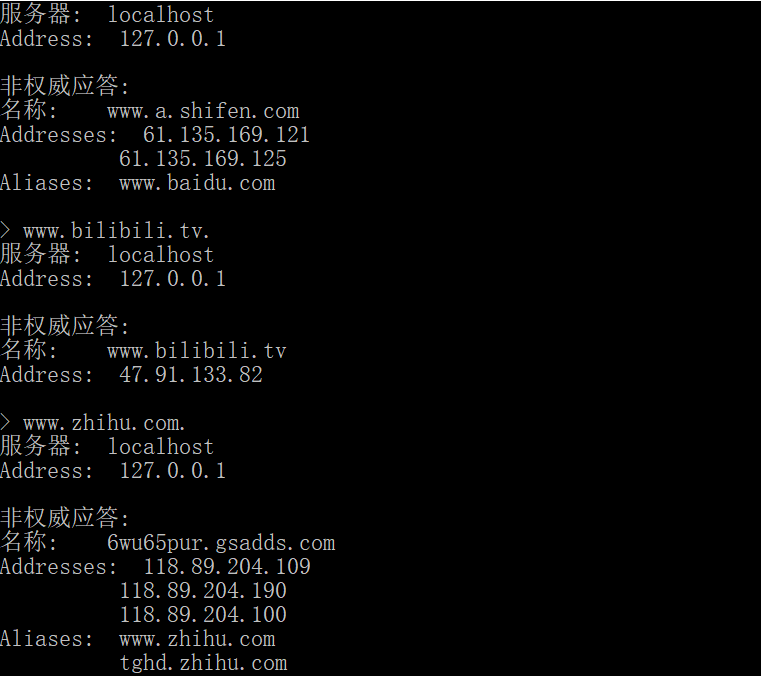
# 测试用例以及运行结果

## 本地查找



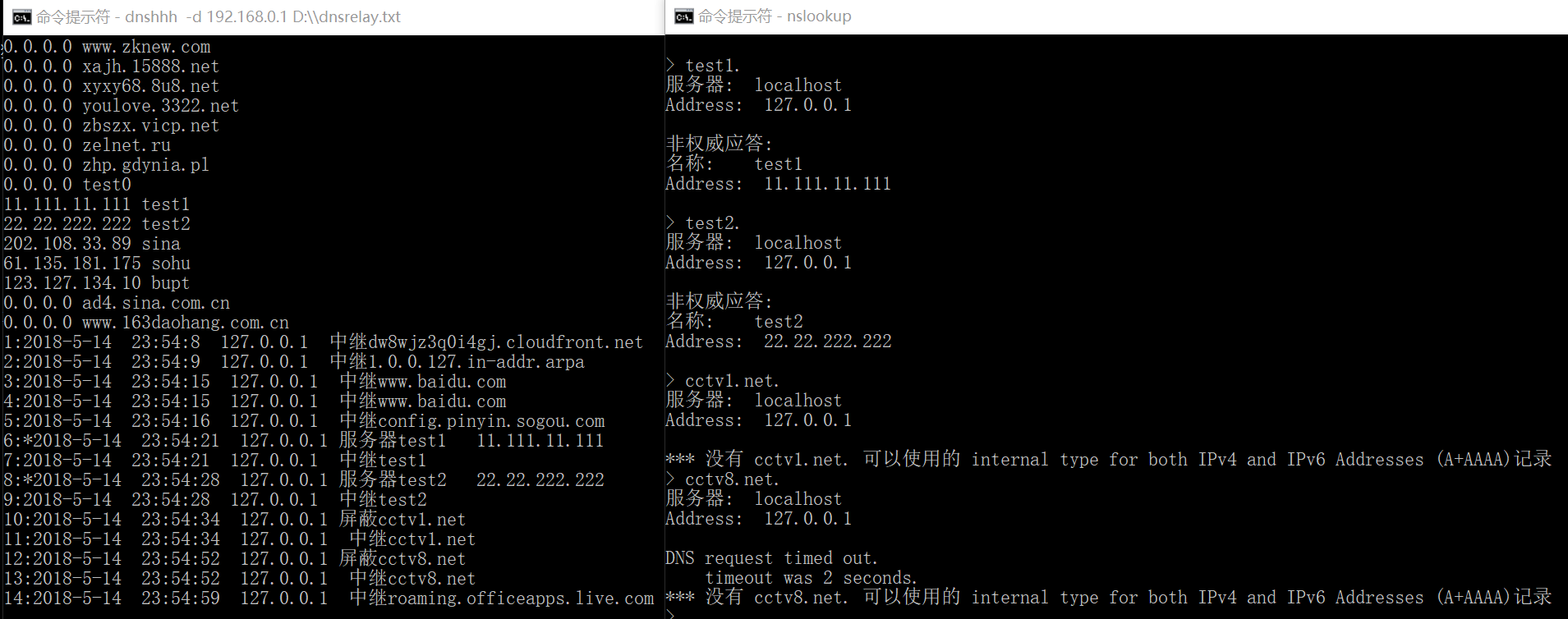
从图中可见，解析的域名与dnsrelay.txt中对应的域名相同，可见结果正确。当为屏蔽时，因为将回答域改为0，所以cmd显示的是没有可以使用的类型。

## 中继



在dns对照表中没找到的时候，软件将这个请求转发给外部服务器，外部服务器解析完毕后返回，可见结果正确。

## 调试信息级别1



这个打开的是位于D盘目录下的dnsrelay信息。由于nslookup每次发送两个请求，分别是IPV4和IPV6，而在设计软件时将ipv6转发给了外部服务器，所以即使在dns对照表中含有该域名，也会有中继这种情况。Baidu本身这个域名有两台主机，所以他被中继了两次，这与情况是对应的。

## 调试信息级别2



输出了相应的包信息，调试信息2因为包含打印时间，所以有时会超时，比无调试信息输出慢。但可以查看报文信息和格式。

## 批量测试

通过编写脚本进行批量测试





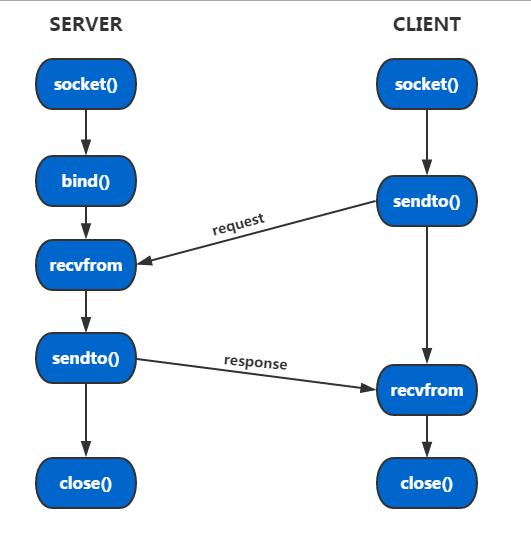
通过测试发现，因为是一次发送多次数据，所以处理起来会有延迟是很正常的。尽管会有延迟，但是处理批量指令的时候还是很准确的，顺序正确。

# 调试中遇到并解决的问题

## 熟悉 UDP

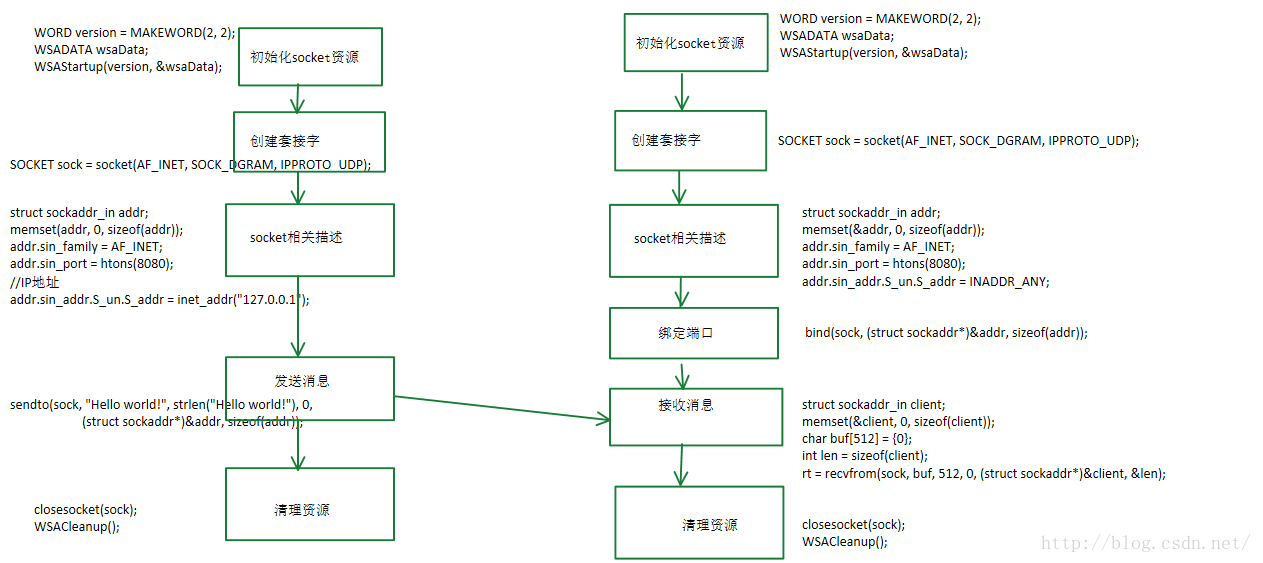
从未进行过socket编程，所以最开始是非常蒙的，在梳理编程逻辑之前，先是要熟悉udp协议。从网上得知：UDP（user datagram protocol）的中文叫用户数据报协议，属于传输层。UDP是面向非连接的协议，它不与对方建立连接，而是直接把我要发的数据报发给对方。所以UDP适用于一次传输数据量很少、对可靠性要求不高的或对实时性要求高的应用场景。正因为UDP无需建立类如三次握手的连接，而使得通信效率很高。

所以从上可知，UDP的socket建立实际上非常简单，简单来说呈现如下情况：



这个图大致模拟了一个本地dns服务器的过程，可以大致得到编程思路，即我需要在本地和服务器各建立一个socket。

## 熟悉socket在udp下的操作过程



后来在客户端，本地，外部服务器各建立了一个socket，但是不知道如何初始化客户端，即host socket绑定53端口，但是client绑定什么呢，需不需要初始化呢？

通过上图，得知了正确的初始化方式，即host和server该建立，host绑定53，server只需要初始化套接字地址。客户端地址则在接受消息和发送消息时候使用。同是通过这张图也理解了发送和接收的代码如何编写。

## 使用nslookup返回多个相同的地址

在使用nslookup的时候，最开始如果查询test，则会返回两个相同的11.111.11.111地址，但是在向外部dns服务器访问时则不会有这个问题，即查询www.baidu.com.的时候虽然也是返回两个地址，但是地址是不同的，即百度公司在这个域名有两台主机。通过调试程序3可以发现type有所不同，即虽然返回的地址都是11.111.11.111，但是数据包type域部分一个是28，一个是1。

通过查询和询问同学，得知nslookup是同是发送ipv4数据包和ipv6数据包的, 即最开始因为没有分辨ipv4和ipv6导致同时进行两个查询。解决方法是将ipv6数据报转发出去，只自己查询ipv4数据报。

## 超时问题

因为要判断接收超时，所以需要考虑两个问题：1.一个已连接UDP套接字能且仅能与一个对端交换数据报，那么客户端发送广播的时候如何防止recvfrom方法阻塞；2.服务端忙的时候，已连接的UDP套接字也会被阻塞。

通过查询资料知道了select（）函数，并学会了使用。

<https://blog.csdn.net/junjun150013652/article/details/37568881>

## 不可靠问题

考虑到可能还会有多客户端的问题曾经考虑过使用多线程，后来想到UDP协议是一个无连接不可靠的协议，本身就是一对多的协议，使用多线程并没有任何用处。对于UDP协议来说，因为本身就不可靠，所以超时或者错误就直接丢包就好，并不需要像TCP/IP一样考虑太多的复杂情况。

# 心得体会

通过这次课程设计，让我有种重新学习了udp协议和dns的感觉。通过使用wireshark抓包来熟悉报文格式，也让我熟悉了wireshark的使用方式。也从0开始学习了socket编程相关知识，虽然没有精通，但是在磕磕绊绊中，也算是稍微懂了一些socket的编写方法，理解了sendto和recvfrom的使用。在理解了并梳理了dns服务的过程之后，代码编写是非常快的。因为在考虑超时问题时，通过查询资料学会了select函数，也学习了select函数的其他用法，虽然并没有使用但是受益匪浅。

# 代码列表