**计算机网络课程设计**

**DNS中继器**

**目录**

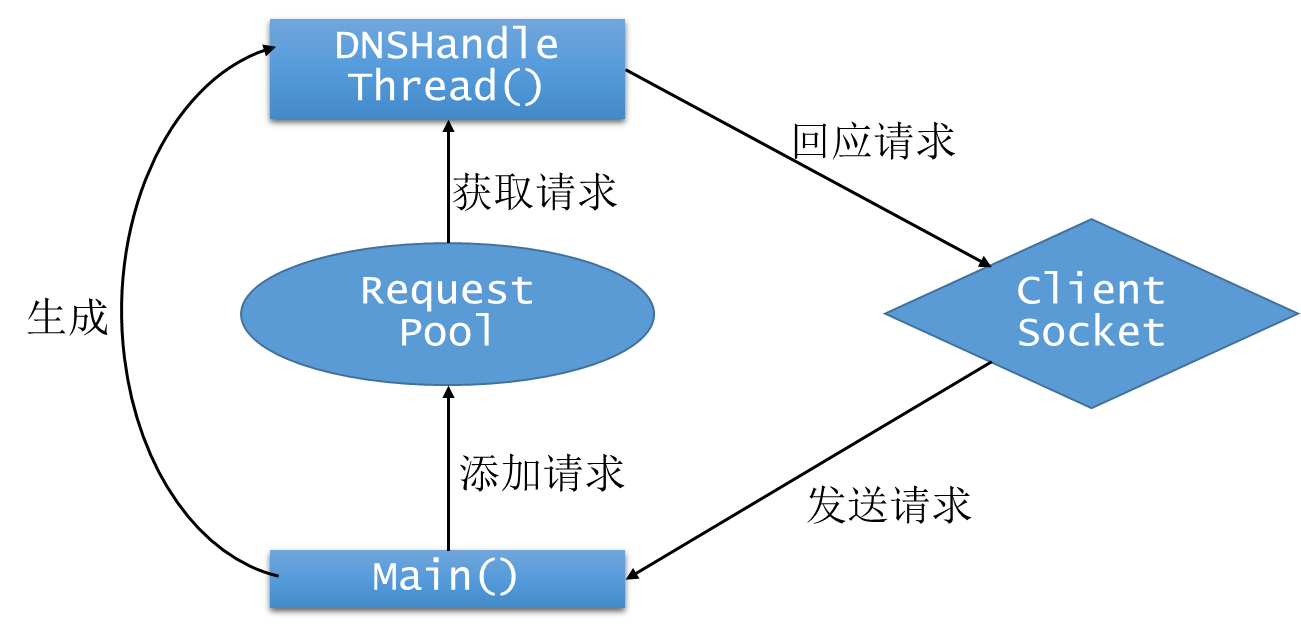
1. 系统结构设计·············································································1
   1. 基本功能··············································································1
2. 运行环境···················································································1
3. 模块划分···················································································2
   1. 存储模块··············································································2
   2. 函数模块··············································································2
4. 程序流程图················································································4
   1. 主函数流程图········································································4
   2. 线程流程图···········································································5
5. 测试用例及运行结果····································································5
   1. 基本功能测试········································································6
6. 遇到问题及解决方法··································································10
7. 实验心得·················································································12
8. **系统功能设计**
   1. 基本要求：
      1. 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：
         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）
         2. 普通IP地址，则向客户端返回该地址（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）
      2. 需要进行消息ID的转换，以满足多个计算机上的客户端会同时查询
9. **实验环境**
   1. 操作系统：Windows 7 x64
   2. 编译环境：Visual Studio 2012
10. **模块划分：**
    1. 存储部分：
       1. host\_item \*hosts\_list[MAX\_HOST\_ITEM]

存储E:\\dnsrelay.txt中导入的所有地址，包括屏蔽地址和已知的缓存地址。

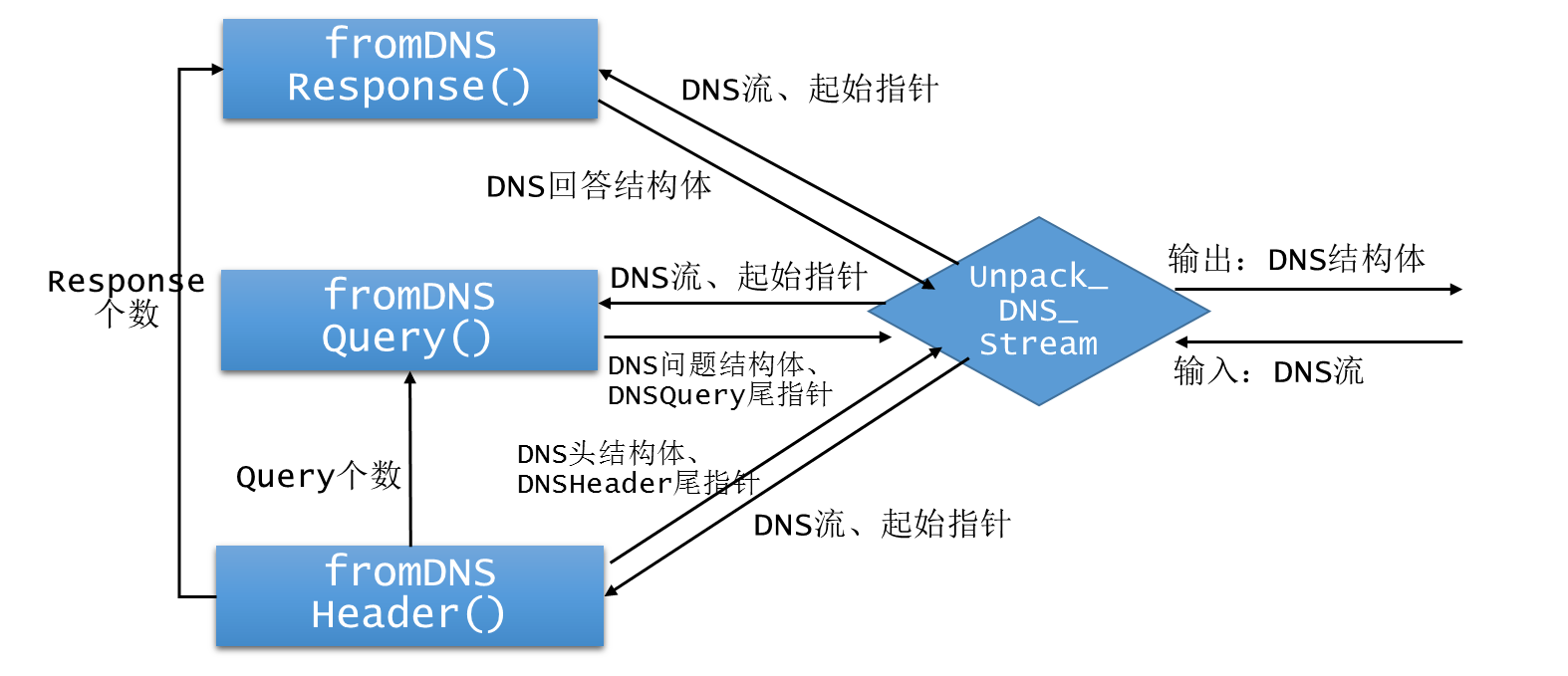
* + 1. std::vector<DNSRequest\*> request\_pool;

当客户端通过53端口发来数据后，处理函数将请求以DNSRequest类型的格式加入到request\_pool这个队列中。所有DNS处理线程从该队列中获取请求，并处理。

* 1. 函数模块
     1. 主体部分：处理DNS请求



* + 1. 处理DNS Packet，将二进制流变为程序可处理的部分：



* + 1. 获取当前地址的类型，包括缓存地址、屏蔽地址以及未知地址：



* + 1. 缓存管理：



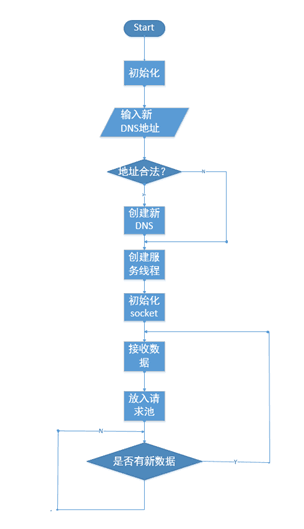
* + 1. ID管理：

在处理过程中，由于各个客户端的发来的DNS请求中ID并不唯一，可能出现重复ID导致错误，因此，添加ID管理以重新分配DNSQuery的请求号，防止重复ID导致的错误。

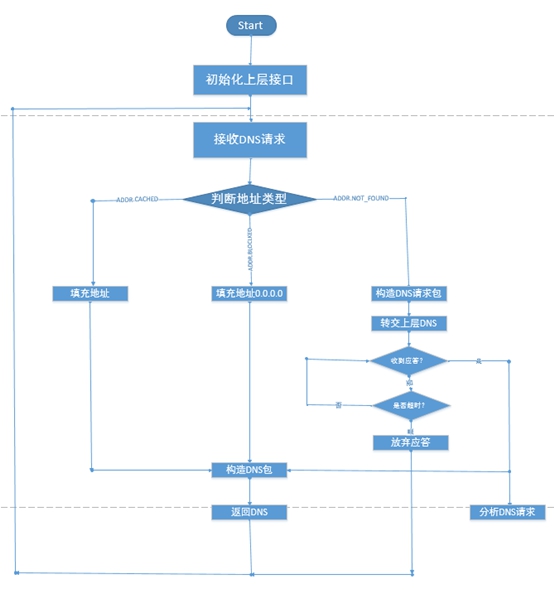


1. **程序流程图**

1、主线程



2、服务线程

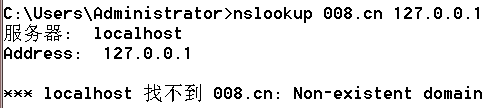


1. **测试用例及运行结果：**
   1. 基本功能测试：
      1. DNS屏蔽功能：

选取DNS屏蔽地址列表中的地址之一：

0.0.0.0 008.cn

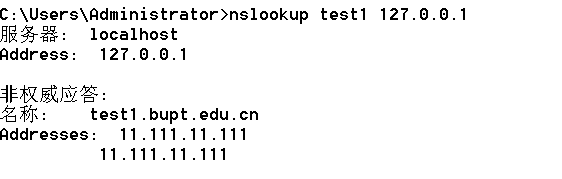
通过nslookup查找该主机地址，结果如下图所示



结果表明，DNS屏蔽功能有效。

* + 1. DNS服务器功能

选择一个在DNS中存在的地址，如：11.111.11.111 test1

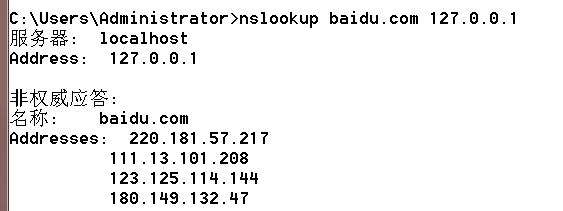


* + 1. DNS中继功能：

选一个在DNS中不存在的地址，如：[baidu.com](http://www.youku.com)

在cmd中输入：nslookup [baidu.com](http://www.youku.com)

结果如下：



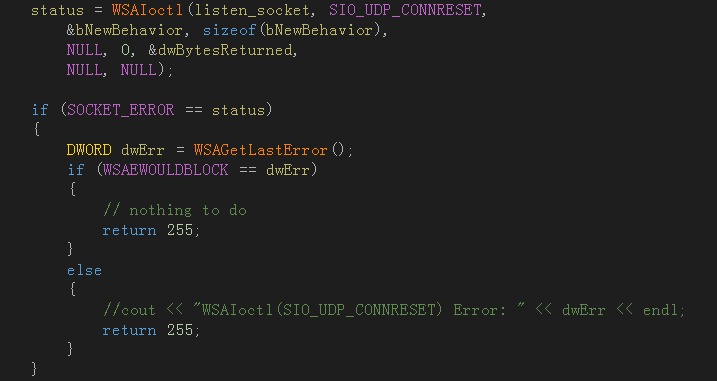
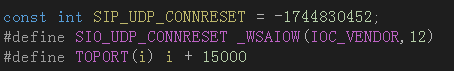
可以正确解析服务器地址。

1. **遇到问题及解决方案：**
   1. DNS使用UDP方式传送数据，但是在使用recvfrom函数时返回错误：10053 WSAECONNRST

但是，UDP根本不需要连接，不知道怎么出现的超时。

解决方案：

在UDP初始化时增加如下代码：



* 1. 大端、小端字序问题

网络字节序用的是大端法，而x86架构使用的是小端法，因此在处理包的时候需要加htons，htonl，ntohs，ntohl这样的宏来完成大小端的转换。

* 1. 缓存无法匹配：

调试过程中，发现在nslookup 时，查询地址[www.tudou.com.bupt.edu.cn](http://www.tudou.com.bupt.edu.cn)和[www.tudou.com](http://www.tudou.com)正确匹配。

解决：

问题在于匹配函数strcmp，strcmp(s1, s2)函数在s1中包含s2字符串时就会返回=0的结果，然而，两串字符串的长度并不相等。可以将两个字符串转换成C++的String类，直接通过if(s1==s2)的方式比较即可，这样也预防了字符串本身即长度等其他方面伪匹配的问题。

1. **实验心得**

关于计算机网络这部分内容，我们在之前的学习和大作业中有过一定的接触，但当时都选择了可靠性较好的TCP协议，而UDP协议的了解较少。在程序开发中，我们发现，UDP协议是一个不可靠、无差错控制的协议，所以在传输数据包的时候，会出现丢包或乱序的问题。基于这个问题，我选择了一个开销比较大，但是保证较好可靠性的方法：使用多线程的方法处理数据的发送和接收，将数据通信进行隔离，以保证通信的可靠性。

既然使用了多线程，自然要考虑到线程之间的调度和锁的问题。程序中，在对DNS请求池等线程间共有资源进行读写操作的时候，我们进行了加锁处理。

除此之外，这次课程设计对我们启发较大的是，对于软件架构的设计十分重要。在软件或程序开发初期，一个好的架构设计可以让程序员在开发上省去很多工作量。