北京郵電大學

计算机网络课程设计报告

——“DNS 中继服务器”的实现

姓　　名　董奕柳、李岩

班　　级　2021211304

学　　号　2021210868、2021211494

任课教师　高占春

2023 年 6 月

目录

1 课程设计简介

1.1 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：检索结果：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：普通 IP 地址，则向客户端返回该地址（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。

补充，对照表中虽然是 IPv4 的 0.0.0.0，但 IPv6 也是应该要拦截报错的，以“域名”为准。

多客户端并发：允许多个客户端（可能会位于不同的多个计算机）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（DNS 协议头中 ID 字段的作用)。
超时处理：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。

1.2 设备环境

C 语言。
Windows 11，Visual Studio Code，学校校园网，Wireshark。
电脑、手机等可联网设备。

2 系统的功能设计

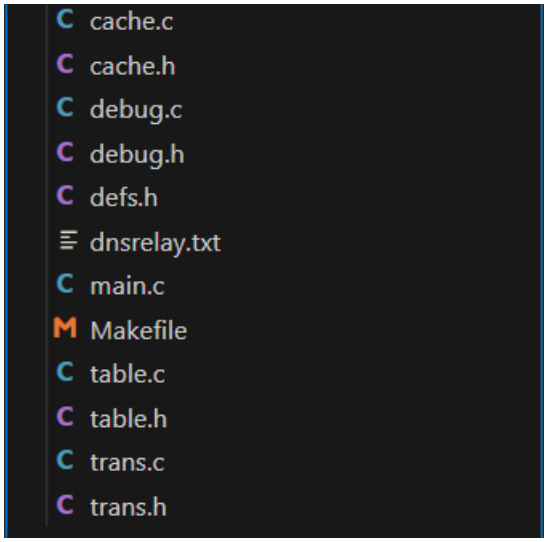
2.1 程序功能

能够对程序的参数进行解析，读入配置文件，启动一个可供使用的 DNS 中继服务器，完成中继和转发和屏蔽的功能。
使用阻塞型 socket 完成多用户多并发的处理，使用广集的方式获取主机端和其它 dns 服务器数据报。通过本地缓存中的转发表，完成对不同用户的区分，将对应的查询报文转发到正确的用户主机中，可以从逻辑上避免长时间监听一地址的信息导致程序处于长期等待。

2.2 使用的数据结构与算法

1 对读入的表按域名字典序排序，供此后二分查找减小时间复杂度。
2 cache 中维护了一个优先队列，自动进行 LRU 原则的替换，由于链式结构的有序性，可以很方便地“挤出”一个最不常用数据，从而释放内存。
3 ID 转换中实现了一个最小时间常数的哈希算法，就近探测空闲空间，选取转换后的 ID 并记录。并且通过设置存入时间和生存时间的方式避免了多个计时器占用大量时间空间的问题。

3 模块划分和软件流程图



defs 模块定义了一些基本常量和结构体，以及初始化的函数。常量包括最大请求大小、请求超时时间、DNS 服务器 IP 地址、DNS 表文件名、最大缓存条目数、最大表条目数、DNS 消息大小限制、名称大小限制、DNS 查询类型等。结构体包括 DNS 消息头、DNS 问题、DNS 资源记录。函数包括 socketInit()和 socket_close()，前者用于初始化套接字和绑定端口，后者用于关闭套接字和终止 DLL 的使用。

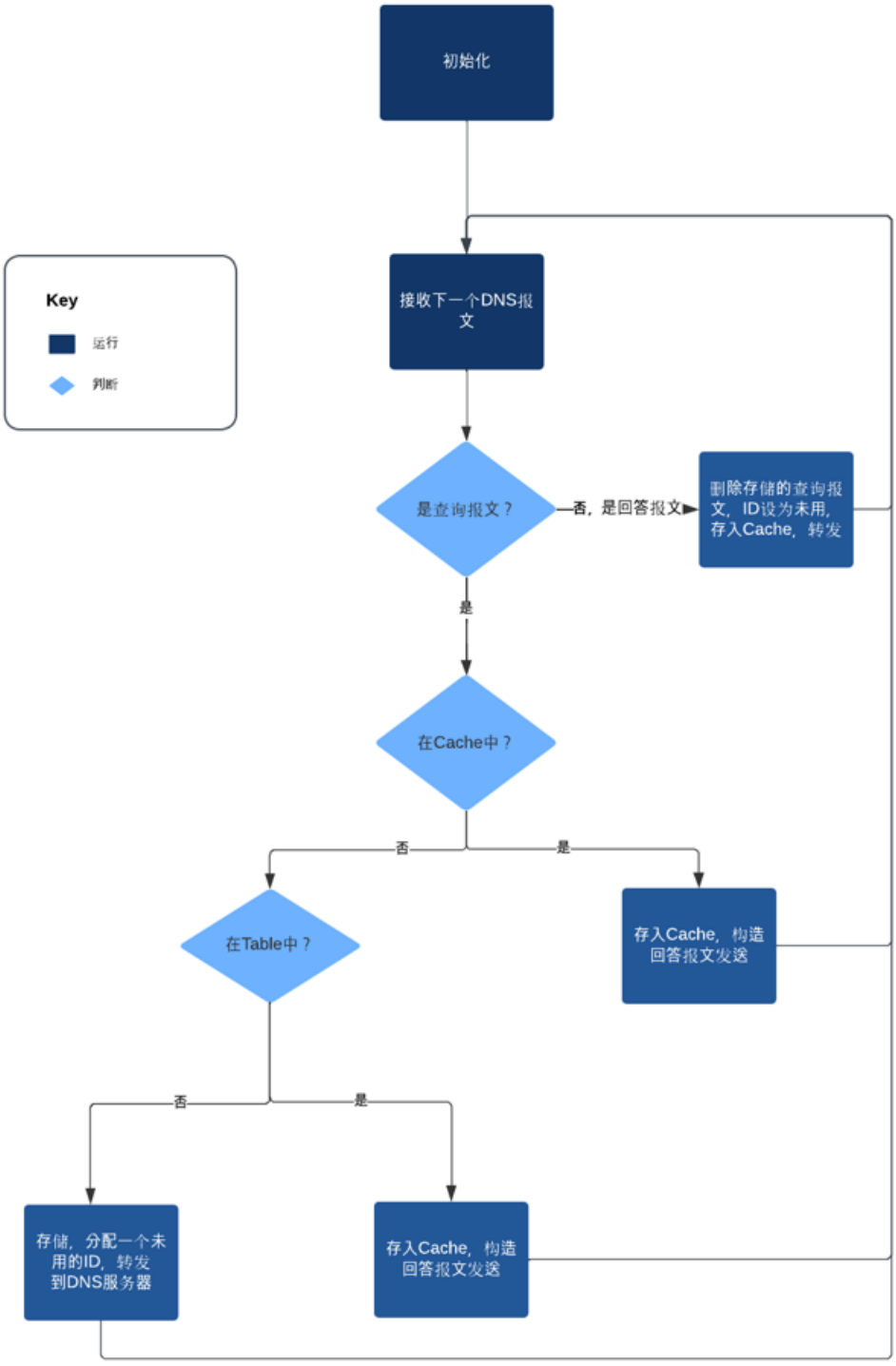
trans 模块定义了 requset_info 的结构体，其中包含了请求信息的一些字段，如是否使用、开始时间、原始 ID、IP 地址和端口号等。同时，还定义了两个函数，分别是 saveRequest()和 acquireRequest()。前者用于保存请求信息到哈希表中，后者用于从哈希表中获取请求信息。这些函数的参数包括源地址、原始 ID 和目标地址等。

table 模块包含了一个 DNS 表，用于存储域名和对应的 IP 地址。定义了 DNSTable 的结构体，其中包含了 DNSRecord 结构体数组，用于存储域名和对应的 IP 地址。同时，还定义了两个函数，分别是 parseTable()和 find_table()。前者用于解析 DNS 表文件并将其存储到 DNSTable 结构体中，后者用于从 DNSTable 结构体中查找域名对应的 IP 地址。这些函数的参数包括 DNS 表文件路径、DNS 查询问题和 DNS 资源记录等。

cache 模块包含了一个 DNS 缓存。定义了 Node 的结构体，用于表示双向链表的节点，包含了域名、IP 地址和过期时间等字段。同时，还定义了一些函数，如 createNode()、insertNode()、deleteFirstNode()、deleteNode()、printList()和 freeList()等，用于创建节点、插入节点、删除节点、打印链表和释放链表内存等操作。此外，还定义了 cache()和 find_cache()两个函数，用于将 DNS 查询结果缓存到链表中，并从链表中查找域名对应的 IP 地址。这些函数的参数包括 DNS 查询问题和 DNS 资源记录等。

debug 模块包含输出调试信息的函数。其中定义了一些用于调试的函数，如 debug()、debugTime()、debugQname()、debugIp()和 get_ms()等。这些函数用于输出调试信息，如时间戳、域名、IP 地址等。同时，还包含了 debug_level 的全局变量和 epoch 的全局变量，用于控制调试信息的输出级别和计算时间戳。

流程图如下：



4 测试用例以及运行结果

4.1 使用测试

机器把 DNS 端口设置为运行程序的机器的 IP 地址（若为本机即为环回地址）后，可以进行

正常的网络使用，与往常无异。

4.2 多并发测试

```
0260s356ms Query    from 10.128.201.239  ID: caa6 -> caa6       Type: A         ldc.lenovo.com.cn -> ?
0260s363ms Response from 202.99.96.68    ID: caa6 <- caa6       Type: CNAME     ldc.lenovo.com.cn <- notA
0263s698ms Query    from 10.128.201.239  ID: 0629 -> 0629       Type: A         pronhub.com -> ?
0263s824ms Query    from 10.128.201.239  ID: 0629 -> 0729       Type: A         pronhub.com -> ?
0263s830ms Response from 202.99.96.68    ID: 0629 <- 0629       Type: A         pronhub.com <- 66.254.114.211
0263s833ms Response from 202.99.96.68    ID: 0629 <- 0729       Type: A         pronhub.com <- 66.254.114.211
0274s826ms Query    from 10.128.201.239  ID: 7a91 -> 7a91       Type: A         dingzhen.com -> ?
0274s951ms Query    from 10.128.201.239  ID: 7a91 -> 7b91       Type: A         dingzhen.com -> ?
0274s961ms Response from 202.99.96.68    ID: 7a91 <- 7b91       Type: A         dingzhen.com <- 154.39.68.41
0274s982ms Response from 202.99.96.68    ID: 7a91 <- 7a91       Type: A         dingzhen.com <- 154.39.68.41
0288s237ms Query    from 10.128.201.239  ID: b09d -> b09d       Type: A         oth.eve.mdt.qq.com -> ?
0288s244ms Response from 202.99.96.68    ID: b09d <- b09d       Type: CNAME     oth.eve.mdt.qq.com <- notA
0326s122ms Query    from 10.128.201.239  ID: f123 -> f123       Type: A         w.deepl.com -> ?
0326s127ms Response from 202.99.96.68    ID: f123 <- f123       Type: CNAME     w.deepl.com <- notA
0343s126ms Query    from 10.128.201.239  ID: d9a3 -> d9a3       Type: A         data.ab.qq.com -> ?
0343s131ms Response from 202.99.96.68    ID: d9a3 <- d9a3       Type: CNAME     data.ab.qq.com <- notA
0344s149ms Query    from 127.0.0.1       ID: 7c5c -> 7c5c       Type: A         www.msftconnecttest.com -> ?
0344s152ms Response from 202.99.96.68    ID: 7c5c <- 7c5c       Type: CNAME     www.msftconnecttest.com <- notA
0379s699ms Query    from 127.0.0.1       ID: e6ee -> e6ee       Type: A         www.msftconnecttest.com -> ?
0379s705ms Response from 202.99.96.68    ID: e6ee <- e6ee       Type: CNAME     www.msftconnecttest.com <- notA
```

```
1837s810ms Query    from 10.28.190.10    ID: a1aa -> a1aa       Type: 28        vv.video.qq.com -> ?
1837s810ms Query    from 10.28.190.10    ID: 9dc3 -> 9dc3       Type: A         appcfg.v.qq.com -> ?
1837s825ms Query    from 10.28.190.10    ID: a369 -> a369       Type: A         vv.video.qq.com -> ?
1837s827ms Response from 202.99.96.68    ID: 9a8c <- 9a8c       Type: CNAME     appcfg.v.qq.com <- notA
1837s827ms Response from 202.99.96.68    ID: a1aa <- a1aa       Type: CNAME     vv.video.qq.com <- notA
1837s827ms Response from 202.99.96.68    ID: 9dc3 <- 9dc3       Type: CNAME     appcfg.v.qq.com <- notA
1837s827ms Response from 202.99.96.68    ID: a369 <- a369       Type: CNAME     vv.video.qq.com <- notA
1879s686ms Query    from 127.0.0.1       ID: 6da5 -> 6da5       Type: A         chinaeast2-0.in.applicationinsights.azure.cn -> ?
1879s693ms Response from 202.99.96.68    ID: 6da5 <- 6da5       Type: CNAME     chinaeast2-0.in.applicationinsights.azure.cn <- notA
1882s554ms Query    from 127.0.0.1       ID: 0495 -> 0495       Type: A         service-aggregation-layer.juno.ea.com -> ?
1882s557ms Response from 202.99.96.68    ID: 0495 <- 0495       Type: CNAME     service-aggregation-layer.juno.ea.com <- notA
1890s743ms Query    from 10.28.190.10    ID: 5257 -> 5257       Type: 28        iv6.gtimg.cn -> ?
1890s743ms Query    from 10.28.190.10    ID: de1f -> de1f       Type: A         iv6.gtimg.cn -> ?
1890s743ms Response from 202.99.96.68    ID: 5257 <- 5257       Type: CNAME     iv6.gtimg.cn <- notA
1890s758ms Response from 202.99.96.68    ID: de1f <- de1f       Type: CNAME     iv6.gtimg.cn <- notA
```

尝试了手机(10.28.190.10)、和非本地的电脑(10.128.201.239)、和本地(127.0.0.1)三者同时运行。均能正常使用。

4.3 压力测试

使用下图的脚本软件快速发送大量 DNS 请求，观察是否出现错误。

```python
import os

while True:
    domain = input("请输入要查询的域名（输入 'exit' 退出）: ")

    if domain == 'exit':
        break

    # 运行nslookup命令
    command = f"nslookup {domain}"
    response = os.system(command)

    # 检查命令的执行结果
    if response == 0:
        print("查询成功！")
    else:
        print("查询失败！")
```

```
0180s338ms Response from 10.3.9.44      ID: 0003 <- 0003      Type: 6       meituan.com <- notA
0180s376ms Query    from 10.128.201.239 ID: 0001 -> 0001      Type: 12      165.224.128.10.in-addr.arpa -> ?
0180s379ms Response from 10.3.9.44      ID: 0001 <- 0001      Type: Null    165.224.128.10.in-addr.arpa <- No such domain
0180s383ms Query    from 10.128.201.239 ID: 0002 -> 0002      Type: A       suning.com -> ?
0180s395ms Response from 10.3.9.44      ID: 0002 <- 0002      Type: A       suning.com <- 112.25.234.196
0180s403ms Query    from 10.128.201.239 ID: 0003 -> 0003      Type: AAAA    suning.com -> ?
0180s448ms Response from 10.3.9.44      ID: 0003 <- 0003      Type: 6       suning.com <- notA
0180s487ms Query    from 10.128.201.239 ID: 0001 -> 0001      Type: 12      165.224.128.10.in-addr.arpa -> ?
0180s492ms Response from 10.3.9.44      ID: 0001 <- 0001      Type: Null    165.224.128.10.in-addr.arpa <- No such domain
0180s499ms Query    from 10.128.201.239 ID: 0002 -> 0002      Type: A       vip.com -> ?
0180s502ms Response from 10.3.9.44      ID: 0002 <- 0002      Type: A       vip.com <- 14.119.64.131
0180s509ms Query    from 10.128.201.239 ID: 0003 -> 0003      Type: AAAA    vip.com -> ?
0180s560ms Response from 10.3.9.44      ID: 0003 <- 0003      Type: 6       vip.com <- notA
0180s597ms Query    from 10.128.201.239 ID: 0001 -> 0001      Type: 12      165.224.128.10.in-addr.arpa -> ?
0180s602ms Response from 10.3.9.44      ID: 0001 <- 0001      Type: Null    165.224.128.10.in-addr.arpa <- No such domain
0180s607ms Query    from 10.128.201.239 ID: 0002 -> 0002      Type: A       tuniu.com -> ?
0180s622ms Response from 10.3.9.44      ID: 0002 <- 0002      Type: A       tuniu.com <- 180.97.6.14
0180s630ms Query    from 10.128.201.239 ID: 0003 -> 0003      Type: AAAA    tuniu.com -> ?
0180s642ms Response from 10.3.9.44      ID: 0003 <- 0003      Type: 6       tuniu.com <- notA
0180s678ms Query    from 10.128.201.239 ID: 0001 -> 0001      Type: 12      165.224.128.10.in-addr.arpa -> ?
0180s682ms Response from 10.3.9.44      ID: 0001 <- 0001      Type: Null    165.224.128.10.in-addr.arpa <- No such domain
0180s688ms Query    from 10.128.201.239 ID: 0002 -> 0002      Type: A       maoyan.com -> ?
0180s698ms Response from 10.3.9.44      ID: 0002 <- 0002      Type: A       maoyan.com <- 81.70.60.59
0180s706ms Query    from 10.128.201.239 ID: 0003 -> 0003      Type: AAAA    maoyan.com -> ?
0180s712ms Response from 10.3.9.44      ID: 0003 <- 0003      Type: 6       maoyan.com <- notA
0180s750ms Query    from 10.128.201.239 ID: 0001 -> 0001      Type: 12      165.224.128.10.in-addr.arpa -> ?
0180s756ms Response from 10.3.9.44      ID: 0001 <- 0001      Type: Null    165.224.128.10.in-addr.arpa <- No such domain
0180s764ms Query    from 10.128.201.239 ID: 0002 -> 0002      Type: A       pbc.gov.cn -> ?
0180s780ms Response from 10.3.9.44      ID: 0002 <- 0002      Type: 6       pbc.gov.cn <- notA
0180s785ms Query    from 10.128.201.239 ID: 0003 -> 0003      Type: AAAA    pbc.gov.cn -> ?
0180s789ms Response from 10.3.9.44      ID: 0003 <- 0003      Type: 6       pbc.gov.cn <- notA
0180s795ms Query    from 10.128.201.239 ID: 0004 -> 0004      Type: A       pbc.gov.cn -> ?
0180s799ms Response from 10.3.9.44      ID: 0004 <- 0004      Type: 6       pbc.gov.cn <- notA
0180s804ms Query    from 10.128.201.239 ID: 0005 -> 0005      Type: AAAA    pbc.gov.cn -> ?
0180s809ms Response from 10.3.9.44      ID: 0005 <- 0005      Type: 6       pbc.gov.cn <- notA
0180s846ms Query    from 10.128.201.239 ID: 0001 -> 0001      Type: 12      165.224.128.10.in-addr.arpa -> ?
0180s849ms Response from 10.3.9.44      ID: 0001 <- 0001      Type: Null    165.224.128.10.in-addr.arpa <- No such domain
0180s856ms Query    from 10.128.201.239 ID: 0002 -> 0002      Type: A       xinhuanet.com -> ?
0180s860ms Response from 10.3.9.44      ID: 0002 <- 0002      Type: A       xinhuanet.com <- 202.108.119.194
0180s868ms Query    from 10.128.201.239 ID: 0003 -> 0003      Type: AAAA    xinhuanet.com -> ?
0180s876ms Response from 10.3.9.44      ID: 0003 <- 0003      Type: 6       xinhuanet.com <- notA
0180s913ms Query    from 10.128.201.239 ID: 0001 -> 0001      Type: 12      165.224.128.10.in-addr.arpa -> ?
0180s917ms Response from 10.3.9.44      ID: 0001 <- 0001      Type: Null    165.224.128.10.in-addr.arpa <- No such domain
0180s923ms Query    from 10.128.201.239 ID: 0002 -> 0002      Type: A       cnsa.gov.cn -> ?
0180s932ms Response from 10.3.9.44      ID: 0002 <- 0002      Type: 6       cnsa.gov.cn <- notA
```

baidu.com
qq.com
alibaba.com
163.com
sina.com.cn
taobao.com

weibo.com
zhihu.com
jd.com
sohu.com
boc.cn
10086.cn
chinaunicom.com
chinatelecom.com.cn
pinduoduo.com
douban.com
ke.com
tmall.com
58.com
youku.com
kugou.com
meituan.com
suning.com
vip.com
tuniu.com
maoyan.com
pbc.gov.cn
xinhuanet.com
cnsa.gov.cn
bilibili.com
cnstock.com
12306.cn
caac.gov.cn
music.163.com
sport.gov.cn
waimai.meituan.com
unionpay.com
chnmuseum.cn
mail.163.com
cea.gov.cn
程序面对压力测试表现良好，未发现特殊情况，且均能正确查看。

5 调试中遇到并解决的问题

1. 53 端口
   一开始我们在熟悉 socket 通讯时随便设了一个端口 8888，没有仔细读文档，造成了一些问题。后面使用 53 端口，发现被其他服务进程绑定，无法运行写好的程序。这就需要解绑。
   使用 netstat -aon|findstr "53"查看，发现效果如下，

```
TCP    [::]:53853             [::]:0                 LISTENING       1232
UDP    0.0.0.0:53             *:*                                    3992
UDP    0.0.0.0:5353           *:*                                    13892
UDP    0.0.0.0:5353           *:*                                    13924
```

继续输入命令 tasklist|findstr "3992"，查看对应进程，
发现为

```
svchost.exe                  3992 Services                   0      2,948 K
```

选择使用 taskkill /T /F /PID 3992 指令直接杀死进程，发现这个进程还不一般，

```
svchost.exe                  3992 Services                   0      2,948 K
PS C:\Users\86182> taskkill /T /F /PID 3992
错误: 无法终止 PID 3992 (属于 PID 1232 子进程)的进程。
原因: 拒绝访问。
```

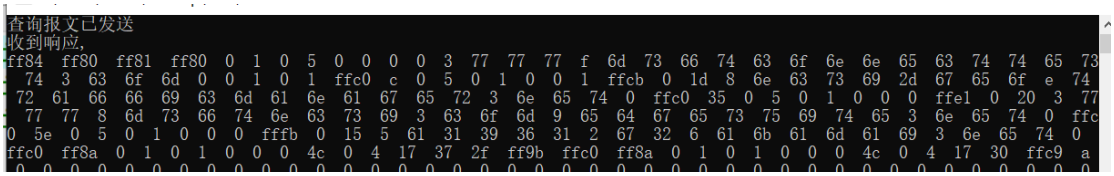
遂查找教程，转发端口信息，解决问题。

2. 主机和程序
   在开始编写程序之前，需要研究主机是如何向我们的程序发送 DNS 报文的，这种 DNS 报文是否包含一些 IP/UDP 包头，遂编写程序创建 socket 监听 8888 号端口，然后使用 nslookup 查找，希望能捕获报文，nslookup -port=<8888> <127.0.0.1>，未能接收到，后经重新阅读后发现需要占用 53 号端口，发送 DNS 请求进行查询。

3. 程序和 DNS
   研究如何向其他 DNS 服务器发送和接受报文的问题，使用 Wireshark 工具找到一个向外界发送的 DNS 报文，然后按 byte 依次赋值构造一个无 IP 包头无 UDP 包头的 DNS 请求（学习 socket 中了解到的）。

```
31      // ?DNS?????
32      char dnsQuery[] = {
33          // 0x00, 0x01,  // ????ID
34          //  0x01, 0x00,  // ??-
35          //  0x00, 0x01,  // ???????
36          // 0x00, 0x00,  // ?s????????
37          // 0x00, 0x00,  // ??E????????
38          // 0x00, 0x00   // ???????????
39          // ... ?????????DNS?????????
40          0x84, 0x80, 0x01, 0x00, 0x00, 0x01,
41          0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
42          0x03, 0x77, 0x77,  (int)99  F, 0x6D,
43          0x73, 0x66, 0x74, 0x63, 0x6F, 0x6E,
44          0x6E, 0x65, 0x63, 0x74, 0x74, 0x65,
45          0x73, 0x74, 0x03, 0x63, 0x6F, 0x6D,
46          0x00, 0x00, 0x01, 0x00, 0x01
47      };
48
```

发送后收到的数据包：



4. 大端小端

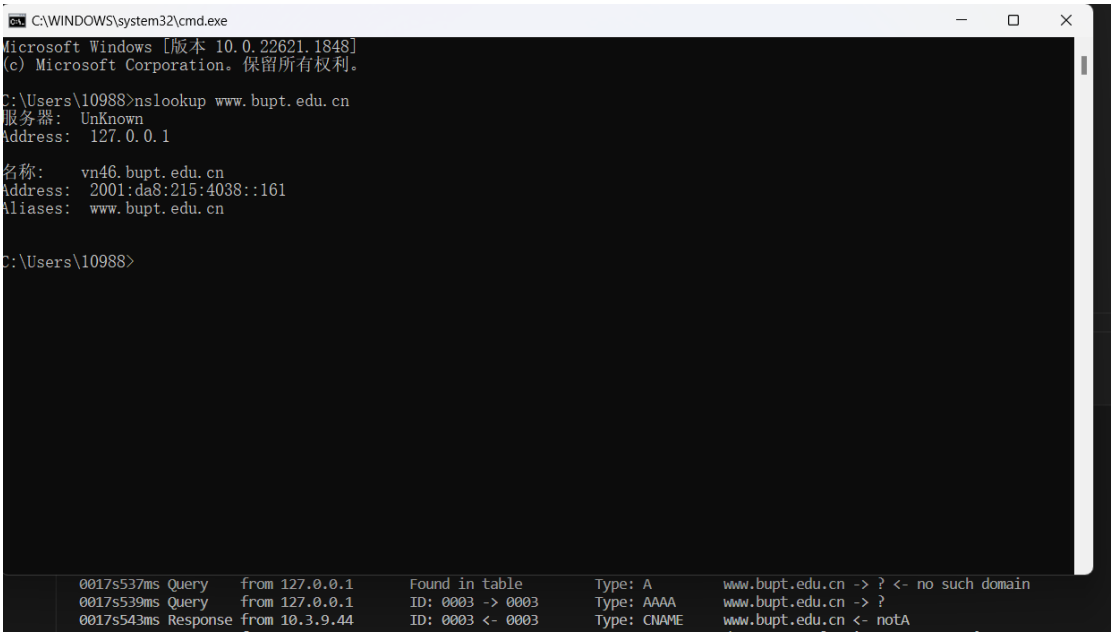
主机使用的多半是小端序，而网络端使用大端，需要 ntohl ntohs htons htonl 来进行转换。

5. 解读与构造报文时的 0xc00c

DNS 报文中记录地址字符串使用的是偏移量，通过这样减少了传输量，构造 answer 的 rr 时使用 0xc00c 来表示此前在 query 中已出现过的域名头指针位置 DNS 应答包中的C00C_dns 响应包_xth21 的博客-CSDN 博客。

6 验收中的发现的问题和解决

1. IPv6 的拦截，一次 nslookup 会发送 A 类型和 AAAA 类型两个 query，而我们此前只考虑了 IPv4 的 0.0.0.0，拦截并回应了 A 类型的 query。



原来因为表中只有 0.0.0.0，错误地只判断了 TYPE_A。AAAA 被转发到外部 DNS 服务器。然后外部 DNS 服务器的 response 被转发回去了。所以还能打开这个网页。

在老师验收时的指点帮助下，我们解决了这个问题，使拦截域名功能更完善了。

解决这个问题，代码只需改变 table 和 cache 的下面这一行即可。两次发回的报文的 Answer RR 是一样构造的，返回的都是 A 类型(IPv4)的 0.0.0.0。

```c
int find_table( struct QUESTION* q, struct RR* rr) {
    if (q->qtype != TYPE_A && q->qtype != TYPE_AAAA) return 0;
    //if (q->qtype != TYPE_A) return 0;


    int left = 0;
    int right = table->count - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        int cmpResult = strcmp(table->records[mid].domain, q->qname);

        if (cmpResult == 0) {
            rr->type = htons(TYPE_A);
            rr->class = htons(q->qclass);
            rr->ttl = htons(3600);
```

```
C:\Users\10988>nslookup www.bupt.edu.cn
服务器:  UnKnown
Address:  127.0.0.1

*** UnKnown 找不到 www.bupt.edu.cn: Non-existent domain

C:\Users\10988>nslookup www.bupt.edu.cn
服务器:  UnKnown
Address:  127.0.0.1

*** UnKnown 找不到 www.bupt.edu.cn: Non-existent domain

C:\Users\10988>
  0003s485ms Query    from 127.0.0.1      Found in table      Type: A       www.bupt.edu.cn -> ? <- no such domain
  0003s486ms Query    from 127.0.0.1      Found in cache      Type: AAAA    www.bupt.edu.cn -> ? <- no such domain
  0003s487ms Query    from 127.0.0.1      Found in cache      Type: A       www.bupt.edu.cn -> ? <- no such domain
  0003s488ms Query    from 127.0.0.1      Found in cache      Type: AAAA    www.bupt.edu.cn -> ? <- no such domain
  0004s048ms Query    from 127.0.0.1      ID: 0001 -> 0001    Type: 12      1.0.0.127.in-addr.arpa -> ?
  0004s050ms Response from 10.3.9.44      ID: 0001 <- 0001    Type: Null    1.0.0.127.in-addr.arpa <- No such domain
  0004s058ms Query    from 127.0.0.1      Found in cache      Type: A       www.bupt.edu.cn -> ? <- no such domain
  0004s059ms Query    from 127.0.0.1      Found in cache      Type: AAAA    www.bupt.edu.cn -> ? <- no such domain
  0004s060ms Query    from 127.0.0.1      Found in cache      Type: A       www.bupt.edu.cn -> ? <- no such domain
  0004s061ms Query    from 127.0.0.1      Found in cache      Type: AAAA    www.bupt.edu.cn -> ? <- no such domain
```

可以看到，第一次在 table 中拦截，后续 7 次在 cache 中拦截。

2. 请求的存储，对于缓冲区的分配问题，每次存入缓冲区时会一并存入一个时间，可以根据这个时间判断这个缓冲数据是否过期，好处是不用再维持很多的计时器，或是每隔一段时间进行一次遍历。这样实现可以达到 O(1)的时间复杂度，验收时我们可能没解释的非常清楚。

3. 对于拥塞情况，我们直接采取了 exit，其实可以随机丢一些，这个也不复杂。直接调用普通的随机数就能实现。

4. 为什么只有两个人？我们两个人加上邢智恺同学完成了计算机网络实验一，链路层重传协议，本来计划到计算机网络课程设计中依然沿用三人小组，但是邢智恺同学未选修，再找人也比较迟了，只能二人一队。虽然是两人，我们两人也能很好的完成分工和合作，提升了团队协作的能力。

7 总结和心得体会

首先是一开始不知道怎么做，所以最初就在设计上存在一些纰漏。因此有一些不完善之处，比如验收时的两个问题。这两个问题其实挺好解决，但是因为文档不详细和思考不周密，没有做出来。也存在一些可能的可改进之处：1. table 和 cache 可以直接存 RR 而非 IPv4 地址。虽然增加了时空复杂度，但这样转发时不用自己构造 RR，且保留了其它非 A 类型的 Answer，更好确保了转发的完整性；2. 改进为多线程处理也许更好，现在 while(1)的方式必须执行完一个 while 流程再收一个新的 DNS 报文；3. 可以用 Trie 树存储域名，优化搜索的时间复杂度。

完成计算机网络课程设计还是很有收获的，虽然花费了很多的时间，但是和以往的课程设计实践相比，第一次完成了一个“工程化“的项目，做出了一个可供使用的 DNS 中继服务器，实在是一份难得的体验。