**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** | **高占春** |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | 简要介绍课程设计的主要内容，包括课程设计教学目的、基本内容、实验方法和团队分工等    此次课程设计的主要目的在于熟悉计算机网络中的 DNS 协议，动手实践编写程序实现 DNS 中继程序。 此次课程设计使用 C++ 语言在 Windows 10 系统上实现了一个 DNS 中继服务器程序。该程序实现了拦截不良网站、使用本地域名 - IP映射表响应本地进程的域名查询请求以及将本地进程的域名查询请求中继至外部 DNS 服务器等功能。  此次课程设计的程序设计与编写、文档撰写工作由\*\*\*独立完成。 | | | | | |
| **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。

**一、系统的功能设计**

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：

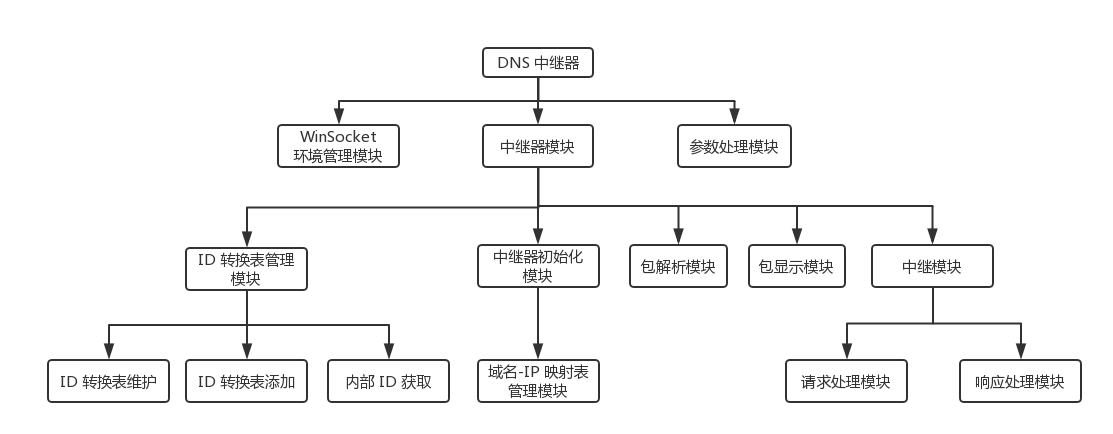
a) ip地址0.0.0.0，则向客户端返回“域名不存在”的报错消息（不 良网站拦截功能）

b) 普通IP地址，则向客户端返回该地址（服务器功能）

c) 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同时查询的情况。

**二、模块划分**

**1. 模块划分图**

**2. 模块说明**

|  |  |
| --- | --- |
| **模块名** | **功能** |
| WinSocket环境管理模块 | 负责 WinSocket 环境的初始化及清理，为后续 Socket连接提供支持。 |
| 参数处理模块 | 处理输入的参数，生成相应配置的 DNS 中继器。 |
| 中继器模块 | DNS 中继器的核心功能模块，完成接收 DNS 请求及对 DNS 请求的响应处理等功能。 |
| ID 转换表管理模块 | 完成中继转发处理DNS 请求及其响应中消息 ID 的转换功能。 |
| ID 转换表维护 | 按照时间戳删除 ID 转换表中超时的转换映射。 |
| ID 转换表添加 | 添加内外 ID 的转换关系，并保存发送 DNS 请求的进程的地址信息。 |
| 获取内部 ID | 完成外部 ID 向内部 ID 的转换，实现中继外部 DNS 响应的功能。 |
| 中继器初始化模块 | 初始化中继器模块，包括导入域名-IP映射表等。 |
| 域名-IP映射表管理模块 | 按照输入文件名从文件中读入域名-IP映射关系并进行存储。 |
| 中继模块 | 对本机内部进程的 DNS 请求进行处理，并将外部 DNS 服务器的请求中继转发给本机内部进程。 |
| 请求处理模块 | 对本机内部进程的 DNS 请求进行处理。对本机域名-IP映射表中存在的域名构造响应包后直接返回，对本机域名-IP映射表中不存在的域名或者 IPV6 请求发送给外部 DNS 服务器进行中继。 |
| 响应处理模块 | 对外部 DNS 服务器发回的响应进行处理。 |
| 包解析模块 | 对接收到的 DNS 请求及相应进行解析，以便后续处理。 |
| 包显示模块 | 按照不同的调试等级对包进行显示。 |

**3. 主要数据结构**

|  |  |  |
| --- | --- | --- |
| **数据结构名称** | **数据结构** | **说明** |
| \_header | struct \_header {  unsigned short ID;  unsigned short QR;  unsigned short OPCODE;  unsigned short AA;  unsigned short TC;  unsigned short RD;  unsigned short RA;  unsigned short Z;  unsigned short RCODE;  unsigned short QDCOUNT;  unsigned short ANCOUNT;  unsigned short NSCOUNT;  unsigned short ARCOUNT;  }; | 按照 DNS 协议中规定的 DNS 分组头部格式保存 DNS 分组头部。包括分组序号、请求/响应表示、各分部计数等信息。 |
| \_query | struct \_query {  std::string QNAME;  unsigned short QTYPE;  unsigned short QCLASS;  }; | 按照 DNS 协议中规定的查询段的格式保存查询内容。包括查询域名、类型及分类等。 |
| \_RR | struct \_RR {  std::string NAME;  unsigned short TYPE;  unsigned short CLASS;  unsigned int TTL;  unsigned short RELENGTH;  std::string RDATA;  }; | 按照 DNS 协议中规定的资源记录的格式保存资源记录。包括资源名称、类别、类型、TTL、资源长度及资源数据段等。 |
| dnsPacket | struct dnsPacket {  \_header header;  std::list<\_query> query;  std::list<\_RR> answer;  std::list<\_RR> authority;  std::list<\_RR> additional;  int length;  }; | 保存反序列化后的DNS 分组。 |
| innerID | struct innerID {  unsigned short inID;  sockaddr\_storage addr;  clock\_t timeStamp;  }; | 内部 ID。其中包括发出 DNS 请求的序列号、其 socket 地址以及收到请求的时间戳。 |

**4. 面向对象设计**

**(1) mySocket 类**

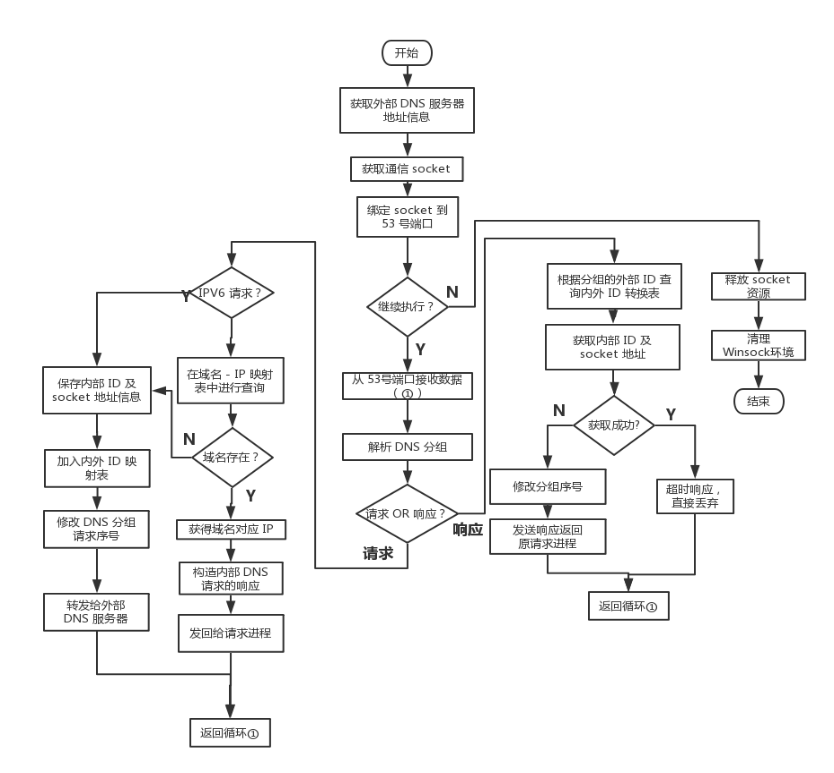
|  |  |  |  |
| --- | --- | --- | --- |
| **类名** | mySocket | | |
| **说明** | 作为使用 Socket 进行通信的基类，负责初始化 Winsock环境及运行结束后清理 Winsock环境等 | | |
| **成员变量** | | | |
| **变量名** | **类型** | | **说明** |
| 无 | | | |
|  | | | |
| **函数名** | **返回类型** | **参数** | **说明** |
| mySocket | 无 | void | mySocket 类的构造函数，调用 initialize 函数进行环境的初始化 |
| initialize | bool | void | 初始化 Winsock 环境，初始化成功返回 true，失败则返回 false。 |

**(2) convertTable 类**

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** | convertTable | | |
| **说明** | 负责管理及维护内部及外部 DNS 分组的 ID 转换表 | | |
| **成员变量** | | | |
| **变量名** | **类型** | | **说明** |
| nextOutID | unsigned short | | 下一个外部 ID。在添加转换内外 ID 转换关系时，分配给当前内部 ID 的外部 ID。 |
| idTable | std::unordered\_map<unsigned short, struct innerID> | | ID 转换表。格式如下：  (外部 ID ，内部 ID ，socket 地址，时间戳) |
|  | | | |
| **函数名** | **返回类型** | **参数** | **说明** |
| convertTable | 无 | void | convertTable 的构造函数，负责初始化  nextOutID。 |
| maintainTable | void | void | 维护转换表，删除过时条目。 |
| addTable | void | unsigned short & outID,  struct innerID & inner | 向转换表中加入新的转换关系。 |
| fetchInnerID | bool | const unsigned short & outID, struct innerID & inner | 根据外部 ID 获取内部ID及其 socket 地址等信息。 |

**(1) dnsRelayer 类**

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** | dnsRelayer | | |
| **说明** | DNS 中继器。负责完成 DNS 中继功能。 | | |
| **成员变量** | | | |
| **变量名** | **类型** | | **说明** |
| dnsTable | std::unordered\_map<std::string, std::string> | | 为了提高查询效率，使用关联容器存储域名-IP映射表 |
| idTable | convertTable | | 内外消息 ID 转换表 |
| sock | SOCKET | | 通信 socket |
| debug | int | | Debug 等级 |
| servAddr | sockaddr | | 外部 DNS 服务器地址 |
| servAddrLen | size\_t | | 外部 DNS 服务器地址长度 |
| count | int | | 已处理的 DNS 查询请求数目 |
|  | | | |
| **函数名** | **返回类型** | **参数** | **说明** |
| dnsRelayer | 无 | int level,  const string ip,  const string file | 构造函数，初始化 dnsRelayer 环境 |
| relay | void | void | 中继函数 |
| addrToIP | std::string | const sockaddr\_storage & addr | 转换 socket 地址为字符串 |
| response | void | char \* recvBuf, const dnsPacket & packet, sockaddr\_storage & theirAddr | 响应处理 DNS 分组 |
| displayPacket | void | const dnsPacket & packet | 显示 DNS 分组 |
| parsePacket | dnsPacket | const char \* buf, const int bytes | 解析 DNS 分组 |
| loadTable | bool | const std::string file | 从外部文件载入域名 - IP映射表 |

**三、软件流程图**

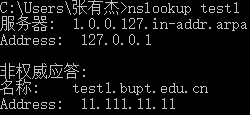
**四、测试用例以及运行结果**

1. 测试 dnsrelay.txt 中域名

测试用例： test1

用例说明： test1 是在 dnsrelay.txt 中已经写好的域名。

结果：



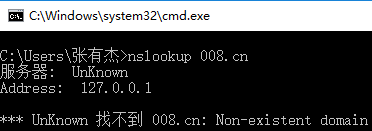
可以看出，我们编写的 DNS 中继器成功将本机 dnsrelay.txt 中对应域名的 IP 地址返还给请求进程。

2. 测试被屏蔽网址

测试用例： 008.cn

用例说明： 008.cn 是在 dnsrelay.txt 中设置为要屏蔽的域名。

结果：



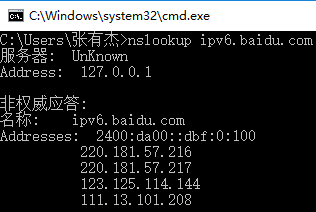
可以看出，我们编写的 DNS 中继器成功屏蔽本机进程对 008.cn 的域名查询请求。

3. 处理 IPv6 请求

测试用例： ipv6.baidu.com

用例说明： 对 ipv6.baidu.com 的 IPv6 地址进行请求。

结果：



可以看出，我们编写的 DNS 中继器成功处理对相应域名的 IPv6地址的查询请求，并正确返回 IPv6 地址。

4. 测试 DNS 请求中继

测试用例： www.baidu.com

用例说明： 将 对[www.baidu.com](http://www.baidu.com) 的域名查询请求转发至外部 DNS 服务器。

结果：



可以看出，我们编写的 DNS 中继器成功将在本地域名 - IP 映射表中不存在的域名的查询请求转发给了外部 DNS 服务器，并将外部 DNS 服务器的响应成功转发给本机请求进程。

**五、调试中遇到并解决的问题**

1. recvfrom 函数调用报错

(1)问题描述：

在程序编写初步完成进行调试时，通过捕捉函数调用出错后抛出的异常的方式发现了程序在 recvfrom 函数的调用偶尔会返回 -1 的错误。

(2)解决方式：

首先查询 MSDN 得知，在 recvfrom 函数返回错误值后调用 WSAGetLastError 函数可以得到最近一次错误发生时的错误代码。将其加入代码段中，在 recvfrom 函数返回 -1 的时候得到错误代码 #10054。

查询 MSDN 上的错误代码说明后发现该代码表示 connection reset by peer。即连接的另一端主机重置了连接。但明显这个解释并不成立，因为我们在该中继器的编写中使用的是无连接的 UDP 通信。再次上网查询后，发现这其实是 Winsocket 的一个bug，在实际使用中可以直接忽略该错误。（详情可见：<https://support.microsoft.com/en-us/help/263823/winsock-recvfrom-now-returns-wsaeconnreset-instead-of-blocking-or-timing-out>）

2. 对 IPv6 请求的处理

(1)问题描述：

在完成中继器初步的调试时，我意识到中继器并没有处理一次 nslookup 中同时产生的 IPv6 请求。

(2)解决方式：

由于本机的 dnsrelay.txt 中并不存在 IPV6 地址，所以要处理 IPv6 的请求我们只能将该请求转发给外部 DNS 服务器。

因此在处理外部 DNS 请求时我们要判断当前请求是否为 IPv6 请求，即 QTYPE 是否为 AAAA。如果是则将其转发给外部 DNS 服务器，否则进一步处理。

3. 使用位运算拼接高低字节出错

(1)问题描述：

中继器在运行过程中使用 nslookup 以及 ping 命令进行测试时总会出现超时的情况。

(2)解决方式：

临时增加代码，将所有的请求与响应都发送一遍给外部 DNS服务器（因为发送给本机进程的 DNS 响应并未经过网卡，所以不能够被捕捉到）。使用 WireShark 软件对这些发送的结果进行抓包，观察我们自己构造和转发的 DNS 请求和响应是否能够被 WireShark 软件成功解析。

观察所有外部 DNS 服务器的响应发现，它们的 DNS 分组 ID 经过转换后都为 0xff 开头。但是观察发现我们的内外 ID 转换表中的外部 ID 及内部 ID 值基本上都不是以 0xff 开头的。所以推测可能是代码中对分组 ID 进行解析及填写的部分发生了错误。进一步调试代码发现，在解析及填写分组 ID 时候代码进行的位运算错误地将每一个分组 ID 的高八位都置为了 1。更正拼接 ID 高低字节的位运算代码后，程序正常工作，问题解决。

4. 处理内外 ID 转换

(1)问题描述：

中继器可能同时处理本机多个进程发来的 DNS 查询请求，将它们转发给外部 DNS 服务器之后接收到外部 DNS 服务器的响应时需要区分不同的进程的请求，并将响应转发回对应进程。因此，需要设计一种机制，完成这样的区分工作。

(2)解决方式：

设计出一个转换表类，专门负责内外DNS 分组 ID 的转换。当收到本机进程的 DNS 请求时，为其分配一个外部 ID，将内外 ID 的映射关系加入转换表，并保存本机请求进程的地址。在修改 DNS 分组的 ID 为该外部 ID 后，将其转发给外部 DNS 服务器。当收到外部 DNS 服务器发回的响应时，以该响应的 ID 查询转换表获取其内部 ID 及其请求进程地址。以内部 ID 修改该 DNS 响应的 ID后，将分组转发回请求进程。

**六、课程设计工作总结**

本次课程设计中，我完成了一个 DNS 中继器，实现对本机进程的 DNS 查询请求的响应与转发。

本次课设难度适中，总体耗时两周左右。在本次课设中遇到了许多问题，但都成功地解决了，也因此有了很多收获，主要如下：

在进行网络相关的编程时，使用 Wireshark等类似的抓包工具可以帮助我们调试程序，理解网络协议里路程。同时，阅读 RFC 文档，对我完成这次课设也有着很大的帮助。另外，由于本次课设是使用 C++ 在 Windows 系统下完成的，许多遇到的问题比如 Winsock API的使用等问题都需要通过查询 MSDN 解决。熟练地使用 MSDN 解决问题是本次课设中我最主要的收获。让我感受最深的时，在碰到问题时应该耐下心，一步步地缩小范围，排查原因。所有的 BUG 背后都是有原因的。只要耐下心来，最后一定能解决问题。

通过本次课程设计，我认真完成了每一行代码的编写，锻炼了自己的编程能力，同时熟悉了 DNS 协议的相关细节，对 DNS 协议的工作过程有了进一步的体会，相信此次课设为今后网络相关的编程有很大的帮助。