**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** | **张乙然** |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
| **2023211313** |  | **2023210887** | | **孙明皓** |  | |
| **2023211313** |  | **2023210881** | | **杨睿博** |  | |
| **2023211313** |  | **2023210911** | | **李志敏** |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | **1. 课程设计教学目的**  本次课程设计旨在使学生深入理解 DNS协议的工作原理，掌握网络编程的基本技能，特别是 UDP 协议的应用。通过亲自动手实现一个 DNS 中继服务器，学生将能够：   * **理解 DNS 协议**: 熟悉 DNS 消息的格式、查询与响应流程，以及域名解析的核心机制。 * **掌握套接字编程**: 学习并应用 UDP 套接字进行数据收发，理解网络字节序与主机字节序的转换。 * **实践数据结构与算法**: 运用哈希表实现高效的域名查找和存储，结合链表实现 LRU缓存淘汰策略。   **2. 课程设计基本内容**  本次课程设计的主要内容是实现一个功能相对完整的 DNS 中继服务器（或称 DNS 代理）。该服务器将作为客户端和上游 DNS 服务器之间的桥梁，提供以下核心功能：   * **DNS 消息解析与构建**: 能够解析传入的 DNS 查询请求，并根据解析结果构建正确的 DNS 响应消息。 * **本地 DNS 解析**: 支持从本地配置文件 (dnsrelay.txt) 加载静态的域名到 IP 地址映射，优先响应这些本地配置的查询。 * **DNS 缓存机制**:   + **缓存存储**: 使用哈希表存储 DNS 响应中的域名和 IP 地址映射。   + **LRU 淘汰策略**: 当缓存空间达到上限时，采用 LRU (Least Recently Used) 策略淘汰最久未使用的缓存条目，以保证缓存的有效性。   + **缓存查找与更新**: 对于缓存命中的请求，直接返回缓存中的数据；同时更新命中条目的“最近使用”状态。 * **请求转发与响应回传**: 对于本地无法解析（既不在本地表也不在缓存中）的 DNS 请求，能够将其透明地转发到预设的上游公共 DNS 服务器，并将收到的上游响应回传给原始客户端。 * **事务 ID 管理**: 维护一个 DNS 事务 ID 映射表，以正确地将上游服务器的响应关联到对应的客户端请求，并支持超时过期清理。 * **I/O 多路复用**: 使用 select() 实现基于阻塞 socket 的 I/O 多路复用机制，轮询监听本地 socket 与上游 DNS socket 的读事件，避免传统阻塞模型在并发处理多个客户端和上游响应时的性能瓶颈。 * **日志与调试**: 提供日志记录功能和调试输出，方便程序运行状态的监控和问题排查。   **3. 实验方法**  本次课程设计主要采用 C 语言进行实现，并利用 Winsock 库进行网络编程。   * **模块化开发**: 将整个系统划分为多个模块，如 DNS 协议解析模块 (protocol.c)、本地表模块 (table.c)、缓存模块 (cache.c )、网络通信与主模块（main.c/server.c）、工具与调试模块（util.c）。 * **增量式开发**:   **第一步**：实现基础的 DNS 协议解析和本地表查询，确保服务器能正确响应本地已知域名。  **第二步**：增加中继功能，实现本地查不到时自动转发到上游DNS，并正确处理并发请求的 ID 映射。  **第三步**：实现超时检测与处理和ipv4/v6多协议支持，完善功能与健壮性。  **第四步**：引入缓存机制，优先从缓存查找，未命中再查本地表或中继，提升性能。缓存采用哈希表+LRU顺序管理，结合TTL自动失效。  **第五步**：完善调试与日志输出，便于测试和维护。   * **调试与测试**: 利用 print\_debug\_info、print\_hex 等调试函数输出关键流程和数据，辅助定位问题。   通过 nslookup、dig 等工具对服务器进行功能测试，验证本地解析、拦截、中继、缓存、超时等各项功能的正确性和稳定性。  **4. 团队分工** ：  孙明皓   * + **主流程控制**: 搭建一个DNS服务器从启动，分配socket到终止，释放内存的顺序执行流程（main.c, server.c）   + **协议与报文处理**: 编写要转发的DNS 报文的构造过程的相关代码（protocol.c）   + **本地表与数据结构**: 负责本地域名-IP映射表的设计与实现（table.c）和 uthash 的集成和使用   李志敏：   * + **网络通信**: 实现向上游服务器转发功能，包括ID映射表、中继转发、超时处理等（server.c）   杨睿博：   * + **缓存与LRU管理**: 负责LRU缓存模块的设计与实现（cache.c）   + **用户接口与日志**: 编写处理命令行输入，实现多级调试信息的工具函数（util.c） | | | | | |
| **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。

**DNS服务器课程设计实验报告**

**1. 设计环境**

* 操作系统：Windows 10
* 编译器：GCC 9.4.0 / MinGW-w64
* 开发工具：VSCode
* 测试工具：nslookup, dig
* 主要依赖：uthash.h（哈希表实现）

**2. 系统的功能设计**

**2.1 功能描述**

* 支持UDP协议的DNS查询与响应
* 本地表（dnsrelay.txt）查找与拦截
* DNS缓存（哈希表+LRU+TTL）
* 本地查不到时自动中继转发到上游DNS服务器
* 支持并发请求的ID映射
* 超时检测与错误响应
* 日志与调试输出

**2.2 RFC1035 报文结构**

* 遵循RFC1035标准，实现DNS报文的头部、问题区、资源记录等结构体定义与解析。



**RFC1035 是什么**

* RFC1035（Request for Comments 1035）是DNS（域名系统）协议的核心标准文档，定义了DNS报文格式、查询/响应流程、资源记录类型等内容。
* 绝大多数DNS服务器和客户端实现都遵循这个标准。

**3. 模块划分**

**3.1 项目结构图**

DNS-server/

├── main.c // 主流程与网络通信

├── server.c/h // 服务器主流程与中继转发

├── protocol.c/h // 协议解析与报文构造

├── table.c/h // 本地表管理

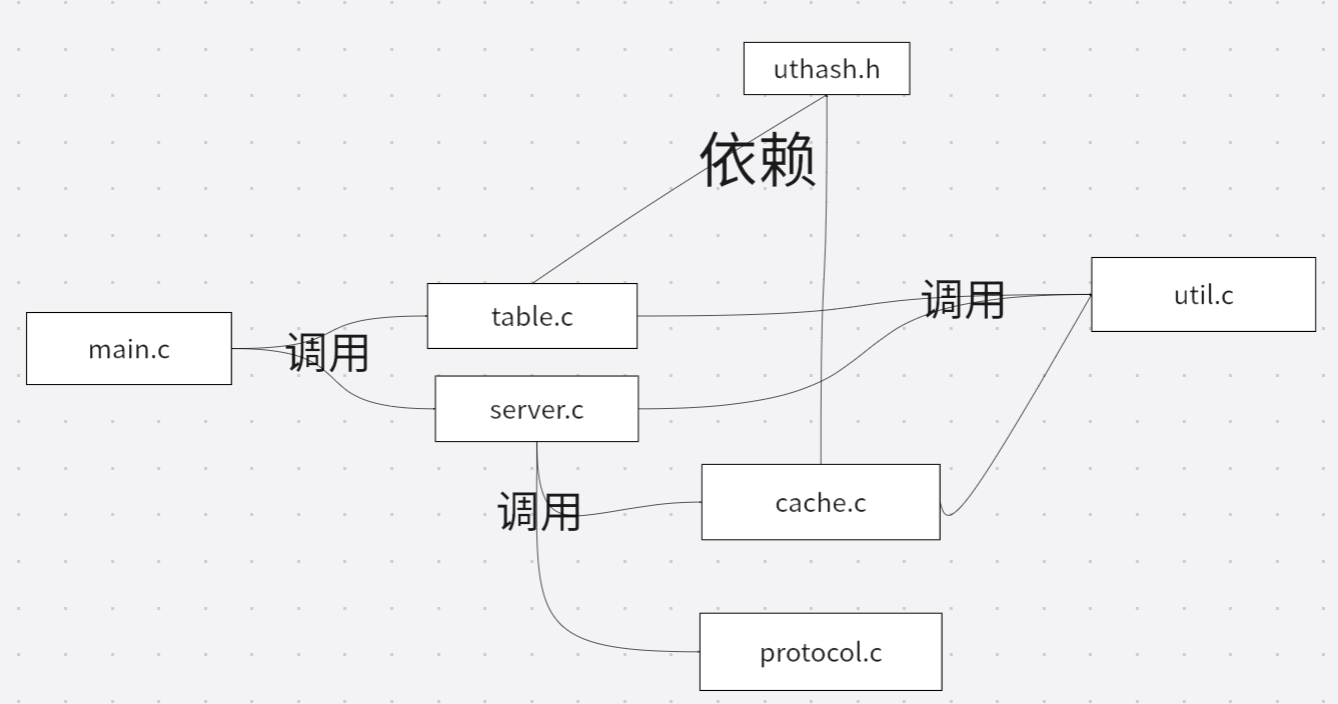
├── cache.c/h // 缓存与LRU管理

├── uthash.h // 哈希表库

├── util.c/h // 工具与调试

└── dnsrelay.txt // 本地表配置

**3.2 模块关系图**

****

**3.3 软件流程图**

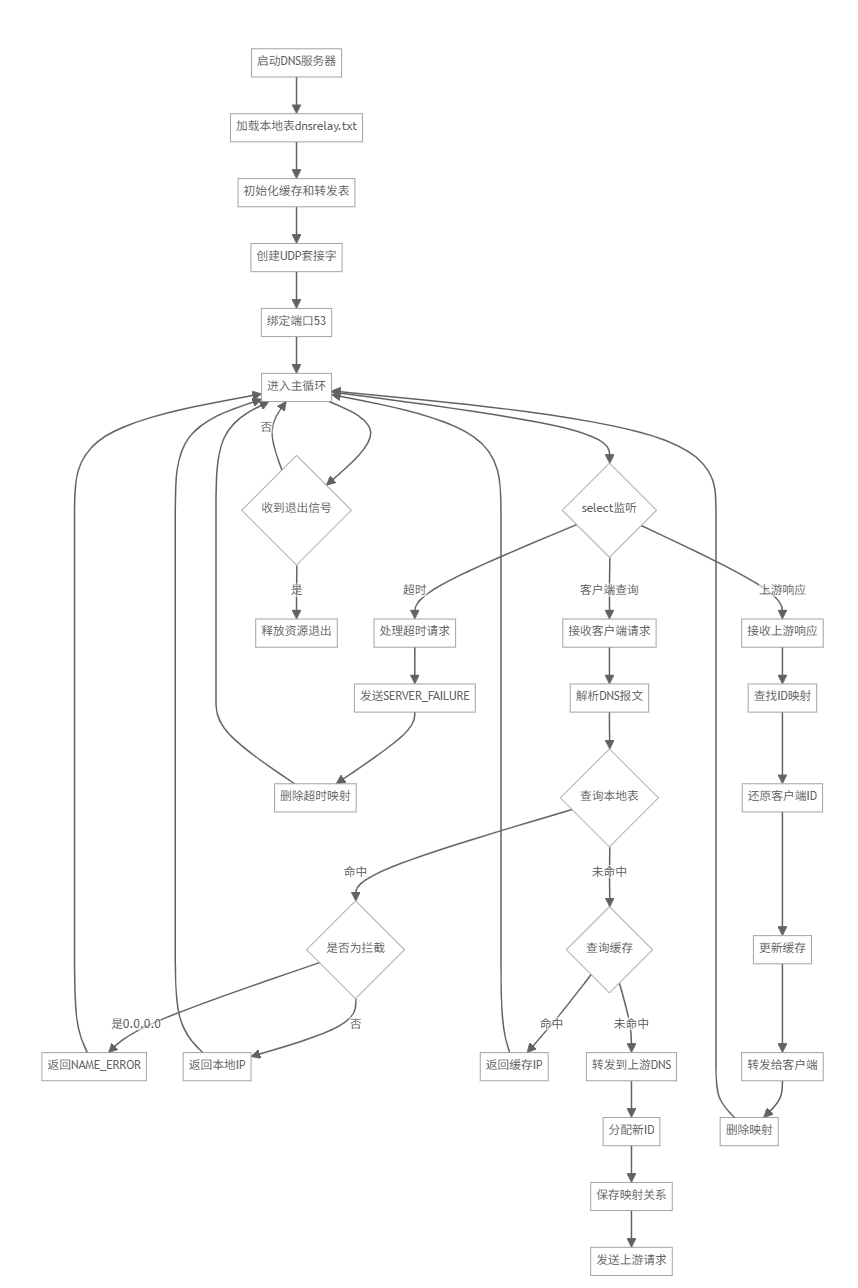
1. 初始化阶段：加载本地表、初始化缓存和转发表、创建套接字
2. 主循环：使用select多路复用监听客户端查询和上游响应（转3或5或6）
3. 查询处理：

先查本地表（命中则返回或拦截）

再查缓存（命中则返回）

最后转发到上游DNS（转4）

1. 中继处理：分配新ID、保存映射、转发请求
2. 响应处理：查找映射、还原ID、更新缓存、转发响应
3. 超时处理：定期检查超时请求，发送错误响应
4. 退出处理：收到信号后释放资源退出



**4. 命令行参数格式**

程序接受命令行参数格式如：

dnsrelay [-d|-dd] [dns-server-ipaddr] [filename]

-d：用调试模式 1（打印查询信息）。

-dd：启用调试模式 2（打印详细调试信息）

dns-server-ipaddr：指定 DNS 服务器的 IP 地址。

filename：指定包含静态 DNS 条目的文件名。

您也可以使用`dnsrelay -h|--helper`命令查看参数详细说明**。**

**5. 代码实现**

**5.1 socket 实现（UDP）**

**5.1.1 宏定义**

1. #define DNS\_PORT 53

运行在 53 端口。

1. #define MAX\_CACHE\_SIZE 1024

DNS 缓存的最大容量，超过后采用LRU淘汰。

1. #define RELAY\_TIMEOUT 5

上游DNS服务器超时时间，单位为秒。

1. #define MAX\_DOMAIN\_LENGTH 256

域名最大长度。

1. #define MAX\_DNS\_PACKET\_SIZE 512

DNS UDP报文最大长度。

**5.1.2 全局变量声明**

1. SOCKET sock = INVALID\_SOCKET;

UDP套接字的文件描述符。初始化为无效值，表示尚未创建。

1. SOCKET upstream\_sock = INVALID\_SOCKET;

上游DNS服务器通信用的UDP套接字。

1. struct sockaddr\_in server\_addr, upstream\_addr;

分别存储本地服务器和上游DNS服务器的地址信息。

1. fd\_set readfds;

用于select多路复用的文件描述符集合。

1. uint8\_t recv\_buffer[MAX\_DNS\_PACKET\_SIZE];

接收数据缓冲区。

1. uint8\_t send\_buffer[MAX\_DNS\_PACKET\_SIZE];

发送数据缓冲区。

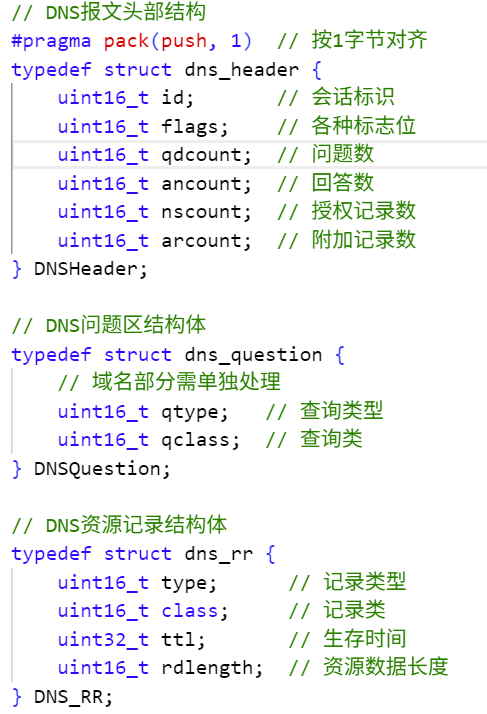
**5.1.3 使用套接字API实现socket**

|  |  |  |
| --- | --- | --- |
| **操作** | **调用** | **关键参数说明** |
| 创建套接字 | socket() | AF\_INET:IPv4协议；  SOCK\_DGRAM: UDP协议 |
| 绑定套接字 | bind() | sin\_family: 地址族；  sin\_addr.s\_addr: 监听IP；  sin\_port: 监听端口（htons转换） |
| 接收数据 | recvfrom() | fd: 套接字描述符；  buffer: 接收缓冲区；  client\_addr: 客户端地址；  addr\_len: 地址长度 |
| 发送数据 | sendto() | response: 发送数据；  response\_len: 长度；  client\_addr: 客户端地址；  addr\_len: 地址长度 |
| 关闭套接字 | closesocket() | sock: 已创建的套接字 |

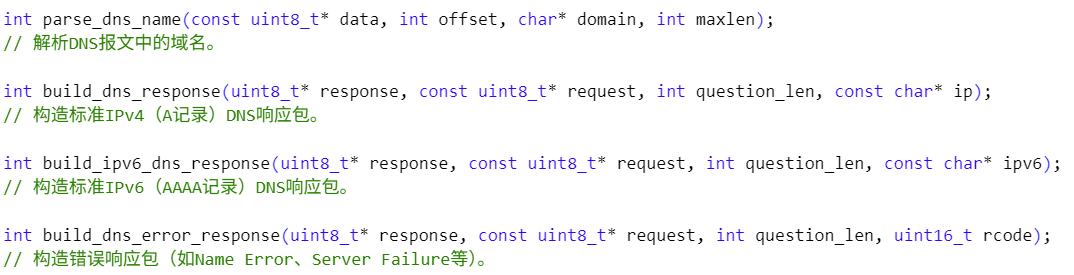
**5.2 协议解析模块**

**5.2.1 结构体定义**

* DNSHeader：DNS报文头部结构，包含ID、标志位、问题数、回答数等字段。
* DNSQuestion：DNS问题区结构，包含查询类型和类。
* DNSResourceRecord：资源记录结构，支持A记录。

****

**5.2.2 主要函数**

****

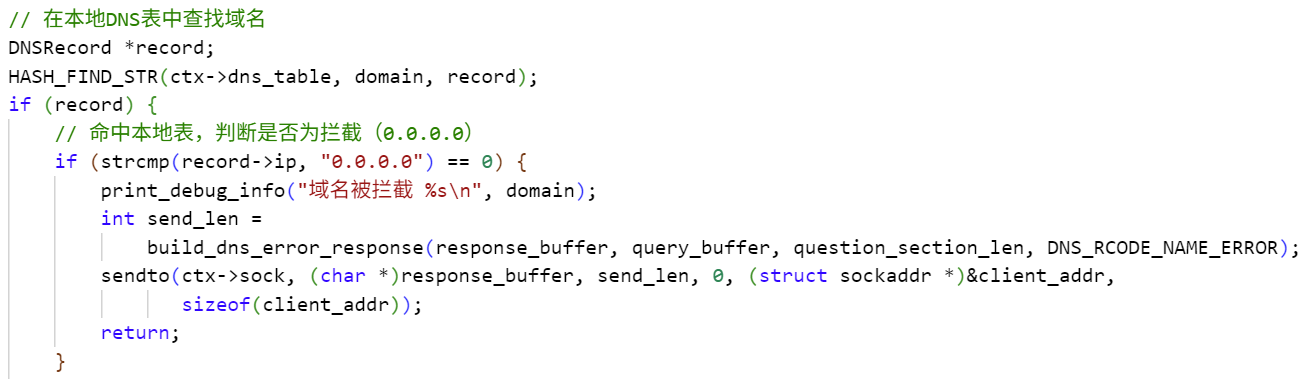
**5.3 本地表模块**

**5.3.1 结构体与加载**

* DNSRecord：包含域名、IP、哈希表句柄。
* load\_dns\_table(const char\* filename, DNSRecord\*\* table);

从dnsrelay.txt加载本地表到哈希表。

**5.3.2 查找与拦截**

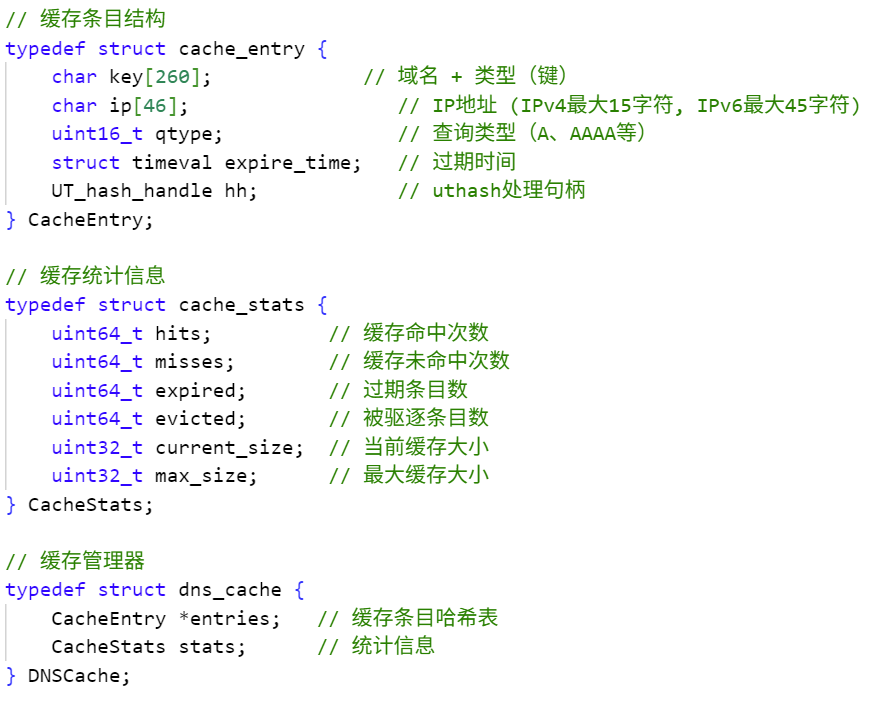
****

* 使用HASH\_FIND\_STR(table, domain, record);查找域名。
* 查到IP为0.0.0.0时，调用build\_dns\_error\_response构造错误响应，实现拦截功能。

**5.4 缓存与LRU管理**

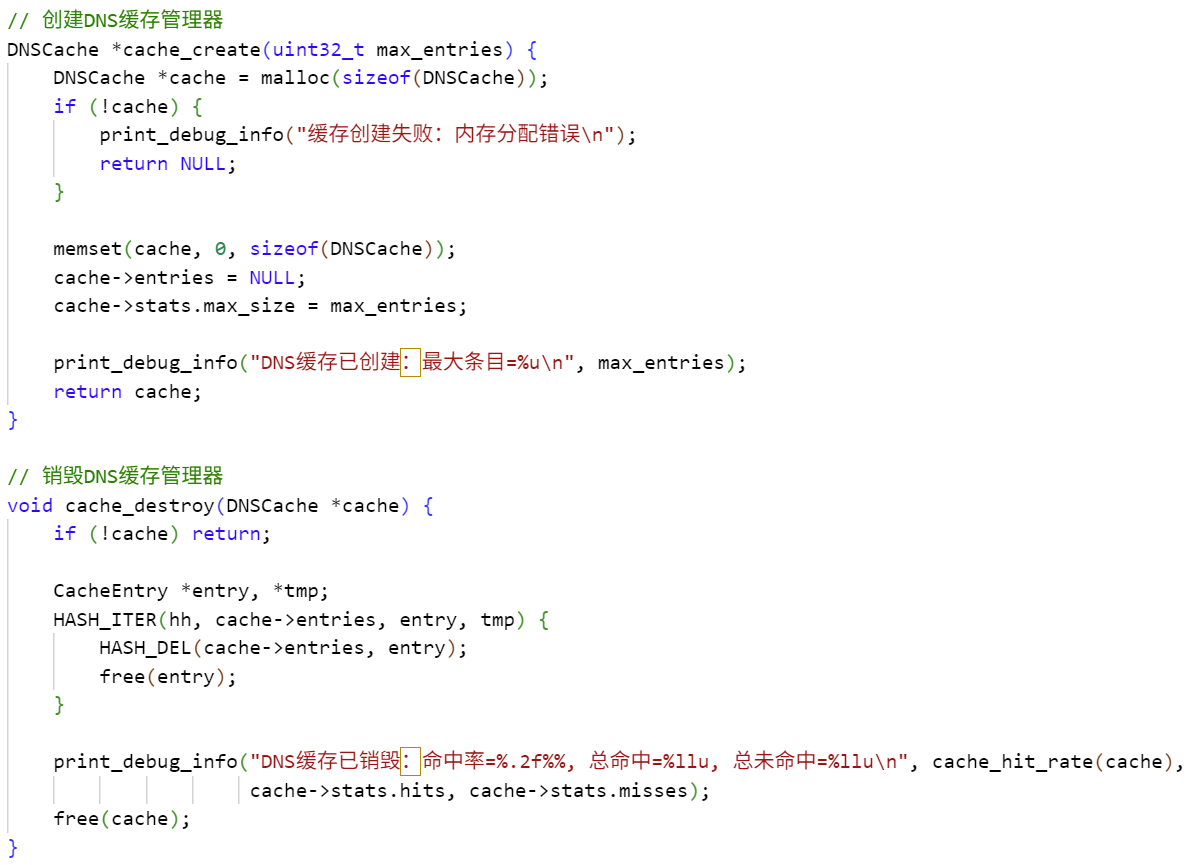
**5.4.1 缓存结构设计**

缓存模块采用哈希表+LRU顺序管理，用于存储最近查询过的域名解析结果



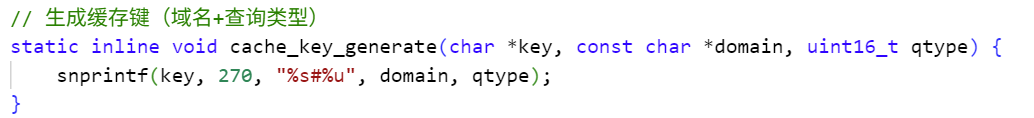
**5.4.2 缓存操作实现**

**1. 缓存创建与销毁**

****

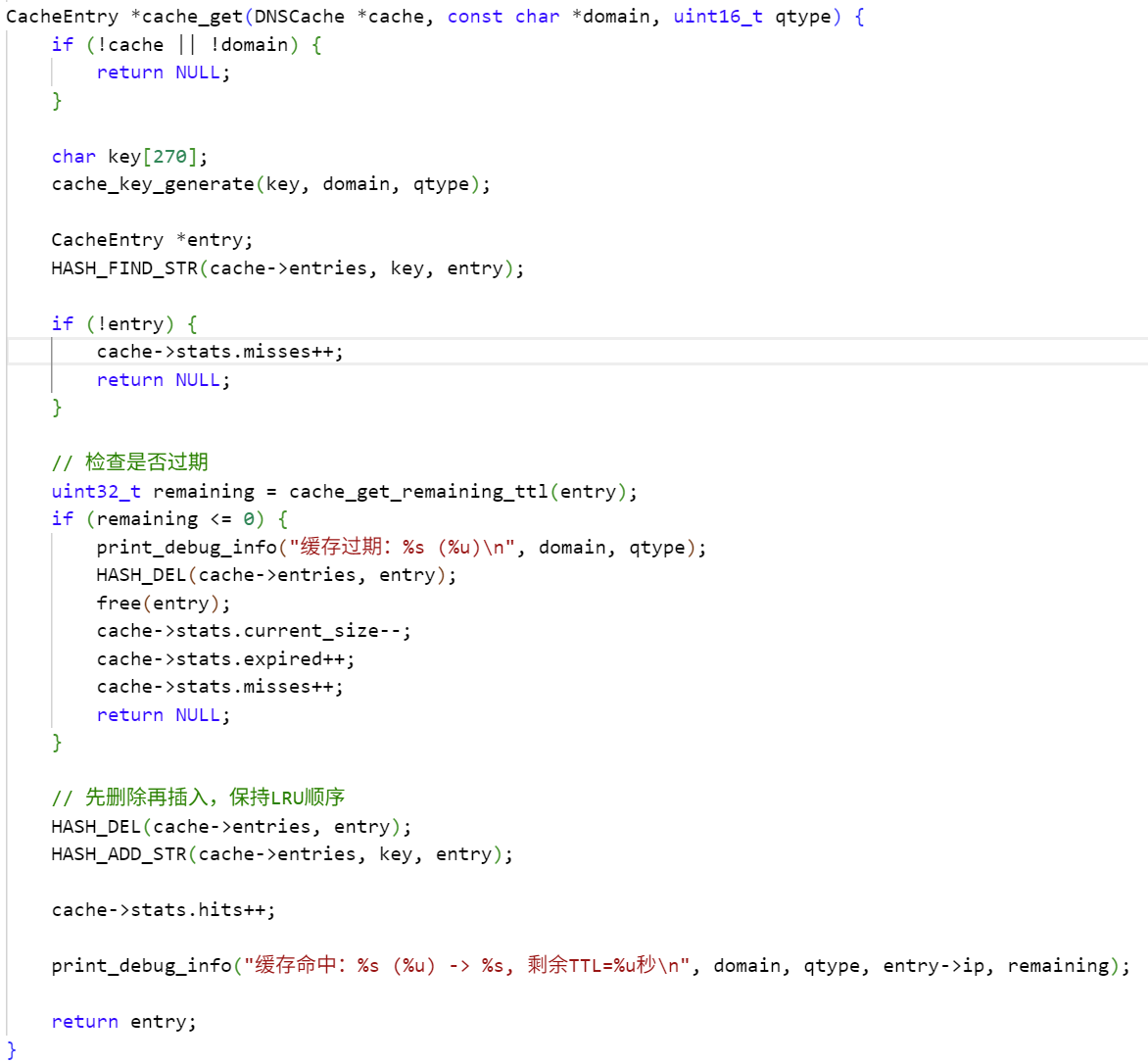
**2. 缓存键生成**

将域名和查询类型组合生成唯一键，格式为"域名#类型"。

****

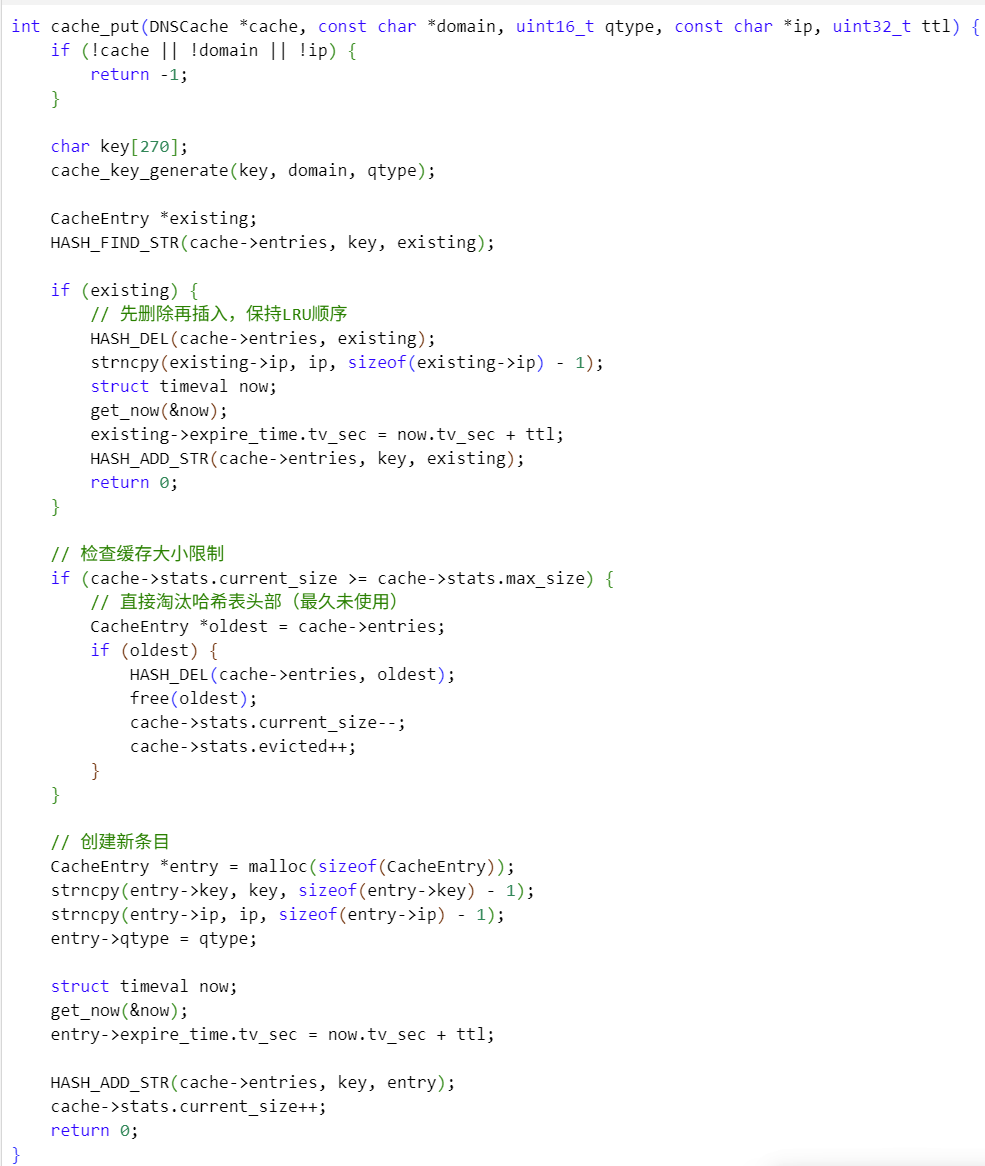
**3. 缓存查询（LRU维护）**

命中时采用"先删后加"策略，将访问的条目移动到哈希表尾部，保证最久未用的项在头部，优先被淘汰

****

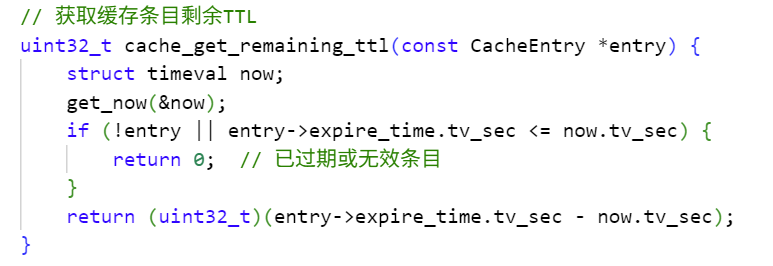
**4. 缓存插入（LRU淘汰）**

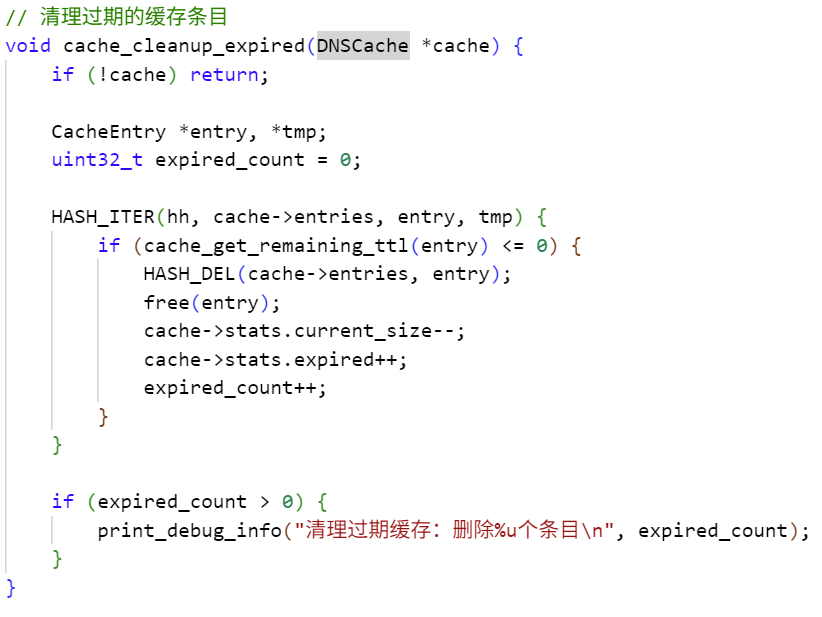
 当缓存满时，直接删除哈希表头部的条目（最久未使用），然后插入新条目到尾部。

****

**5. TTL管理与过期清理**

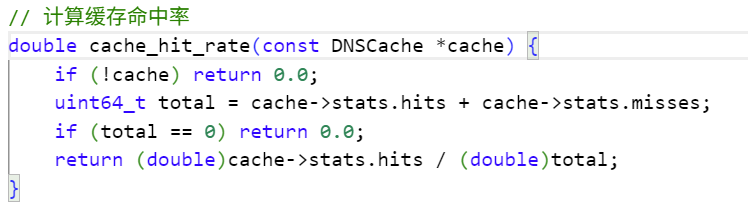
 每个缓存条目记录过期时间，查询时检查剩余TTL，过期则自动删除。定期清理函数遍历所有条目，删除过期项。

****

****

**6. 缓存统计**

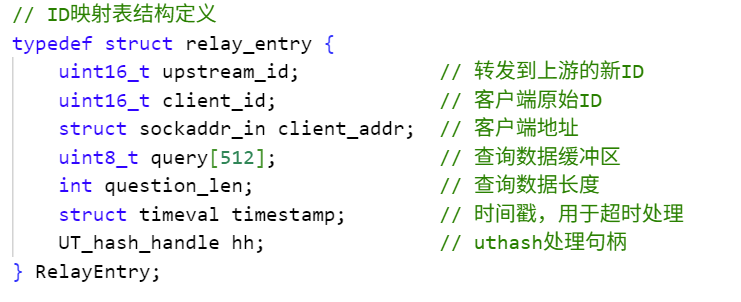
提供命中率、命中次数、未命中次数、过期条目数、被驱逐条目数等统计信息。

****

**5.5 中继与ID映射**

**5.5.1 ID映射结构设计**

当本地表和缓存都未命中时，服务器需要将查询转发到上游DNS服务器。为保证并发请求的正确匹配，设计了ID映射机制：



**5.5.2 转发查询实现**

1. 分配新的RelayEntry结构体
2. 生成唯一的上游ID（递增计数器）
3. 保存客户端原始ID、地址、查询数据等信息
4. 将映射条目添加到哈希表
5. 修改查询包中的ID为新的上游ID
6. 发送到上游DNS服务器



**5.5.3 上游响应处理**

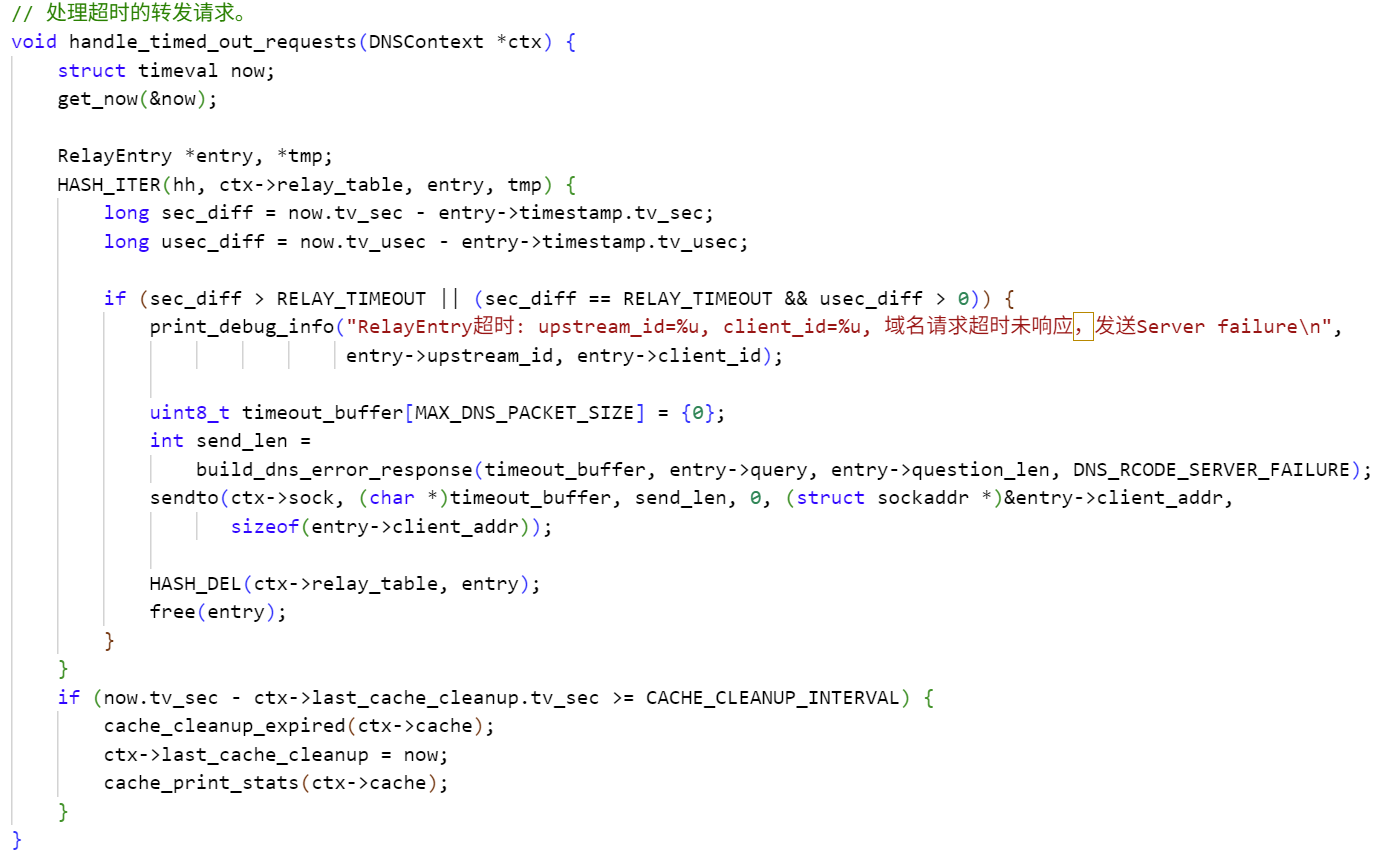
1. 从响应包中提取上游ID
2. 根据上游ID查找对应的RelayEntry
3. 将响应包中的ID还原为客户端原始ID
4. 转发响应给原始客户端
5. 更新缓存（如果响应包含有效记录）
6. 删除映射条目，释放内存



**5.5.4 超时处理机制**

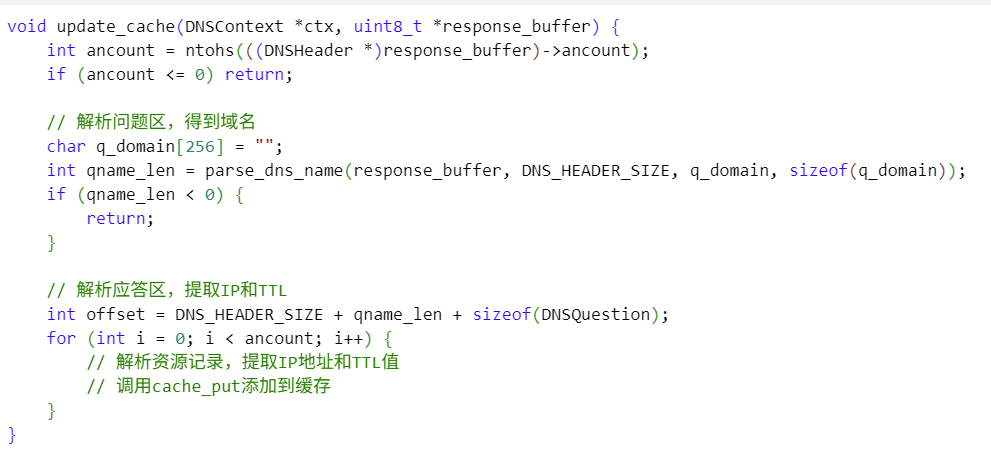
**超时处理：**

1. 定期遍历所有RelayEntry
2. 检查时间戳，超过RELAY\_TIMEOUT（1秒）则视为超时
3. 向客户端发送SERVER\_FAILURE错误响应
4. 删除超时的映射条目，释放资源



**5.5.5 缓存更新**

收到上游DNS响应后，解析应答区中的资源记录，提取IP地址和TTL值，调用cache\_put函数将结果添加到缓存中。



**6. 测试用例及运行结果**

**6.1 服务端**

* 以“管理员身份”运行命令提示符或MSYS2终端，然后运行：

dns\_server.exe

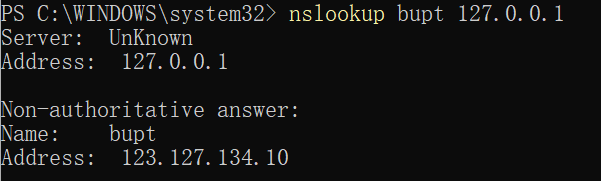
**6.2 客户端**

**6.2.1 使用nslookup验证**

在命令行输入：nslookup example.com 127.0.0.1

example.com就是需要查询的域名

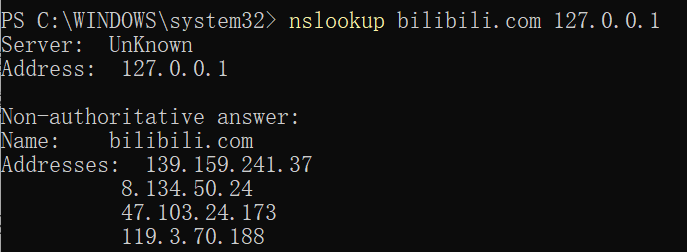
* 查询本地表命中域名，返回预期IP



* 查询被拦截域名，返回“域名不存在”错误



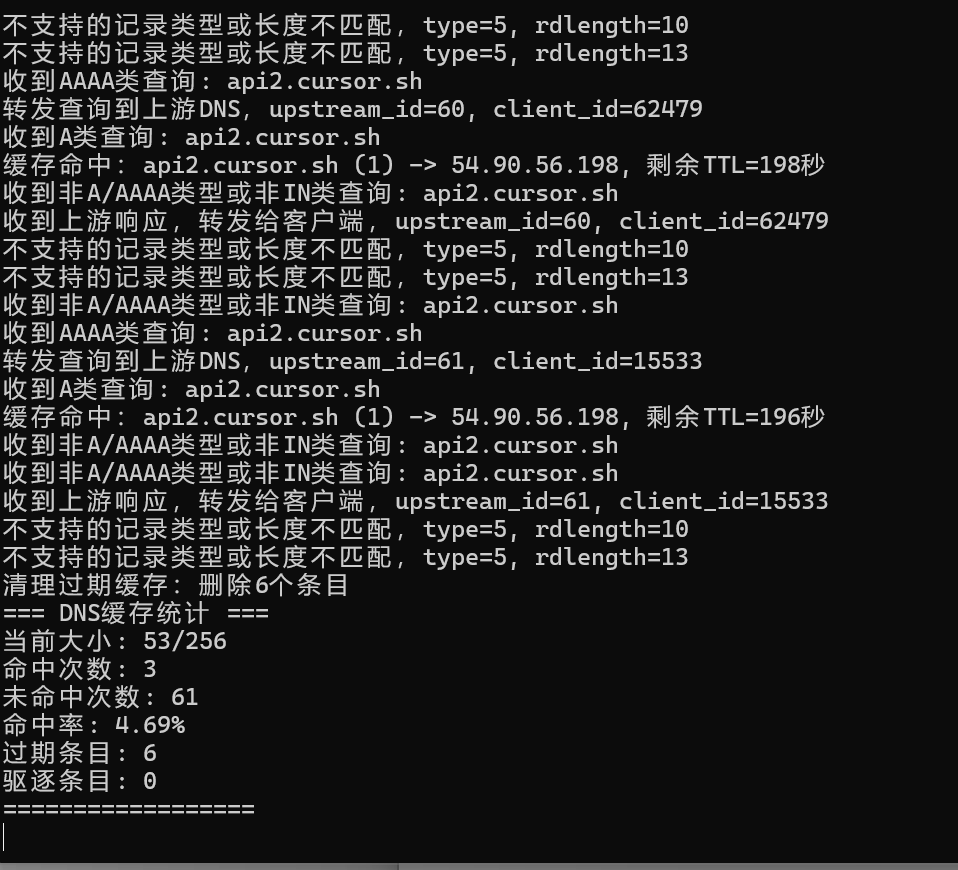
* 查询本地表未命中域名，能正常中继并返回上游结果



**6.2.2 正常上网验证**

* 将本机DNS设置为127.0.0.1，浏览器访问常用网站，解析正常



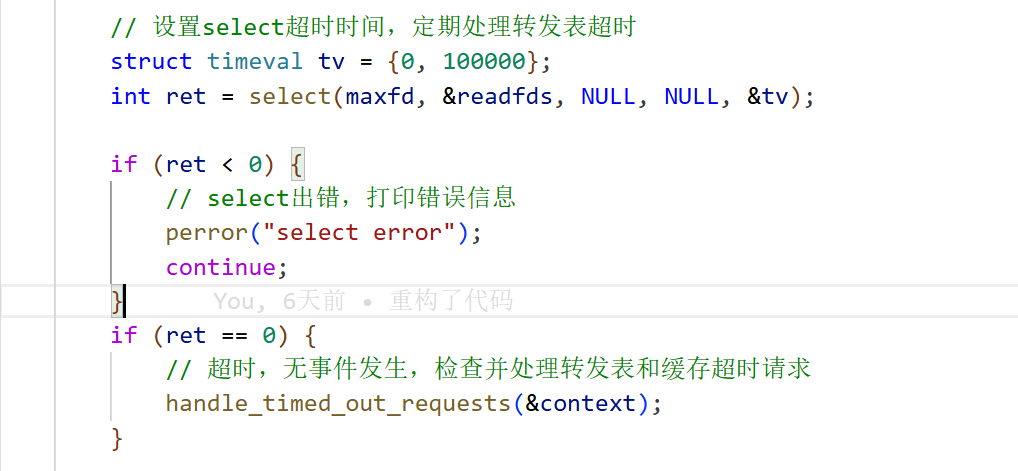


**7. 调试中遇到并解决的问题**

**7.1 超时处理未实际发挥作用**

一开始在ret = select(maxfd, &readfds, NULL, NULL, NULL);后检查if (sec\_diff > RELAY\_TIMEOUT)，这会导致select一直阻塞到客户端或者上游返回响应后才会执行判断，在没有其他响应解除阻塞时原客户端依然会跳出超时未响应，查询失败。

* 解决：利用select函数本身的特性将最后一个参数改为&tv，实现定期检查上游查询是否超时。

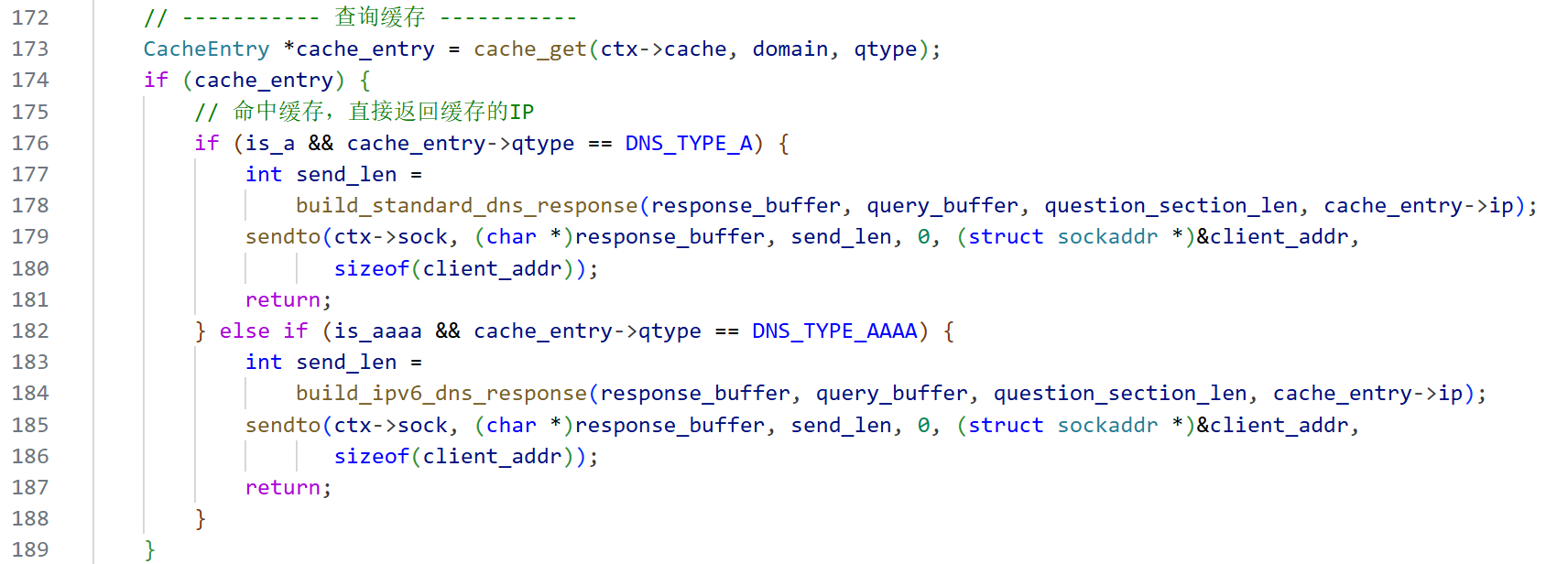


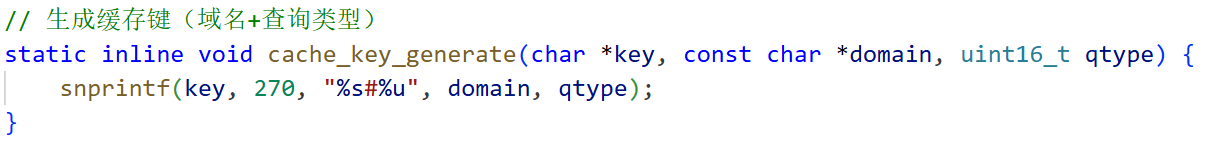
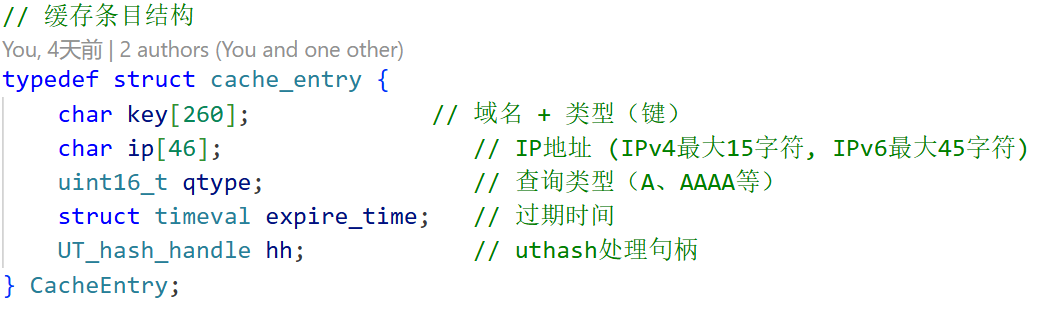
**7.2 ipv6请求无法正常回答**

初始判断缓存命中逻辑复用了本地表在命中A类型时对AAAA返回空响应的逻辑，而客户端一般是都是先发送A请求后发送AAAA请求，在AAAA请求到达时缓存中已有A记录，导致ipv6查询永远无法响应。

* 解决：分离本地表和缓存的判断逻辑，并对缓存数据结构进行改造，可以在哈希表的键中区分type。







**7.3 字段对齐错误**

一开始上游返回资源记录时debug信息始终跳出“不支持的记录类型或长度不匹配“，且type不是正常数字，经抓包分析发现是资源记录中含有压缩指针0xC0xx，而域名解析函数无法正确识别，返回的name字段长度错误。

* 解决：修改了parse\_dns\_name函数，可以正确处理压缩指针，填充正确域名和返回正确偏移量。



**8. 总结和心得体会**

通过本次DNS服务器课程设计，我们成功实现了一个功能完整的DNS代理服务器，支持本地表查询、缓存管理、上游中继等核心功能。虽然我的编程基础相对薄弱，但在队友的耐心指导和帮助下，最终完成了项目的开发工作。在项目开发过程中，我掌握了UDP socket编程、select多路复用、DNS协议解析、哈希表应用、LRU缓存算法等核心技术。通过团队协作，我学会了如何与队友沟通协调、分工配合，体验了完整的项目开发流程。

然而，我必须诚实地反思，虽然完成了分配给自己的那部分任务，但我在学习过程中存在明显的不足。很多时候我只是按照队友的指导完成了代码编写，但对背后的原理和设计思路理解得不够深入。对于DNS协议、网络编程、数据结构等核心知识点，我的理解还停留在表面层次，缺乏系统性的掌握。这种"知其然不知其所以然"的学习方式，虽然能完成任务，但不利于真正的技术成长。

在撰写实验报告的过程中，我不得不重新审视整个项目的架构和实现细节。通过梳理代码结构、分析算法原理、总结设计思路，我对项目的理解从零散变得系统化。为了准确描述技术实现，我深入理解了LRU缓存的"先删后加"机制、ID映射的并发处理原理、DNS协议的报文格式等核心概念。写报告的过程让我学会了主动思考，不再是简单地记录做了什么，而是要理解为什么这样做、如何做的、有什么优缺点。

通过这次项目经历，我深刻认识到真正的学习不仅仅是完成任务，更重要的是理解原理、掌握方法、培养能力。虽然我的基础相对薄弱，但通过努力和团队协作，最终完成了项目目标。感谢队友的帮助和指导，感谢老师的悉心教导。这次经历让我在技术能力、学习方法和团队协作等方面都有了显著提升，为我后续的学习和工作奠定了良好的基础。

——李志敏