计算机网络实验1 DNS实验

实验背景与目标

DNS（域名系统）是互联网的重要组成部分，负责将易于记忆的域名转换为IP地址。本实验的目标是在OpenNetLab平台上实现一个DNS服务器。该服务器需满足以下功能：拦截特定域名，返回域名对应的IP地址，并处理不在本地映射中的DNS请求。

实验过程

实验中的DNS服务器需要处理以下三种类型的请求：

1. **DNS服务器实现** (`server.py`):
   - 服务器初始化时，从`ipconf.txt`读取域名-IP映射。
   - 对于每个接收到的DNS请求，服务器根据请求中的域名执行相应操作：
     - **拦截**：如果域名映射到`0.0.0.0`，返回“域名不存在”的错误。
     - **本地解析**：如果域名映射到有效IP，返回该IP地址。
     - **中继**：如果域名不在映射中，将请求转发给公网DNS服务器，并将响应转发回客户端。
2. **DNS报文处理** (`dns_packet.py`):
   - 实现DNS报文的解析和构建，以正确处理DNS查询和响应。
3. **客户端和主程序** (`client.py`, `main.py`):
   - 用于测试DNS服务器的功能，验证服务器是否正确处理各种DNS请求。

对于 todo 部分，

1. **接收和解析DNS请求：**
   - 使用`DNSPacket`类解析收到的字节流`data`，提取DNS请求的详细信息，包括查询的域名。
2. **处理域名查询：**
   - 检查解析出的域名是否在服务器的`url_ip`字典中。
     - **域名拦截**：如果域名映射到`0.0.0.0`，服务器使用`generate_response`方法构造一个表示域名不存在的DNS响应。
     - **本地解析**：如果域名映射到有效的IP地址，则构造一个包含该IP地址的DNS响应。
     - **请求中继**：如果域名不在映射中，服务器使用`generate_request`方法构造一个新的DNS请求，并准备将其发送到公网DNS服务器。
3. **发送响应或转发请求：**
   - 使用`send`方法将构造的响应或新的DNS请求发送出去。对于非查询类型的DNS请求（例如响应），服务器会直接转发原始数据。

这个实现过程体现了DNS服务器的三个核心功能：解析DNS请求、根据本地映射处理域名查询、以及处理不在映射中的查询请求。通过这种方式，服务器能够对特定域名进行拦截，对已知域名进行快速解析，并将其他请求转发到更广泛的DNS网络中。

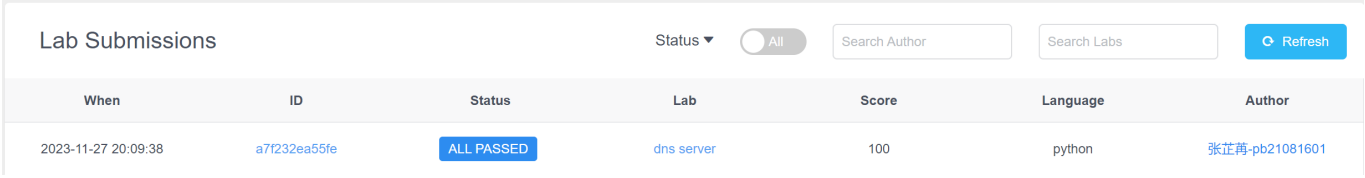
```
import socket
from typing import Dict
from os.path import abspath, dirname, join

from onl.device import UDPDevice
```

```
from onl.sim import Environment

from dns_packet import DNSPacket


class DNSServer(UDPDevice):
    def __init__(self, env: Environment, debug: bool = False):
        super().__init__()
        self.env = env
        # map url to ip address
        self.url_ip: Dict[str, str] = dict()
        with open(join(dirname(abspath(__file__)), "ipconf.txt"), "r",
encoding="utf-8") as f:
            for line in f:
                ip, name = line.split(" ")
                self.url_ip[name.strip("\n")] = ip
        # public DNS server address
        self.name_server = ("223.5.5.5", 53)
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.server_socket.bind(("", 0))
        self.server_socket.setblocking(True)
        self.trans = {}
        self.debug = debug

    def recv_callback(self, data: bytes):
        """
        TODO: 处理DNS请求，data参数为DNS请求数据包对应的字节流
        1. 解析data得到构建应答数据包所需要的字段
        2. 根据请求中的domain name进行相应的处理:
            2.1 如果domain name在self.url_ip中，构建对应的应答数据包，发送给客户端
            2.2 如果domain name不再self.url_ip中，将DNS请求发送给public DNS server
        """

        query_dns = DNSPacket(data)
        if query_dns.QR == 0:  # 检查是否为查询请求
            if self.url_ip.get(query_dns.name) is not None:  # 检查域名是否在本地配
置中
                if self.url_ip[query_dns.name] == "0.0.0.0":
                    self.send(query_dns.generate_response("0.0.0.0", 1))  # 域名拦
截
                else:

self.send(query_dns.generate_response(self.url_ip[query_dns.name], 0))  # 本地解析
            else:
                self.send(query_dns.generate_request(query_dns.name))  # 中继到公网
DNS
        else:
            self.send(data)  # 直接转发响应
```

实验结果

经测试，在本地和 openetlab 上，都能通过所有的样例，因此可以认为实验成功。

```
minerva@minerva-VirtualBox:~/桌面/dns$ ./tester
Running testcase 1: passed
Running testcase 2: passed
Running testcase 3: passed
Running testcase 4: passed
Running testcase 5: passed
Running testcase 6: passed
Running testcase 7: passed
Running testcase 8: passed
Running testcase 9: passed
Running testcase 10: passed
Running testcase 11: passed
Running testcase 12: passed
Running testcase 13: passed
Running testcase 14: passed
Running testcase 15: passed
Running testcase 16: passed
Running testcase 17: passed
Running testcase 18: passed
Running testcase 19: passed
Running testcase 20: passed
All testcases passed, grade is 100
```



反思与收获

通过这个实验，我加深了对DNS工作原理的理解，特别是在域名解析、DNS请求处理以及网络通信方面。实验过程中遇到的挑战，如正确解析DNS报文和设计有效的请求处理逻辑，都极大地提高了我的问题解决能力和编程技巧。此外，这个实验也强化了我对网络编程和通信协议的理解，为未来在网络技术领域的学习和工作奠定了坚实的基础。