layout: post
title: DNS中继服务器
subtitle: 计算机网络课程设计实验报告
date: 2017-11-03
author: Awybupt
header-img: img/page-CL.jpg
catalog: true
tags:
- DNS Relay Server

## - Python 3.6

# 计算机网络课程设计实验报告

# [未完待续2018.04.14]

## DNS中继服务器

> 计算机学院计算机科学与技术2015211306班
> 学号:2015211301
> 魏晓

### 设计目标

- 设计一个DNS服务器程序,读入“IP地址-域名(事先编好)”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,有三种可能的检索结果
    - 检索结果:IP地址0.0.0.0,则向客户端返回“域名不存在”报错信息
        - 不良网站安拦截功能

- 检索结果:普通IP地址,则向客户端返回这个地址,完成一次DNS应答.
  - 服务器功能
- 表中未捡到该域名,则向因特网DNS服务器查询,并将结果返回到客户端,完成一次DNS应答
  - 中继功能

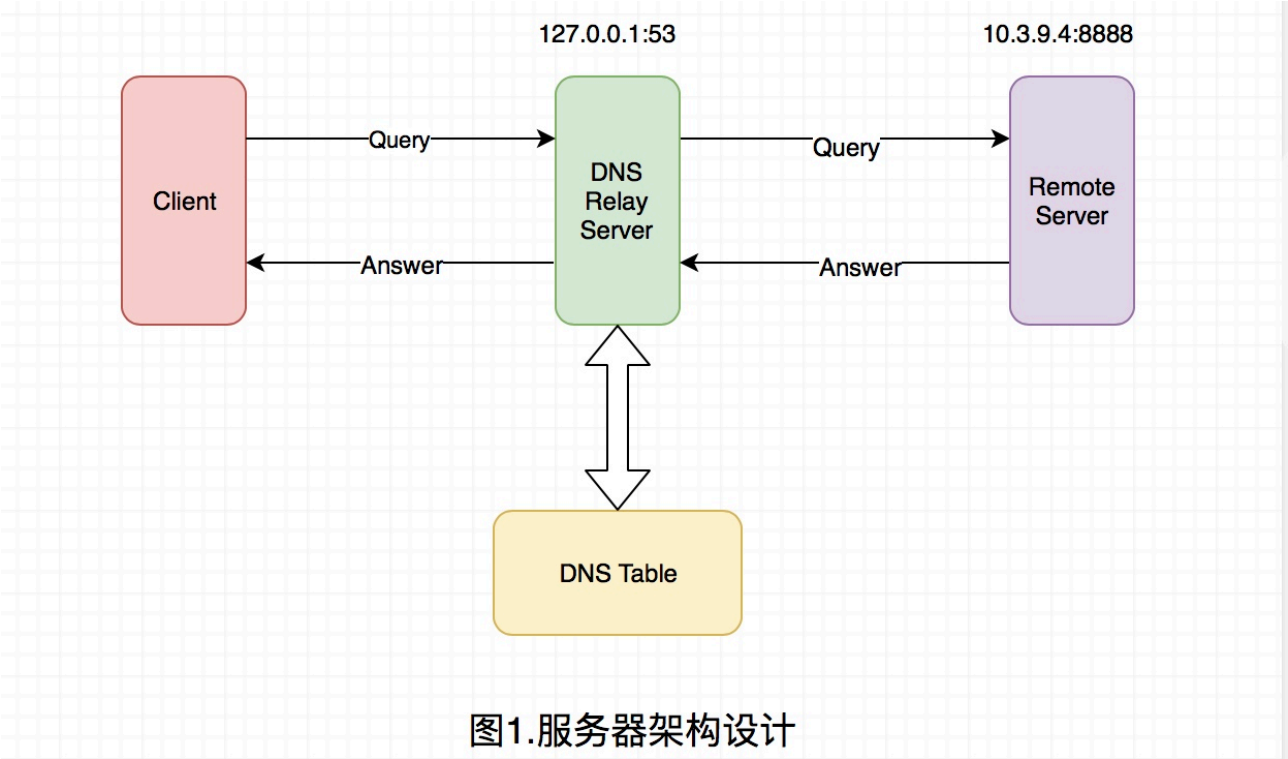


图1.服务器架构设计

---

# 设计思想

**价值定位**

- DNS中继服务器承担的作用不仅仅是中继DNS报文,还担负着不良网站拦截,健康冲浪的作用
- 对于常用网站,拥有可扩展的检索结果预存功能,加速网络访问速度
- DNS中继服务器面向互联网中的众多请求,应当具有处理高并发访问的能力

**架构思路**

- 从client收到的请求应当首先检索本地数据库,如果检索到对应的IPV4 地址应当直接返回信息
- 对于不存在的网址加入请求队列,远程请求端每次都检查请求队列,不为空的时候像远程发送请求,直到空为止
- 对于从远程收到的dns服务包,解析之后,先处理,再加入应答队列

- 本地中继服务器每次都检查应答队列,把存在的应答都发送到请求的端口
- 为了实现DNS服务的稳定性,能够承担足够的并发数量,在DNS SERVER 中使用两个队列处理未在本地DNS表中的请求

### 数据结构

- __TableDNS
  - 本地DNS表
- __QueryQu
  - 远端查询队列
- __ReQu
  - 远端查询结果队列
- __Querydict
  - 查询地址对应请求信息

---

### 方法

- def InitDNS(self)
  - 读取命令行参数,处理命令属性,从DNS表读取解析数据
- def __GetName(self, DataFrame) ```python3 ------ Header : IP, flags, question_count, answer RRs, authority RRs, additional RRs ------ Queries : name, type, class ------ Answers : name, type, class, ttl, data_length, address ------ Flags : * answer : 0x8180 (1000 0001 1000 0000) [is response, recursion desired and available] * query : 0x0100 (0000 0001 0000 0000) [standard query, recursion desired]

---

## 实验思考:

1. Deny Permission 53Port,操作系统拒绝程序直接访问53端口
   - 解决办法:在bash通过命令运行,运行命令如下

```
    sudo python3 LocalServer.py -d 8.8.8.8 -t 100
```

2. 如果有其他程序占用了端口,也会收到操作系统的Deny Permission:
   - 解决办法:杀死当前访问端口的程序,然后重新sudo

```
sudo lsof -i:8080
# Find the process using port 8080
kill XXXX
# Kill it
```

3. 用户有可能输入无效的命令行参数

   - 解决办法:在参数识别上使用正则判断,只有参数为合理的ip地址才修改

```
        if None==match(r'^(25[0-5]|2[0-4]\d|[0-1]?\d?\d)(\.(25[0-5]|2[0-4]\
d|[0-1]?\d?\d)){3}$',va):
            print("Invalid DNS Server,DNS server change to 8.8.8.8")
        else:
            self.__ServerAdd = va
```

4. 每次从本地文件读取DNS表都会浪费大量的时间,这些文件在内存中占用大量的空间,只有极小一部分有价值

   - 解决办法:将dnstable存储在MySQL数据库之中,在查询的时候再查询,查询结束之后关闭链接,释放资源

```
db = pymysql.connect("localhost", "wx", "password", "mytest")
cursor = db.cursor()#使用cursor（）方法获取操作游标
sql="SELECT address from "+self.__TableName+" where name='"+QueryName+"
'"
try :
    cursor.execute(sql)
    ans= cursor.fetchall()
    ans = ans.__str__()
    ans = ans[3:-5]
except:
    ans=''
finally:
    cursor.close()
    db.close()
```