

计算机网络课程报告: DNS服务器


一、成员信息

姓名	班级	学号	分工配比
卢昱昊	4班	2016210537	33.3%
王增瑞	4班	2016211214	33.3%
黄蔚然	4班	2016211213	33.3%

二、功能设计


- 1. 实现了基本要求的dns功能:
 - 1. 检索结果为正常ipv4, 则返回此地址
 - 2. 检索结果为 0.0.0.0 则返回域名不存在的报错信息
 - 3. 未检测到该域名则向实际远程dns服务器发送查询请求, 并将结果返回给客户端。(中继功能)
- 2. 采用socket I/O多路复用方式, 实现多客户端并发。
- 3. 设置超时处理, 作为中继时, 能够自动断开超时连接并返回信息提示客户端。
- 4. 使用线程锁, 解决ID冲突转换问题。
- 5. 实现文件存储, 自动读取本地数据文件, 并在数据更新时自动存入本地数据文件
- 6. 支持 ttl 判断。对于 ttl 超时的记录予以删除, 且令指定类型没有记录为转发的必要条件。
- 7. 当作为中继进行查询后, 实时更新本地数据记录, 同时写入文件。保存记录的地址、 ttl 与更新时间戳。
- 8. 支持多ip: answer多ip与保存多ip记录。
- 9. 支持ipv6的以上所有功能。(支持 A (ipv4)与 A (ipv6),对于其他类型, 只递归获取 A (AAAA), 不保存递归过程。)
- 10. 支持实时生成log文件。
- 11. 支持带参数运行程序, 通过参数可以指定远程dns地址, 指定是否存入log文件等。

三、实验要求



课程设计题目： DNS中继服务器的实现

- 设计一个DNS服务器程序，读入“域名-IP地址”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，实现下列三种情况：
 - 检索结果为IP地址0.0.0.0，则向客户端返回“域名不存在”的报错消息（即不良网站拦截功能）
 - 检索结果为普通IP地址，则向客户返回这个地址（即DNS服务器功能）
 - 表中未检到该域名，则向实际的本地DNS服务器发出查询，并将结果返给客户端（即DNS中继功能）
 - **注意：**应考虑多个计算机上的客户端同时查询的情况，需要进行消息ID的转换

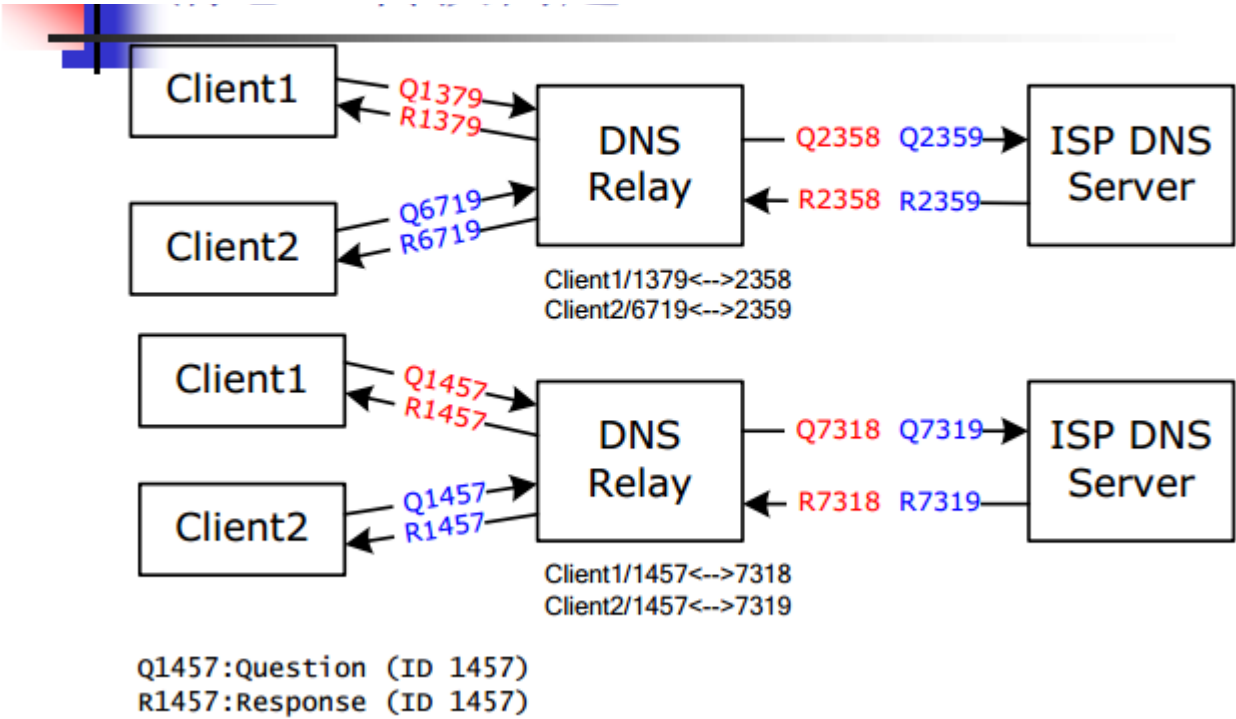


程序必须要考虑的两个问题

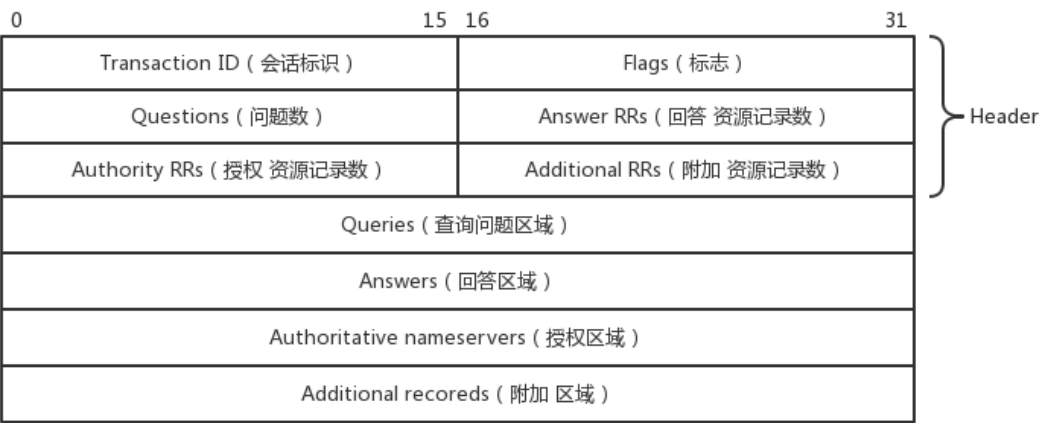
- 多客户端并发
 - 允许多个客户端（可能会位于不同的多个计算机）的并发查询，即：允许第一个查询尚未得到答案前就启动处理另外一个客户端查询请求（DNS报头中ID字段的作用）
- 超时处理
 - 由于UDP的不可靠性，考虑求助外部DNS服务器（中继）却不能得到应答或者收到迟到应答的情形



消息ID转换问题



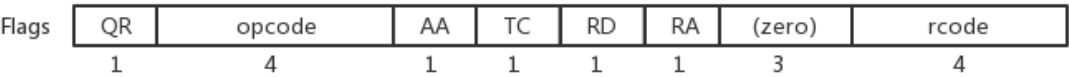
四、DNS报文格式



DNS协议报文格式

头部

#



会话标识（2字节）

是DNS报文的ID标识，对于请求报文和其对应的应答报文，这个字段是相同的，通过它可以区分DNS应答报文是哪个请求的响应

标志（2字节）

标志	含义
QR（1bit）	查询/响应标志，0为查询，1为响应
opcode（4bit）	0表示标准查询，1表示反向查询，2表示服务器状态请求
AA（1bit）	表示授权回答
TC（1bit）	表示可截断的
RD（1bit）	表示期望递归
RA（1bit）	表示可用递归

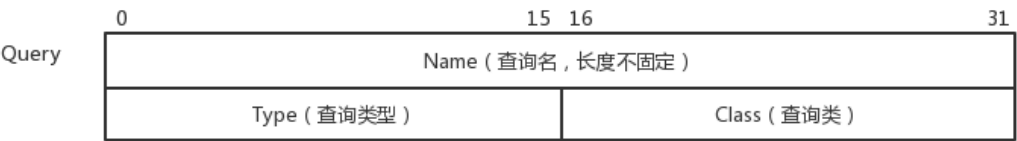
rcode（4bit）表示返回码，0表示没有差错，3表示名字差错，2表示服务器错误（Server Failure）

数量字段（总共8字节）

Questions、Answer RRs、Authority RRs、Additional RRs 各自表示后面的四个区域的数量。Questions表示查询问题区域节的数量，Answers表示回答区域的数量，Authoritative nameservers表示授权区域的数量，Additional records表示附加区域的数量

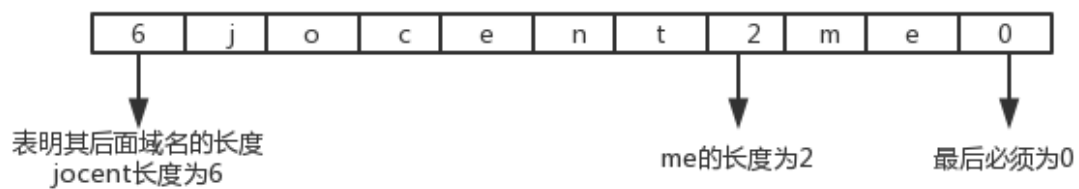
正文#

Queries区域:



查询名

长度不固定，且不使用填充字节，一般该字段表示的就是需要查询的域名（如果是反向查询，则为IP，反向查询即由IP地址反查域名），一般的格式如下图所示。



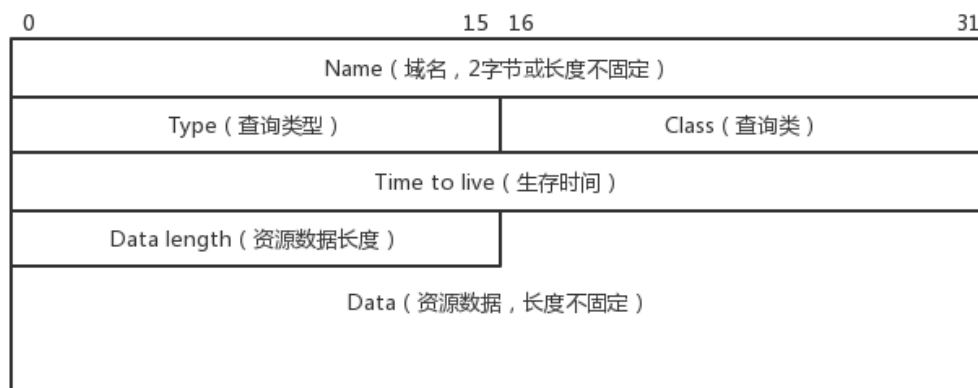
查询类型

类型	助记符	说明
1	A	由域名获得IPv4地址
2	NS	查询域名服务器
5	CNAME	查询规范名称
6	SOA	开始授权
11	WKS	熟知服务
12	PTR	把IP地址转换成域名
13	HINFO	主机信息
15	MX	邮件交换
28	AAAA	由域名获得IPv6地址
252	AXFR	传送整个区的请求
255	ANY	对所有记录的请求

查询类

通常为1，表明是Internet数据

资源记录(RR)区域



资源记录格式

该区域有三个，但格式都是一样的。这三个区域分别是：回答区域，授权区域和附加区域

域名（2字节或不定长）

它的格式和Queries区域的查询名字字段是一样的。有一点不同就是，当报文中域名重复出现的时候，该字段使用2个字节的偏移指针来表示。比如，在资源记录中，域名通常是查询问题部分的域名的重复，因此用2字节的指针来表示，具体格式是最前面的两个高位是 11，用于识别指针。其余的14位从DNS报文的开始处计数（从0开始），指出该报文中的相应字节数。一个典型的例子，C00C(1100000000001100，12正好是头部的长度，其正好指向Queries区域的查询名字字段)。

查询类型

表明资源纪录的类型，见1.2节的查询类型表格所示

查询类

对于Internet信息，总是IN

生存时间（TTL）

以秒为单位，表示的是资源记录的生命周期，一般用于当地址解析程序取出资源记录后决定保存及使用缓存数据的时间，它同时也可以表明该资源记录的稳定程度，极为稳定的信息会被分配一个很大的值（比如86400，这是一天的秒数）。

资源数据

该字段是一个可变长字段，表示按照查询段的要求返回的相关资源记录的数据。可以是Address（表明查询报文想要的回应是一个IP地址）或者CNAME（表明查询报文想要的回应是一个规范主机名）等。

Wireshark分析DNS协议

请求报文

- ▼ Domain Name System (query)
 - [Response In: 1612]
 - Transaction ID: 0xdd9b
 - ▼ Flags: 0x0100 Standard query
 - 0... .. = Response: Message is a query
 - .000 0... .. = Opcode: Standard query (0)
 -0. = Truncated: Message is not truncated
 -1 = Recursion desired: Do query recursively
 -0... = Z: reserved (0)
 -0 = Non-authenticated data: Unacceptable
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 0
 - ▼ Queries
 - ▼ jocent.me: type A, class IN
 - Name: jocent.me
 - [Name Length: 9]
 - [Label Count: 2]
 - Type: A (Host Address) (1)
 - Class: IN (0x0001)

0000	9c 06 1b 00 68 21 50 7b 9d df 48 6b 08 00 45 00h!P{ ..Hk..E.
0010	00 37 30 04 00 00 40 11 00 00 0a 4a 24 5a 0a 4a	.70...@. ...J\$Z.J
0020	01 0b fa b5 00 35 00 23 3a 2d dd 9b 01 00 00 015.# :-.....
0030	00 00 00 00 00 00 06 6a 6f 63 65 6e 74 02 6d 65j ocent.me
0040	00 00 01 00 01

响应报文

```

v Domain Name System (response)
  [Request In: 1611]
  [Time: 0.001135000 seconds]
  Transaction ID: 0xdd9b
  v Flags: 0x8180 Standard query response, No error
    1... .. = Response: Message is a response
    .000 0... .. = Opcode: Standard query (0)
    .... 0... .. = Authoritative: Server is not an authority for domain
    .... ..0... .. = Truncated: Message is not truncated
    .... ..1... .. = Recursion desired: Do query recursively
    .... ..1... .. = Recursion available: Server can do recursive queries
    .... ..0... .. = Z: reserved (0)
    .... ..0... .. = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... ..0... .. = Non-authenticated data: Unacceptable
    .... ..0000 = Reply code: No error (0)
  Questions: 1
  Answer RRs: 2
  Authority RRs: 0
  Additional RRs: 0
  v Queries
    v jocent.me: type A, class IN
      Name: jocent.me
      [Name Length: 9]
      [Label Count: 2]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
  v Answers
    v jocent.me: type A, class IN, addr 104.28.15.151
      Name: jocent.me
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 457
      Data length: 4
      Address: 104.28.15.151
    v jocent.me: type A, class IN, addr 104.28.14.151
      Name: jocent.me
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 457
      Data length: 4
      Address: 104.28.14.151

```

五、模块划分

整个程序的核心功能在DnsRelay的类中。 `__main__.py` 只负责处理参数。以下主要介绍DnsRelay中需要的模块。

- `__init__(self,server_ip="8.8.8.8", config_file="zones.json", debug1=False, debug2=False, autosave=False)`

初始化函数。包括定义远程dns地址，数据文件地址，确定是否打印log，是否自动保存更新数据文件。打开创建线程池，开启socket监听，读入数据文件至内存。

- `load_zones()`

读取json格式数据到内存。

- `get_ip(msg, qtype):`

根据输入的二进制信息，和类型（ A / AAAA ）,解析ip地址字符串，用于打印log

- `get_qname(msg)` 输入的二进制信息，返回查询域名字符串与msg中域名之后的index构成的元组
- `needremote(qname, qtype)`

根据查询的域名，以及其类型，通过判断其是否合法(不为0.0.0.0或::)、ttl是否过期判断是否需要查询远程dns服务器。顺便会删除过期的记录。

- `handle_request()`

用于接收请求，分析head信息与query信息，判断应该返回ipv4记录、返回ipv6记录还是作为中继向远程dns发送请求，通过相应的方式获取响应信息，发送回请求端。

- `create_response(msg, addr_list)`

根据参数请求msg与域名对应的本地ipv4记录，构造响应信息并返回。

- `create_response_ipv6(msg, addr_list)`

功能同上，不同处：ipv6

- `remote_dns(msg, qtype)`

参数为请求msg，qtype为请求类型（`A` / `AAAA`）给远程dns发送请求消息。在发送消息时，采用线程锁，维护ID转换字典。对于冲突的ID，进行hash直到没有冲突ID，然后将转换后ID-原ID存入字典，然后利用转换后ID进行消息转发。转发进行超时处理。对于接收的数据，分析记录并保存到本地数据库，同时更新数据文件。在将转换ID换回原ID后，返回查询信息

- `b2v6(bstr)`

参数为二进制串，返回国际标准格式ipv6字符串(支持0的省略，双冒号等).用于存储数据与log文件

- `v62b(str)`

以上函数的对偶函数。参数为国际标准格式ipv6(不标准也可以,兼容所有逻辑无歧义格式)，返回二进制ipv6串。用于读取数据进行报文构造。

六、软件流程图

IPV4

```
PS C:\Users\huang> nslookup www.github.com 127.0.0.1
DNS request timed out.
    timeout was 2 seconds.
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:     github.com
Address:  13.250.177.223
Aliases:  www.github.com
```

IPV6

```
PS C:\Users\huang> nslookup www.bupt.edu.cn 127.0.0.1
DNS request timed out.
    timeout was 2 seconds.
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:     vn.bupt.edu.cn
Addresses: 2001:da8:215:4038::161
          10.3.9.161
Aliases:  www.bupt.edu.cn
```

中继功能测试

#

可以看到，在查询表中没有的域名时，结果正确；正常返回多个对应的IP结果，并把结果写入配置文件中。

```
PS C:\Users\huang> nslookup www.huangweiran.club 127.0.0.1
DNS request timed out.
    timeout was 2 seconds.
服务器:  UnKnown
Address:  127.0.0.1

非权威应答:
名称:     etodemerzel0427.github.io
Addresses: 185.199.108.153
          185.199.111.153
          185.199.109.153
          185.199.110.153
Aliases:  www.huangweiran.club
```

设置为默认DNS服务器测试

#

同时刷新数个网页情况正常。

其他测试在验收时已经完整展示，此处不再详述。

八、遇到的问题

1. 不同平台包格式的问题。我们在Windows和Linux都进行了测试，发现在Linux上结果不正确。

经过抓包分析，我们发现Linux上报文格式与Windows下不太相同。包括 `Flags` 部分有几位存在差异、`additional section`的存在等问题。

在最初的实现中，我们对IPV4对应 `flags` 的判断为：

```
if (flags == b'\x01\x00' or flags == b'\x00\x00') and qtype == b'\x00\x01' and qclass == b'\x00\x01':  
    pass
```

经过分析，我们认为实际上只要分析 `flags` 的第一个字符即可：

```
if (flags[0] == 0 or flags[0] == 1) and qtype == b'\x00\x01' and qclass == b'\x00\x01':  
    pass
```

另外，由于之前在Windows下，query报文中是不存在additional section的，我们取 `qtype` 和 `qclass` 是从报文末尾往前取的；但在有additional section的情况下，这样的取法会取到additional section的部分，因此是错误的。通过修改 `get_qname` 函数的内容，我们解决了这个问题。

经过这样的处理，我们的程序完成了多平台的适配。

2. 从远程服务器的应答报文中获取IPV6地址出错

一开始我们从报文中获取IPV6地址是简单地在answer中从后往前取16个字节作为IPV6地址，但是我们忽略了Additional部分的存在，导致如果应答报文中含有Additional部分，则获取到的IPV6地址为无效地址，而且这种做法也不支持多IPV6地址的获取。所以我们改变了获取IPV6地址的逻辑，根据之前的 `type` 和 `data_length` 等属性来推断IPV6地址的位置。

部分代码如下：

```
elif atype == b'\x00\x1c':  
    # print("IPV6 RECORD TO ADD")  
    ans_ptr += 4 # skip type and class IN  
    attl = int.from_bytes(ans[ans_ptr:ans_ptr + 4], "big")  
  
    ans_ptr += 4 # skip to data length  
    data_len = int.from_bytes(ans[ans_ptr:ans_ptr + 2], "big")  
  
    ans_ptr += 2 # skip to address  
    addr_str = b2v6(ans[ans_ptr:ans_ptr + data_len])  
  
    ans_ptr += data_len  
    if qname not in self.zones:  
        self.zones[qname] = {}  
        self.zones[qname]["AAAA"] = [{"ttl": attl, "addr": addr_str, "ts": time.time()}]  
        self.zones[qname]["A"] = []  
    else:  
        self.zones[qname]["AAAA"].append({"ttl": attl, "addr": addr_str, "ts":  
time.time()})  
    print("(IPV6)" + qname + ": " + addr_str)
```

3. 在设置被屏蔽网址的 ttl 时忽略的问题

当我们在本地数据存储格式中加入ttl时，对被屏蔽的网址也进行了同样的处理。我们对于更新本地数据的方法是每当收到请求报文，就根据当前时间和数据文件中的ttl和时间戳ts判断该IP是否已经过期，若所有IP均过期则转发请求报文，收到应答后更新本地数据。

这种处理逻辑导致如果被屏蔽的报文ttl超时，则会失去屏蔽作用，自动转发请求报文，所以我们在 `needremote(qname, qtype)` 中加入了对被屏蔽网址的判断，代码如下：

```
def needremote(self, qname, qtype):
    if qname not in self.zones:
        return False
    timestamp = time.time()

    if (len(self.zones[qname]["A"]) and self.zones[qname]["A"][0]["value"] ==
        "0.0.0.0") \
        or (len(self.zones[qname]["AAAA"]) and self.zones[qname]["AAAA"][0]
            ["addr"] == "0:0:0:0:0:0:0:0"):
        return True

    if qtype == b'\x00\x01':
        newA = []
        for item in self.zones[qname]["A"]:
            if "ts" in item.keys() and timestamp - item["ts"] < item["ttl"]:
                newA.append(item)
            elif "ts" not in item.keys() and timestamp - self.starttime < item["ttl"]:
                newA.append(item)
        self.zones[qname]["A"] = newA
        return len(newA)

    if qtype == b'\x00\x1c':
        newAAAA = []
        for item in self.zones[qname]["AAAA"]:
            if "ts" in item.keys() and timestamp - item["ts"] < item["ttl"]:
                newAAAA.append(item)
            elif "ts" not in item.keys() and timestamp - self.starttime < item["ttl"]:
                newAAAA.append(item)
        self.zones[qname]["AAAA"] = newAAAA
        return len(newAAAA)
```

4. 在 Python 中进行 socket 绑定，为了使 DNS 指向自己，是应该显示绑定 ('127.0.0.1', 53) 还是 ('',53)？两者有何区别？

在 Python 的 socket 方法中，不显示指出 127.0.0.1 与 C 语言中的 `INADDR_ANY` 宏具有相同的含义，后者表示绑定 localhost 地址和网络真实地址（可能经过了 NAT）。如果显示指出 127.0.0.1，那么多机测试中，客户端向服务器发送的 DNS 请求不能被此 socket 监听到。

九、心得

通过这次实验，我们加深了对以下方面的认识：

1. dns协议各报文段的含义
2. 域名系统的运作方式
3. nslookup与dig等命令的运作原理
4. 大端与小端的应用场景
5. ipv4与ipv6的地址格式

提高了以下能力：

1. 使用python进行socket多线程编程能力
2. 使用python进行格式转换的能力，如二进制串，整形，字符串的转换。
3. 团队协作能力

虽然已经实现了所有需求的功能，并在最后的验收中取得了AAA的成绩，但是由于精力有限，还有一些可以提高的地方可以添加如：

1. 支持更多的类型。目前只支持 `A` 与 `AAAA`。
2. 优化内存与效率。目前每次中继获得结果后直接dump所有数据重写文件，可以改良使用数据库代替json文件
3. 增加对authority与additional的支持，目前这两段直接舍弃。
4. 目前的查询模式还是递归查询，相当于从一个主机发送给一个主机，我们作为接收主机再请求本地dns服务器。下一步可以升级为迭代查询，成为一个真正的本地dns。