DNS 教程

DNS 的介绍

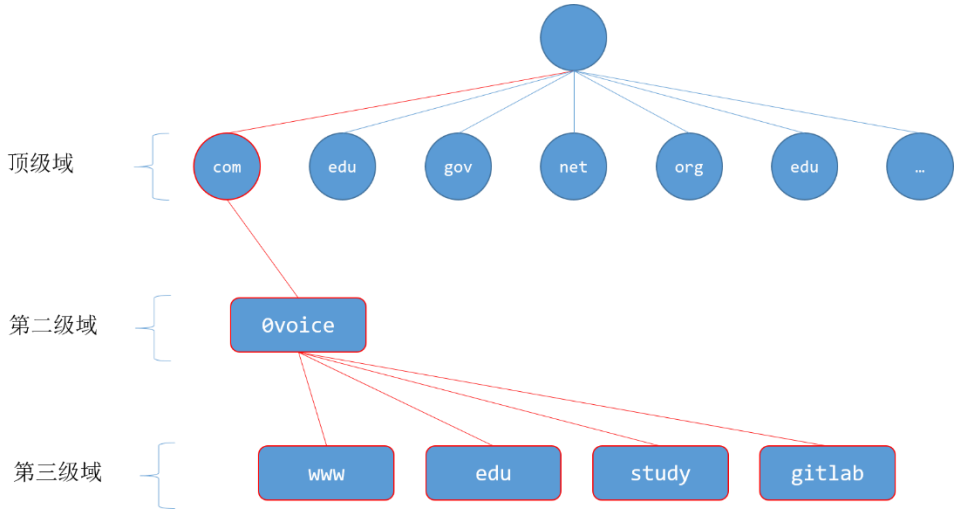
域名系统（英文：Domain Name System，缩写：DNS）是互联网的一项服务。它作为将域名和 IP 地址相互映射的一个分布式数据库，能够使人更方便地访问互联网。DNS 使用 TCP 和 UDP 端口 53。当前，对于每一级域名长度的限制是 63 个字符，域名总长度则不能超过 253 个字符。

域名系统（英文：Domain Name System，缩写：DNS）的作用是将人类可读的域名 (如，www.example.com) 转换为机器可读的 IP 地址 (如，192.0.2.44)。

域名是由一串用点分隔符 . 组成的互联网上某一台计算机或计算机组的名称，用于在数据传输时标识计算机的方位。域名可以说是一个 IP 地址的代称，目的是为了便于记忆后者。例如，www.0voice.com 是一个域名，和 IP 地址 122.152.222.180 相对应。人们可以直接访问 www.0voice.com 来代替 IP 地址，然后域名系统（DNS）就会将它转化成便于机器识别的 IP 地址。这样，人们只需要记忆 www.0voice.com 这一串带有特殊含义的字符，而不需要记忆没有含义的数字。

DNS 的分层

域名系统是分层次的。在域名系统的层次结构中，各种域名都隶属于域名系统根域的下级。域名的第一级是顶级域，它包括通用顶级域，例如 .com、.net 和 .org；以及国家和地区顶级域，例如 .us、.cn 和 .tk。顶级域名下一层是二级域名，一级一级地往下。这些域名向人们提供注册服务，人们可以用它创建公开的互联网资源或运行网站。顶级域名的管理服务由对应的域名注册管理机构（域名注册局）负责，注册服务通常由域名注册商负责。



DNS 服务类型

- **授权型 DNS**- 一种授权型 DNS 服务提供一种更新机制，供开发人员用于管理其公用 DNS 名称。然后，它响应 DNS 查询，将域名转换为 IP 地址，以便计算机可以相互通信。授权型 DNS 对域有最终授权且负责提供递归型 DNS 服务器对 IP 地址信息的响应。阿里云是一种授权型 DNS 系统。
- **递归型 DNS**- 客户端通常不会对授权型 DNS 服务直接进行查询。而是通常连接到称为解析程序的其他类型 DNS 服务，或递归型 DNS 服务。递归型 DNS 服务就像是旅馆的门童：尽管没有任何自身的 DNS 记录，但是可充当代表您获得 DNS 信息的中间程序。如果递归型 DNS 拥有已缓存或存储一段时间的 DNS 参考，那么它会通过提供源或 IP 信息来响应 DNS 查询。如果没有，则它会将查询传递到一个或多个授权型 DNS 服务器以查找信息。

记录类型

DNS 中，常见的资源记录类型有：

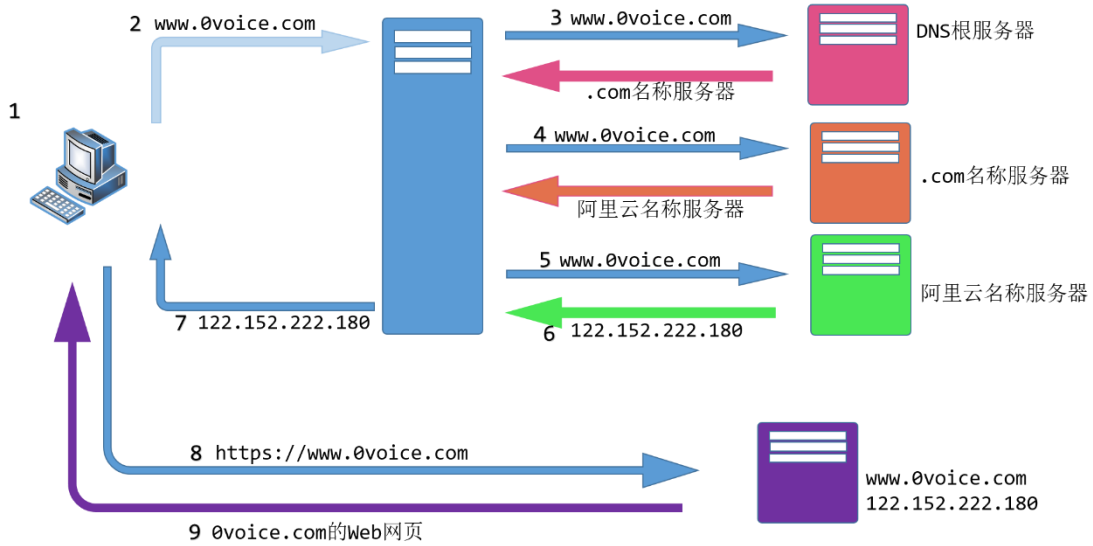
- **NS 记录（域名服务）** – 指定解析域名或子域名的 DNS 服务器。
- **MX 记录（邮件交换）** – 指定接收信息的邮件服务器。
- **A 记录（地址）** – 指定域名对应的 IPv4 地址记录。
- **AAAA 记录（地址）** – 指定域名对应的 IPv6 地址记录。
- **CNAME（规范）** – 一个域名映射到另一个域名或 CNAME 记录（0voice.com 指向 www.0voice.com ）或映射到一个 A 记录。
- **PTR 记录（反向记录）** – PTR 记录用于定义与 IP 地址相关联的名称。 PTR 记录是 A 或 AAAA 记录的逆。 PTR 记录是唯一的，因为它们以 .arpa 根开始并被委派给 IP 地址的所有者。

域名解析

主机名到 IP 地址的映射有两种方式：

- **静态映射** - 在本机上配置域名和 IP 的映射，旨在本机上使用。Windows 和 Linux 的 hosts 文件中的内容就属于静态映射。
- **动态映射** - 建立一套域名解析系统（DNS），只在专门的 DNS 服务器上配置主机到 IP 地址的映射，网络上需要使用主机名通信的设备，首先需要到 DNS 服务器查询主机所对应的 IP 地址。

通过域名去查询域名服务器，得到 IP 地址的过程叫做域名解析。在解析域名时，一般先静态域名解析，再动态解析域名。可以将一些常用的域名放入静态域名解析表中，这样可以大大提高域名解析效率。



上图展示了一个动态域名解析的流程，步骤如下：

1. 用户打开 Web 浏览器，在地址栏中输入 www.0voice.com，然后按 Enter 键。
2. www.0voice.com 的请求被路由到 DNS 解析程序，这一般由用户的 Internet 服务提供商 (ISP) 进行管理，例如有线 Internet 服务提供商、DSL 宽带提供商或公司网络。
3. ISP 的 DNS 解析程序将 www.0voice.com 的请求转发到 DNS 根名称服务器。
4. ISP 的 DNS 解析程序再次转发 www. 0voice.com 的请求，这次转发到 .com 域的一个 TLD 名称服务器。.com 域的名称服务器使用与 0voice.com 域相关的四个阿里云名称服务器的名称来响应该请求。
5. ISP 的 DNS 解析程序选择一个阿里云名称服务器，并将 www. 0voice.com 的请求转发到该名称服务器。
6. 阿里云名称服务器在 0voice.com 托管区域中查找 www. 0voice.com 记录，获得相关值，例如，Web 服务器的 IP 地址 (192.0.2.44)，并将 IP 地址返回至 DNS 解析程序。
7. ISP 的 DNS 解析程序最终获得用户需要的 IP 地址。解析程序将此值返回至 Web 浏览器。DNS 解析程序还会将 0voice.com 的 IP 地址缓存 (存储) 您指定的时长，

以便它能够在下次有人浏览 0voice.com 时更快地作出响应。有关更多信息，请参阅存活期 (TTL)。

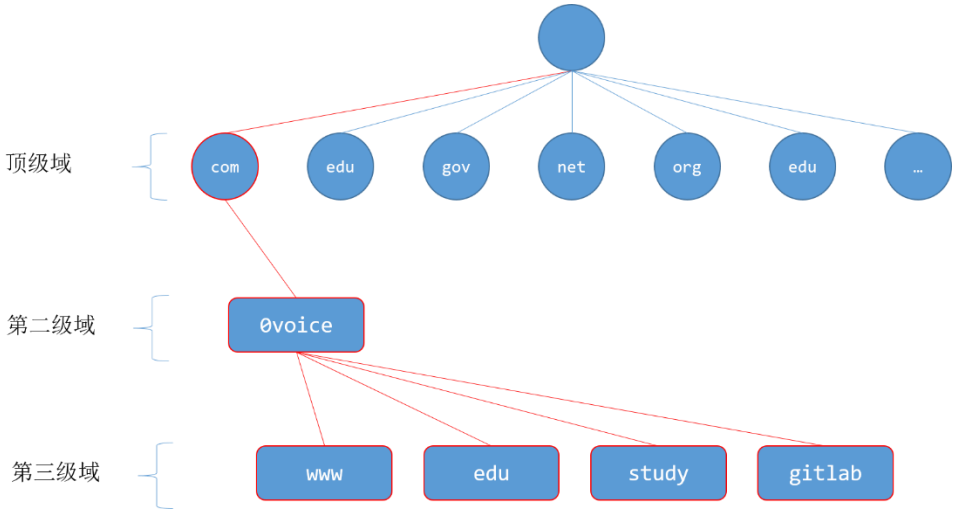
8. Web 浏览器将 www.0voice.com 的请求发送到从 DNS 解析程序中获得的 IP 地址。这是您的内容所处位置，例如，在阿里云实例中或配置为网站终端节点的阿里云存储桶中运行的 Web 服务器。
9. 192.0.2.44 上的 Web 服务器或其他资源将 www.0voice.com 的 Web 页面返回到 Web 浏览器，且 Web 浏览器会显示该页面。

DNS 协议

域名结构

域名系统并不像电话号码通讯录那么简单，通讯录主要是单个个体在使用，同一个名字出现在不同个体的通讯录里并不会出现问题，但域名是群体中所有人都在用的，必须要保持唯一性。为了达到唯一性的目的，因特网在命名的时候采用了层次结构的命名方法。每一个域名（本文只讨论英文域名）都是一个标号序列（labels），用字母（A-Z，a-z，大小写等价）、数字（0-9）和连接符（-）组成，标号序列总长度不能超过 255 个字符，它由点号分割成一个个的标号（label），每个标号应该在 63 个字符之内，每个标号都可以看成一个层次的域名。级别最低的域名写在左边，级别最高的域名写在右边。域名服务主要是基于 UDP 实现的，服务器的端口号为 53。

比如：域名 0voice.com，由点号分割成了两个域名 0voice 和 com，其中 com 是顶级域名（TLD，Top-Level Domain）， 0voice 是二级域名（SLD，Second Level Domain）。关于域名的层次结构，请看下面的示意图。

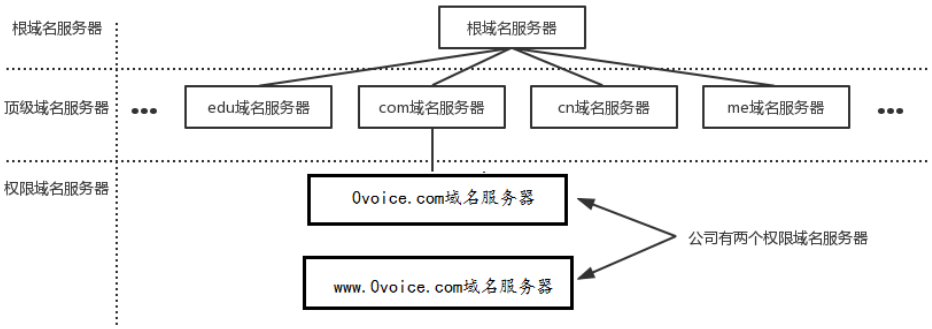


注意：最开始的域名最后都是带了点号的，比如 0voice.com，应该是 0voice.com.，最后面的点号表示根域名服务器，后来发现所有的网址都要加上最后的点，就简化了写法，干脆所有的都不加，但是你在网址后面加上点号也是可以正常解析的。

域名服务器

有域名结构还不行，还需要有一个东西去解析域名，手机通讯录是由通讯录软件解析的，域名需要由遍及全世界的域名服务器去解析，域名服务器实际上就是装有域名系统的主机。由高向低进行层次划分，可分为以下几大类：

- 根域名服务器：最高层次的域名服务器，也是最重要的域名服务器，本地域名服务器如果解析不了域名就会向根域名服务器求助。全球共有 13 个不同 IP 地址的根域名服务器，它们的名称用一个英文字母命名，从 a 一直到 m。这些服务器由各种组织控制，并由 ICANN（互联网名称和数字地址分配公司）授权，由于每分钟都要解析的名称数量多得令人难以置信，所以实际上每个根服务器都有镜像服务器，**每个根服务器与它的镜像服务器共享同一个 IP 地址，**中国大陆地区内只有 6 组根服务器镜像（F，I（3 台），J，L）。当你对某个根服务器发出请求时，请求会被路由到该根服务器离你最近的镜像服务器。所有的根域名服务器都知道所有的顶级域名服务器的域名和地址，如果向根服务器发出对 0voice.com 的请求，则根服务器是不能在它的记录文件中找到与 0voice.com 匹配的记录。但是它会找到.com 的顶级域名记录，并把负责.com 地址的顶级域名服务器的地址发回给请求者。
- 顶级域名服务器：负责管理在该顶级域名服务器下注册的二级域名。当根域名服务器告诉查询者顶级域名服务器地址时，查询者紧接着就会到顶级域名服务器进行查询。比如还是查询 0voice.com，根域名服务器已经告诉了查询者 com 顶级域名服务器的地址，com 顶级域名服务器会找到 0voice.com 的域名服务器的记录，域名服务器检查其区域文件，并发现它有与 0voice.com 相关联的区域文件。在此文件的内部，有该主机的记录。此记录说明此主机所在的 IP 地址，并向请求者返回最终答案。
- 权限域名服务器：负责一个区的域名解析工作
- 本地域名服务器：当一个主机发出 DNS 查询请求的时候，这个查询请求首先就是发给本地域名服务器的。



域名解析过程

域名解析总体可分为两大步骤，第一个步骤是本机向本地域名服务器发出一个 DNS 请求报

文，报文里携带需要查询的域名；第二个步骤是本地域名服务器向本机回应一个 DNS 响应报文，里面包含域名对应的 IP 地址。从下面对 0voice.com 进行域名解析的报文中可明显看出这两大步骤。**注意：第二大步骤中采用的是迭代查询，其实是包含了很多小步骤的，详情见下面的流程分析。**

```
4401 48.221346    192.168.2.200    192.168.2.1      DNS      74 Standard query 0x0002 A www.0voice.com
4402 48.221362    192.168.2.200    192.168.2.1      DNS      74 Standard query 0x0002 A www.0voice.com
4403 48.231341    192.168.2.1      192.168.2.200    DNS      90 Standard query response 0x0002 A www.0voice.com A 122.152.222.180
4404 48.234183    192.168.2.200    192.168.2.1      DNS      74 Standard query 0x0003 AAAA www.0voice.com
4405 48.234201    192.168.2.200    192.168.2.1      DNS      74 Standard query 0x0003 AAAA www.0voice.com
4407 48.454636    192.168.2.1      192.168.2.200    DNS     134 Standard query response 0x0003 AAAA www.0voice.com SOA dns1.hichina.com
```

其具体的流程可描述如下：

1. 主机 192.168.1.124 先向本地域名服务器 192.168.1.2 进行**递归查询**
2. 本地域名服务器采用**迭代查询**，向一个根域名服务器进行查询
3. 根域名服务器告诉本地域名服务器，下一次应该查询的顶级域名服务器 0voice.com 的 IP 地址
4. 本地域名服务器向顶级域名服务器 0voice.com 进行查询
5. 顶级域名服务器 .com 告诉本地域名服务器，下一步查询权限服务器 www.0voice.com 的 IP 地址
6. 本地域名服务器向权限服务器 www.0voice.com 进行查询
7. 权限服务器 www.0voice.com 告诉本地域名服务器所查询的主机的 IP 地址
8. 本地域名服务器最后把查询结果告诉 122.152.222.180

其中有两个概念递归查询和迭代查询，其实在整个描述的过程中已经体现的很明显，这里再说明一下：

- **递归查询**：本机向本地域名服务器发出一次查询请求，就静待最终的结果。如果本地域名服务器无法解析，自己会以 DNS 客户机的身份向其它域名服务器查询，直到得到最终的 IP 地址告诉本机
- **迭代查询**：本地域名服务器向根域名服务器查询，根域名服务器告诉它下一步到哪里去查询，然后它再去查，每次它都是以客户机的身份去各个服务器查询

协议报文格式

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）：

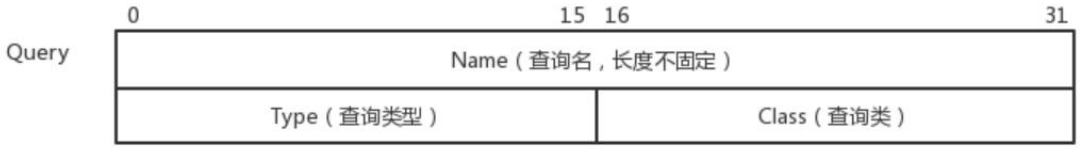
Flags	QR	opcode	AA	TC	RD	RA	(zero)	rcode
	1	4	1	1	1	1	3	4

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）

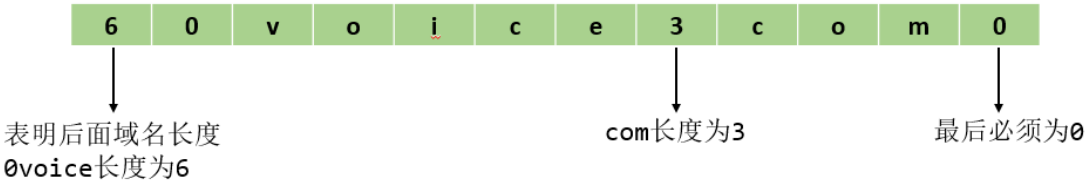
数量字段（总共 8 字节）：Questions、Answer RRs、Authority RRs、Additional RRs 各自表示后面的四个区域的数目。Questions 表示查询问题区域节的数量，Answers 表示回答区域的数量，Authoritative namesversers 表示授权区域的数量，Additional recoreds 表示附加区域的数量

正文

Queries



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

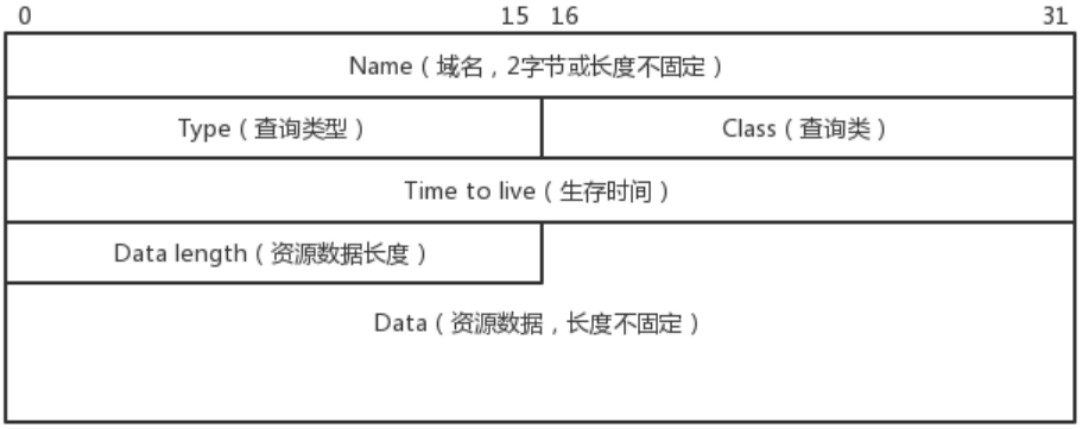
类型	助记符	说明
1	A	由域名获得 IPv4 地址
2	NS	查询域名服务器
5	CNAME	查询规范名称
6	SOA	开始授权
11	WKS	熟知服务
12	PTR	把 IP 地址转换成域名
13	HINFO	主机信息
15	MX	邮件交换
28	AAAA	由域名获得 IPv6 地址
252	AXFR	传送整个区的请求
255	ANY	对所有记录的请求

查询类

通常为 1，表明是 Internet 数据

源记录

源记录（RR）包括回答区域，授权区域和附加区域



资源记录格式

- **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

给出 wireshark 抓包的记录，感兴趣的可以根据上面介绍的协议报文格式手动解析一下，相信会有很大收获。

请求报文

```
Domain Name System (query)
    Transaction ID: 0x0002
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.0voice.com: type A, class IN
          Name: www.0voice.com
          [Name Length: 14]
          [Label Count: 3]
          Type: A (Host Address) (1)
          Class: IN (0x0001)
    [Response In: 4402]
```

```
0000  98 bb 99 06 07 39 ec f4  bb 4a a3 b2 08 00 45 00   .....9.. .J....E.
0010  00 3c 47 5b 00 00 80 11  00 00 c0 a8 02 c8 c0 a8   .<G[.... ........
0020  02 01 c4 63 00 35 00 28  86 53 00 02 01 00 00 01   ...c.5.( .S......
0030  00 00 00 00 00 00 03 77  77 77 06 30 76 6f 69 63   .......w ww.0voic
0040  65 03 63 6f 6d 00 00 01  00 01                     e.com... ..
```

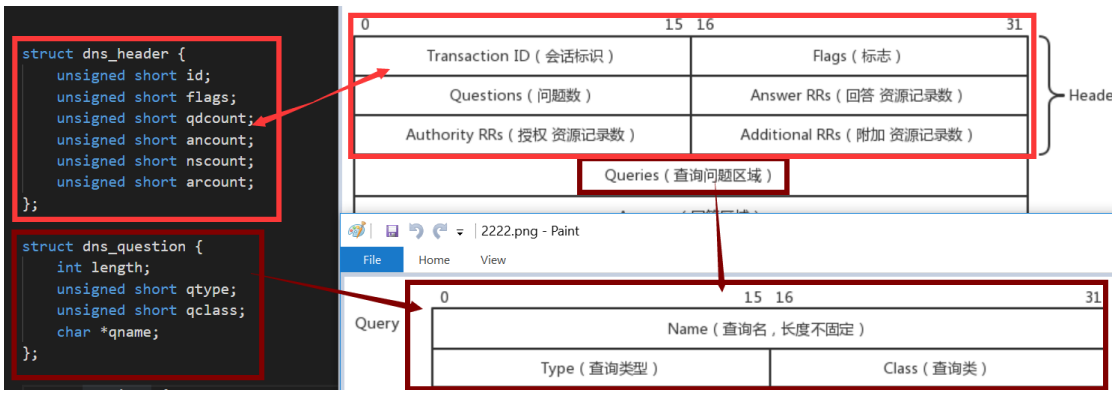
响应报文

```
Domain Name System (response)
    Transaction ID: 0x0002
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.0voice.com: type A, class IN
          Name: www.0voice.com
          [Name Length: 14]
          [Label Count: 3]
          Type: A (Host Address) (1)
          Class: IN (0x0001)
  v Answers
    v www.0voice.com: type A, class IN, addr 122.152.222.180
          Name: www.0voice.com
          Type: A (Host Address) (1)
          Class: IN (0x0001)
```

```
0000  ec f4 bb 4a a3 b2 98 bb  99 06 07 39 08 00 45 00   ...J.... ...9..E.
0010  00 4c 00 00 40 00 40 11  b4 87 c0 a8 02 01 c0 a8   .L..@.@. ........
0020  02 c8 00 35 c4 63 00 38  82 50 00 02 81 80 00 01   ...5.c.8 .P......
0030  00 01 00 00 00 00 03 77  77 77 06 30 76 6f 69 63   .......w ww.0voic
0040  65 03 63 6f 6d 00 00 01  00 01 c0 0c 00 01 00 01   e.com... ........
0050  00 00 01 30 00 04 7a 98  de b4                     ...0..z. ..
```

纯 c 实现 DNS 请求

数据格式



填充 header

```c
int dns_create_header(struct dns_header *header) {

    if (header == NULL) return -1;
    memset(header, 0, sizeof(struct dns_header));

    srandom(time(NULL));

    header->id = random();
    header->flags |= htons(0x0100);
    header->qdcount = htons(1);

    return 0;
}
```

填充 question

```c
int dns_create_question(struct dns_question *question, const char
*hostname) {

    if (question == NULL) return -1;
```

```
    memset(question, 0, sizeof(struct dns_question));

    question->qname = (char*)malloc(strlen(hostname) + 2);
    if (question->qname == NULL) return -2;

    question->length = strlen(hostname) + 2;

    question->qtype = htons(1);
    question->qclass = htons(1);

    const char delim[2] = ".";

    char *hostname_dup = strdup(hostname);
    char *token = strtok(hostname_dup, delim);

    char *qname_p = question->qname;

    while (token != NULL) {

        size_t len = strlen(token);

        *qname_p = len;
        qname_p ++;

        strncpy(qname_p, token, len+1);
        qname_p += len;

        token = strtok(NULL, delim);
    }

    free(hostname_dup);

    return 0;

}
```

准备请求头

```
int dns_build_request(struct dns_header *header, struct dns_question
*question, char *request) {

    int header_s = sizeof(struct dns_header);
    int question_s = question->length + sizeof(question->qtype) +
sizeof(question->qclass);

    int length = question_s + header_s;

    int offset = 0;
    memcpy(request+offset, header, sizeof(struct dns_header));
    offset += sizeof(struct dns_header);

    memcpy(request+offset, question->qname, question->length);
    offset += question->length;

    memcpy(request+offset, &question->qtype, sizeof(question->qtype));
    offset += sizeof(question->qtype);

    memcpy(request+offset, &question->qclass,
sizeof(question->qclass));

    return length;

}
```

解析 response

```
static void dns_parse_name(unsigned char *chunk, unsigned char *ptr,
char *out, int *len) {

    int flag = 0, n = 0, alen = 0;
    char *pos = out + (*len);
```

```
    while (1) {

        flag = (int)ptr[0];
        if (flag == 0) break;

        if (is_pointer(flag)) {

            n = (int)ptr[1];
            ptr = chunk + n;
            dns_parse_name(chunk, ptr, out, len);
            break;

        } else {

            ptr ++;
            memcpy(pos, ptr, flag);
            pos += flag;
            ptr += flag;

            *len += flag;
            if ((int)ptr[0] != 0) {
                memcpy(pos, ".", 1);
                pos += 1;
                (*len) += 1;
            }
        }

    }

}


static int dns_parse_response(char *buffer, struct dns_item **domains)
{

    int i = 0;
    unsigned char *ptr = buffer;

    ptr += 4;
```

```
    int querys = ntohs(*(unsigned short*)ptr);

    ptr += 2;
    int answers = ntohs(*(unsigned short*)ptr);

    ptr += 6;
    for (i = 0;i < querys;i ++) {
        while (1) {
            int flag = (int)ptr[0];
            ptr += (flag + 1);

            if (flag == 0) break;
        }
        ptr += 4;
    }

    char cname[128], aname[128], ip[20], netip[4];
    int len, type, ttl, datalen;

    int cnt = 0;
    struct dns_item *list = (struct dns_item*)calloc(answers,
sizeof(struct dns_item));
    if (list == NULL) {
        return -1;
    }

    for (i = 0;i < answers;i ++) {

        bzero(aname, sizeof(aname));
        len = 0;

        dns_parse_name(buffer, ptr, aname, &len);
        ptr += 2;

        type = htons(*(unsigned short*)ptr);
        ptr += 4;

        ttl = htons(*(unsigned short*)ptr);
        ptr += 4;
```

```
        datalen = ntohs(*(unsigned short*)ptr);
        ptr += 2;

        if (type == DNS_CNAME) {

            bzero(cname, sizeof(cname));
            len = 0;
            dns_parse_name(buffer, ptr, cname, &len);
            ptr += datalen;

        } else if (type == DNS_HOST) {

            bzero(ip, sizeof(ip));

            if (datalen == 4) {
                memcpy(netip, ptr, datalen);
                inet_ntop(AF_INET , netip , ip , sizeof(struct
sockaddr));

                printf("%s has address %s\n" , aname, ip);
                printf("\tTime to live: %d minutes , %d seconds\n", ttl
/ 60, ttl % 60);

                list[cnt].domain = (char *)calloc(strlen(aname) + 1,
1);
                memcpy(list[cnt].domain, aname, strlen(aname));

                list[cnt].ip = (char *)calloc(strlen(ip) + 1, 1);
                memcpy(list[cnt].ip, ip, strlen(ip));

                cnt ++;
            }

            ptr += datalen;
        }
    }

    *domains = list;
```

```
    ptr += 2;

    return cnt;

}
```

域名请求

```
int dns_client_commit(const char *domain) {

    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("create socket failed\n");
        exit(-1);
    }

    printf("url:%s\n", domain);

    struct sockaddr_in dest;
    bzero(&dest, sizeof(dest));
    dest.sin_family = AF_INET;
    dest.sin_port = htons(53);
    dest.sin_addr.s_addr = inet_addr(DNS_SVR);

    int ret = connect(sockfd, (struct sockaddr*)&dest, sizeof(dest));
    printf("connect :%d\n", ret);

    struct dns_header header = {0};
    dns_create_header(&header);

    struct dns_question question = {0};
    dns_create_question(&question, domain);

    char request[1024] = {0};
    int req_len = dns_build_request(&header, &question, request);
    int slen = sendto(sockfd, request, req_len, 0, (struct
sockaddr*)&dest, sizeof(struct sockaddr));
```

```
    char buffer[1024] = {0};
    struct sockaddr_in addr;
    size_t addr_len = sizeof(struct sockaddr_in);

    int n = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct
sockaddr*)&addr, (socklen_t*)&addr_len);

    printf("recvfrom n : %d\n", n);
    struct dns_item *domains = NULL;
    dns_parse_response(buffer, &domains);

    return 0;
}

//同时请求 50 个。

char *domain[] = {
    "www.ntytcp.com",
    "bojing.wang",
    "www.baidu.com",
    "tieba.baidu.com",
    "news.baidu.com",
    "zhidao.baidu.com",
    "music.baidu.com",
    "image.baidu.com",
    "v.baidu.com",
    "map.baidu.com",
    "baijiahao.baidu.com",
    "xueshu.baidu.com",
    "cloud.baidu.com",
    "www.163.com",
    "open.163.com",
    "auto.163.com",
    "gov.163.com",
    "money.163.com",
    "sports.163.com",
    "tech.163.com",
    "edu.163.com",
```

```
    "www.taobao.com",
    "q.taobao.com",
    "sf.taobao.com",
    "yun.taobao.com",
    "baoxian.taobao.com",
    "www.tmall.com",
    "suning.tmall.com",
    "www.tencent.com",
    "www.qq.com",
    "www.aliyun.com",
    "www.ctrip.com",
    "hotels.ctrip.com",
    "hotels.ctrip.com",
    "vacations.ctrip.com",
    "flights.ctrip.com",
    "trains.ctrip.com",
    "bus.ctrip.com",
    "car.ctrip.com",
    "piao.ctrip.com",
    "tuan.ctrip.com",
    "you.ctrip.com",
    "g.ctrip.com",
    "lipin.ctrip.com",
    "ct.ctrip.com"
};

int main(int argc, char *argv[]) {

    int count = sizeof(domain) / sizeof(domain[0]);
    int i = 0;

    for (i = 0;i < count;i ++) {
        dns_client_commit(domain[i]);
    }

    getchar();
}
```

思考实践

实现异步 DNS 请求？