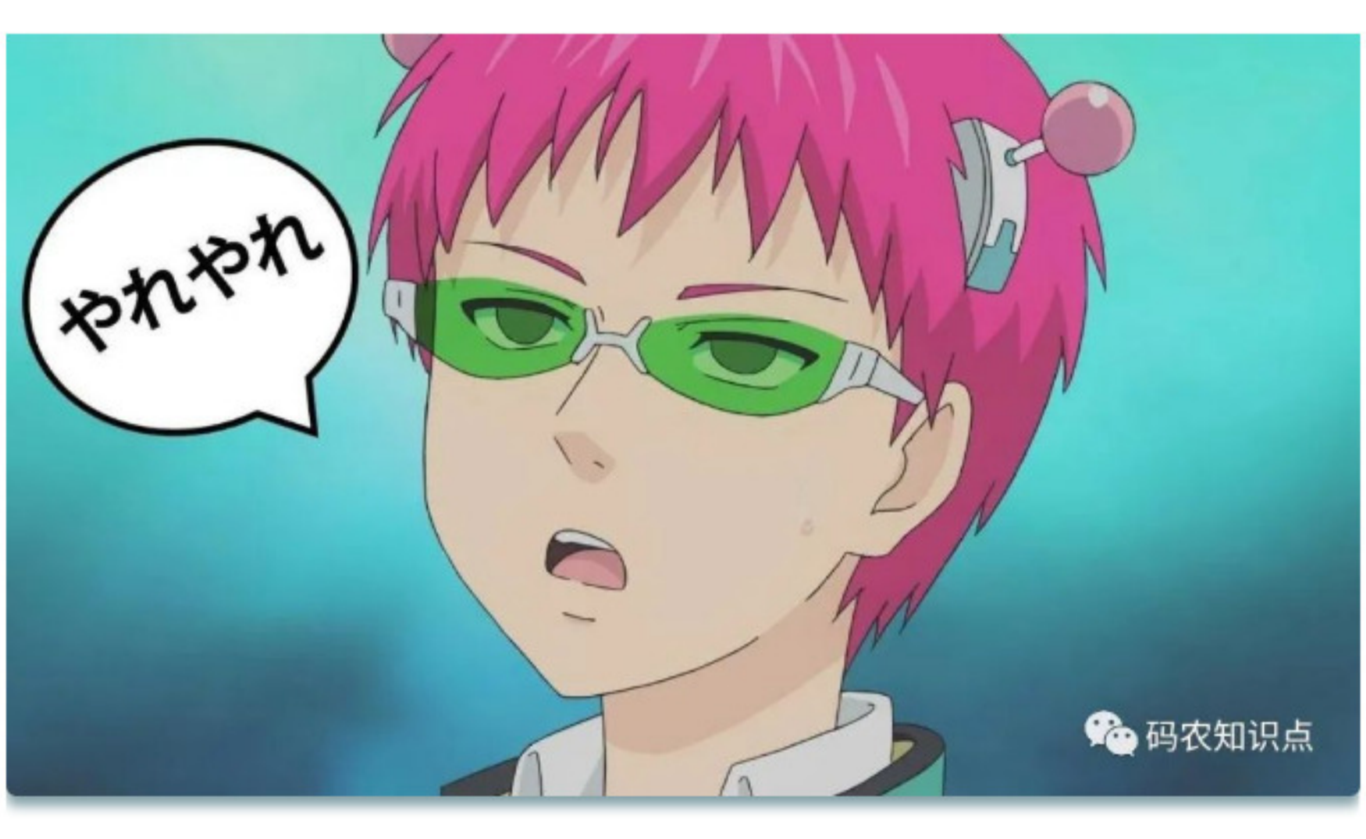
# 面试官：你真的了解127.0.0.1和0.0.0.0吗～

程序员面试 7月5日

以下文章来源于码农知识点 ，作者Monica23334



事出有因，前段时间老大让小姐姐在测试环境搭建一个ELK。我说我搭好了，但Kibana端口不知为啥没暴露出去，其他机子访问不了我的Kibana但确可以ping通这台机子...一个小伙伴马上用netstat命令确定到我把Kibana IP绑到127.0.0.1上了,然后我就收到了组内一堆 " ..... " 的回复。。

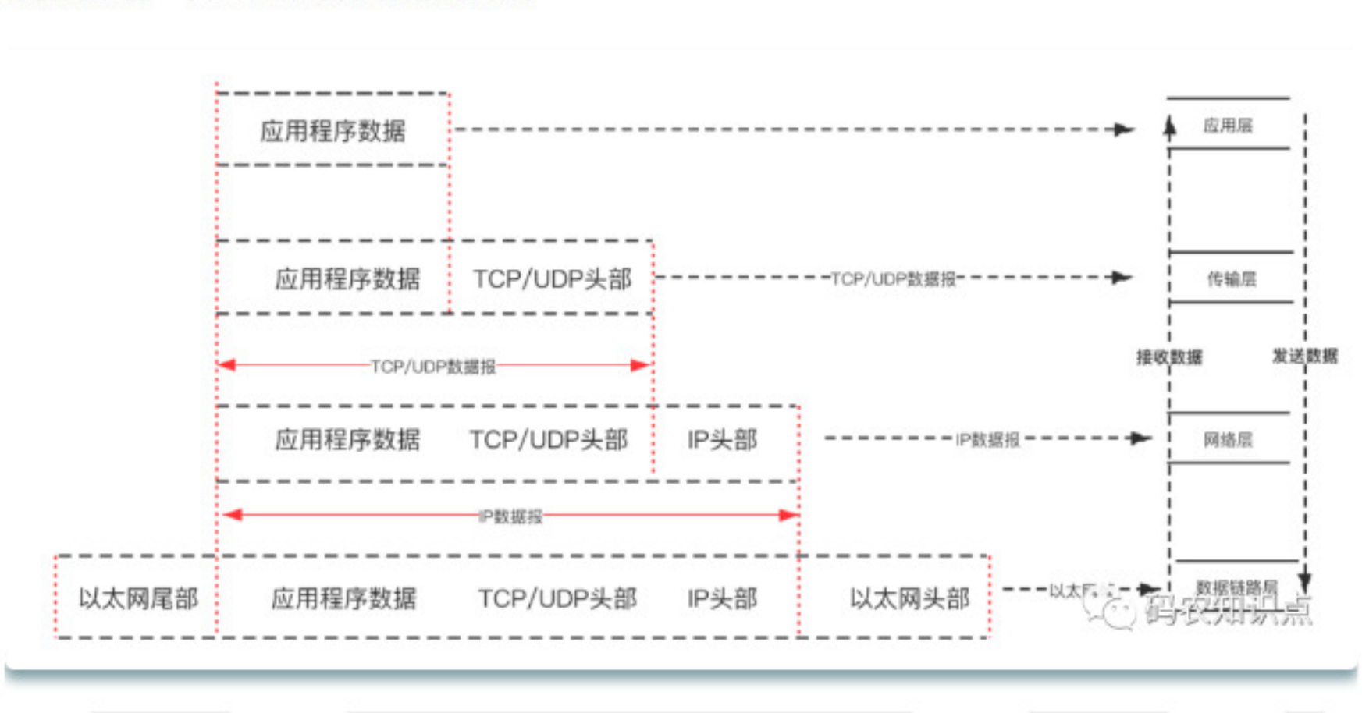
幸好阿姨带着口罩上班，不然他们可能看到我没洗的小红脸（😊。。嗯，做人就是要脸皮厚）。。

在了解127.0.0.1和0.0.0.0是啥之前（因为这个问题实在太常识了），我们先看一下为啥会有IP。

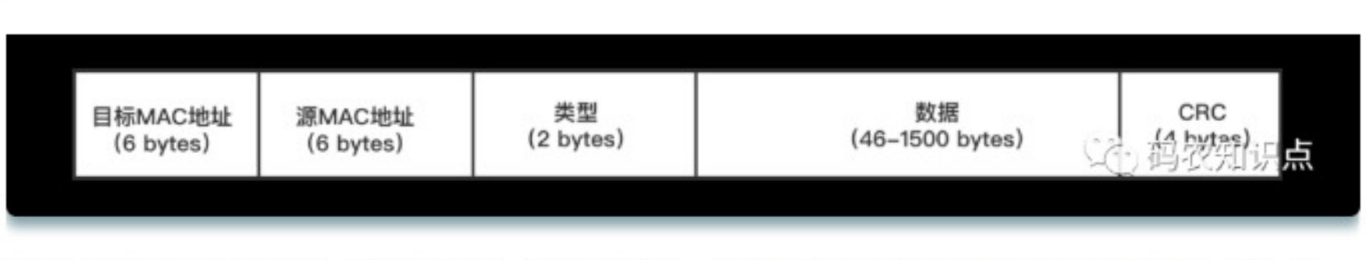
网络传输需要解决的主要问题有两个： **发给谁** 和 **通过怎么样的路线才能发给他（也就是路由）。**

## IP的作用

我们先来看一下TCP/IP封装的数据包结构：



当我们 `发送数据` 时，按照 `应用层 —>数据链路层自上而下封装数据包` 。当我们 `接收数据` 时，按照 `数据链路层 —>应用层 进行拆包` 。这里我们再来看一下数据链路层 `以太网帧` 的结构：



| | | | | |
|---|---|---|---|---|
| 目标MAC地址 (6 bytes) | 源MAC地址 (6 bytes) | 类型 (2 bytes) | 数据 (46–1500 bytes) | CRC (4 bytes) |

每个网卡在出厂时就会有一个全世界唯一的MAC地址，就相当于是我们每个人在世界上都是唯一的。那 `MAC地址` 其实就能解决第一个 `发给谁` 的问题。那为啥还需要IP层呢?

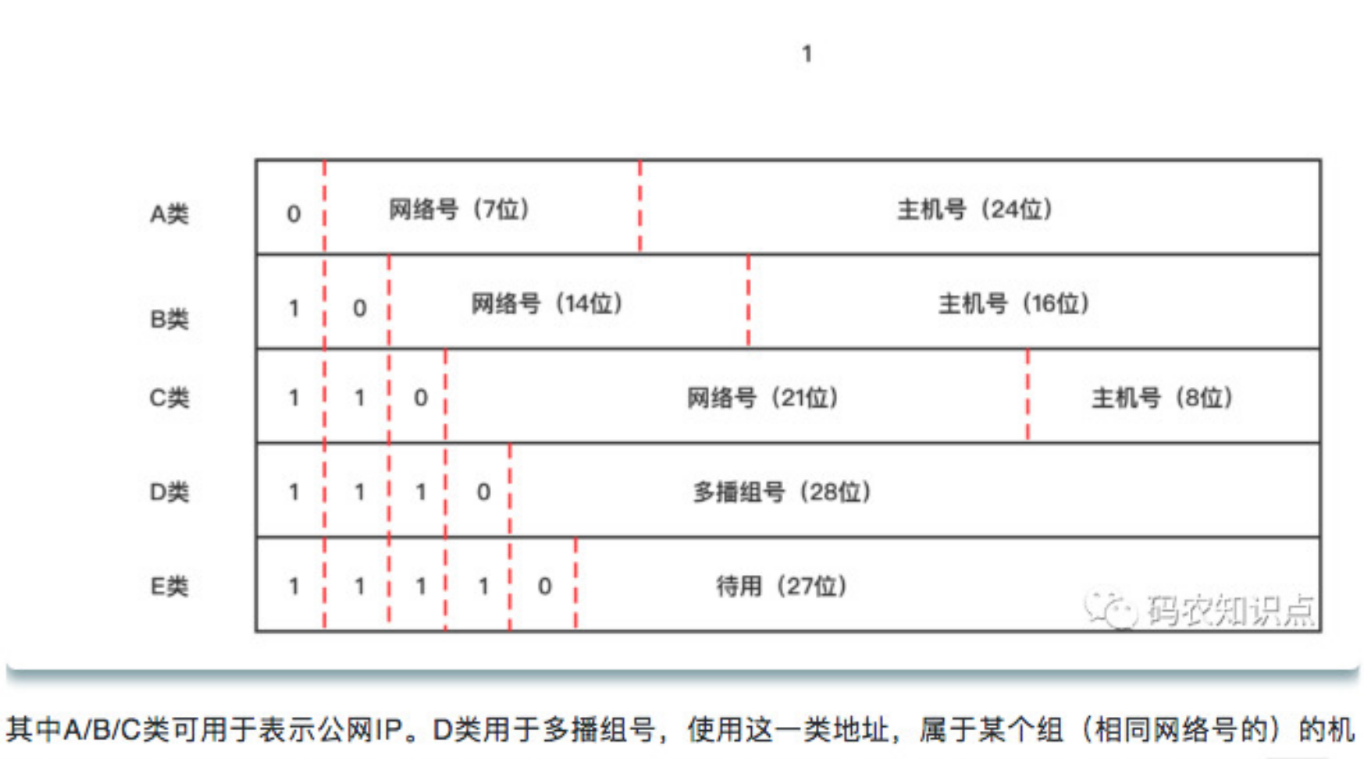
想象你到世界上找一个叫Monica2333的人，你不可能大街上随便逮到一个人就问你认识Monica2333吗..（当然如果你真这么干了，请关注公众号 码农知识点，小姐姐在这里等你～） 你应该先搞清楚Monica2333住址是在哪个国家哪个城市哪个区哪个街道，从国家-》城市-》区-》小区一步步定位，这么找人就符合常理了。。 `IP解决的` 就是网络上 `数据包的路由定位困难问题` 。

## IP的定义

IP怎么表达出"国家城市区"的概念呢，这就是IP地址定义的事了。

**IP** 地址用32位来表示，通常被分割为4个"8位"。以"点分十进制"表示成 [a.b.c.d] 的形式。同时这32bit又被划分为 `网络号` 和 `主机号` 组成。比如10.100.122.2/24 这个IP：10.100.122.2是"点分十进制"表示形式，/24 表示的是前24位是网络号，后8位是主机号。这个 `网络号` 其实充当的就是 `"国家城市区"` 的概念。比如我们只需要记住怎么去往10.100.122.x，就知道怎么去找10.100.122.1/24和10.100.122.2/24了。为了方便获取网络号 ，又出现一个 `子网掩码` 的概念。子网掩码就是 `网络号位上全为1，主机号全为0` 的IP地址。这样当 **IP&子网掩码**得到的就是IP的网络号。所以**10.100.122.2/24**的子网掩码就是**255.255.255.0**。

好了，我们来看一下IPv4对IP地址的划分：

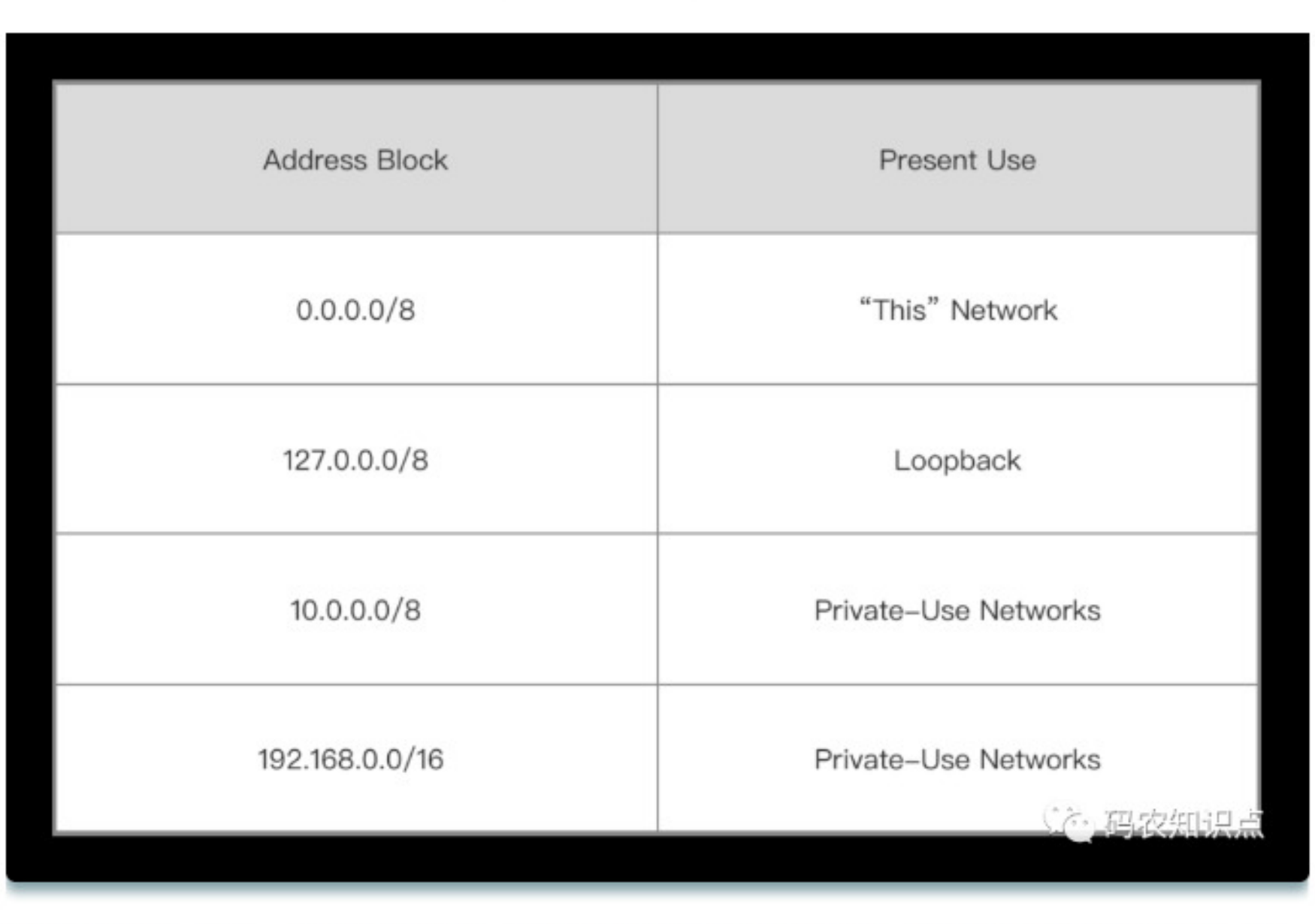


| | | | | | | |
|---|---|---|---|---|---|---|
| A类 | 0 | 网络号（7位） | 主机号（24位） | | | |
| B类 | 1 | 0 | 网络号（14位） | 主机号（16位） | | |
| C类 | 1 | 1 | 0 | 网络号（21位） | 主机号（8位） | |
| D类 | 1 | 1 | 1 | 0 | 多播组号（28位） | |
| E类 | 1 | 1 | 1 | 1 | 0 | 待用（27位） |

其中A/B/C类可用于表示公网IP。D类用于多播组号，使用这一类地址，属于某个组（相同网络号的）的机器都能收到，E类还留待使用。但是我们并不用记住公网IP是属于A/B/C类中哪一类，我们只需要用 `/24或/16这样的CIDR` 方式去区分IP的网络号和主机号就可以。

实际上，A/B/C类划分的IP显然是不够用每个地球人用的，我们日常在接入公网时，都需要走能有公网IP的网关。在整个公网内部，再分配私有IP地址给每个人上网使用就可以了。比如家庭常用的192.168.0.x /24 私有IP网段。

此外，在这五类IP地址基础上还划分出了 `特殊的IP网段` 。



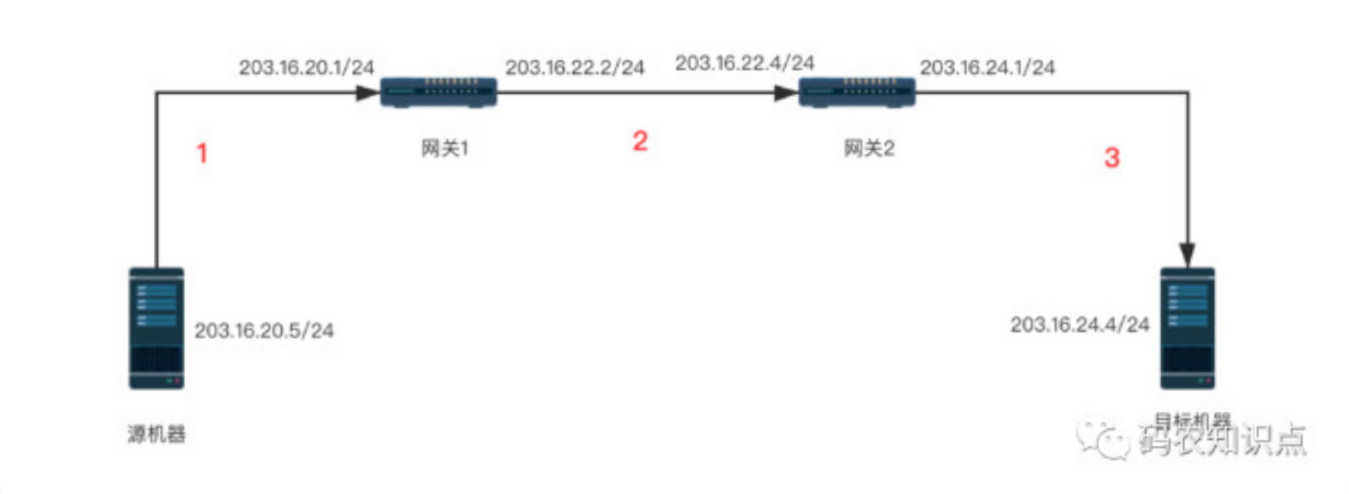
| Address Block | Present Use |
|---|---|
| 0.0.0.0/8 | "This" Network |
| 127.0.0.0/8 | Loopback |
| 10.0.0.0/8 | Private-Use Networks |
| 192.168.0.0/16 | Private-Use Networks |

我看心情选了一部分。小伙伴也发现了，127.0.0.1和0.0.0.0终于出现了，但我们现在先不讲他们俩。我们再回到开始的第二个问题： `通过怎么样的路线才能发给他` 。

## IP路由

上面我们说了IP的目的就是将路由简单化，IP的网络号承担了"一组IP 的路由"出入口的作用。实际上 `网关` 就是这个出入口。不同局域网（IP网络号不同）的网络通信必须经过网关，相同局域网的网络通信可以靠**广播和MAC地址**来送达目标机器（也就是二层协议行的通）。

好了，假使我们从203.16.20.5/24 —> 203.16.24.4/24 发一个包，其中源IP和目标IP都是公网IP。那包的路由过程如下：



源机器网络程序在封装数据包的过程中发现目标机器和自己的IP不在同一局域网内，则需要通过网关将包从网卡发出去。 `1处` 的数据包结构为：

| | | | | | |
|---|---|---|---|---|---|
| 源MAC：203.16.20.5 MAC | 目标 MAC：203.16.20.5 MAC | | 源IP：203.16.20.5 | 目标机器 IP：203.16.24.4 | |

到了网关1，拆包发现目标IP地址是 203.16.24.4/24。查了下自己的**路由表**，发现要想访问203.16.24.4/24，要从 203.16.22.2/24 这个口出去，下一跳为 203.16.22.4/24。此时 `2处` 的数据包结构为：

| | | | | | |
|---|---|---|---|---|---|
| 源MAC：203.16.22.2 MAC | 目标 MAC：203.16.22.4 MAC | | 源IP：203.16.20.5 | 目标机器 IP：203.16.24.4 | |

到了网关2，拆包发现目标IP地址是 203.16.24.4/24。查了下自己的路由表，发现要想访问203.16.24.4/24，要从 203.16.24.1/24 这个口出去。此时 `3处` 的数据包结构为：

| | | | | | |
|---|---|---|---|---|---|
| 源MAC：203.16.24.1 MAC | 目标 MAC：203.16.24.4 MAC | | 源IP：203.16.20.5 | 目标机器 IP：203.16.24.4 | |

到了目标机器，拆包发现目标IP地址就是自己呀，所以进行更上层的拆包，把数据收进来就可以了。

我们可以发现这种方式的路由每经过一次局域网，MAC地址需要改变，但 `IP地址不需要改变` 。这种网关称为 `转发网关` 。而实际上还存在一种 `改变 IP 地址` 的网关，称为 `NAT 网关` 。我们就不展开了，小伙伴可以自行搜索。

现在我们知道**IP/MAC地址/网关**等怎么解决上述两个 发给谁 和 通过怎么样的路线才能发给目标机器 的问题了（当然还有路由策略等内容没有展开，我们这里只讨论如果有路由路线，咋发的问题）。但特么127.0.0.1和0.0.0.0到底有啥特殊的啊。。。

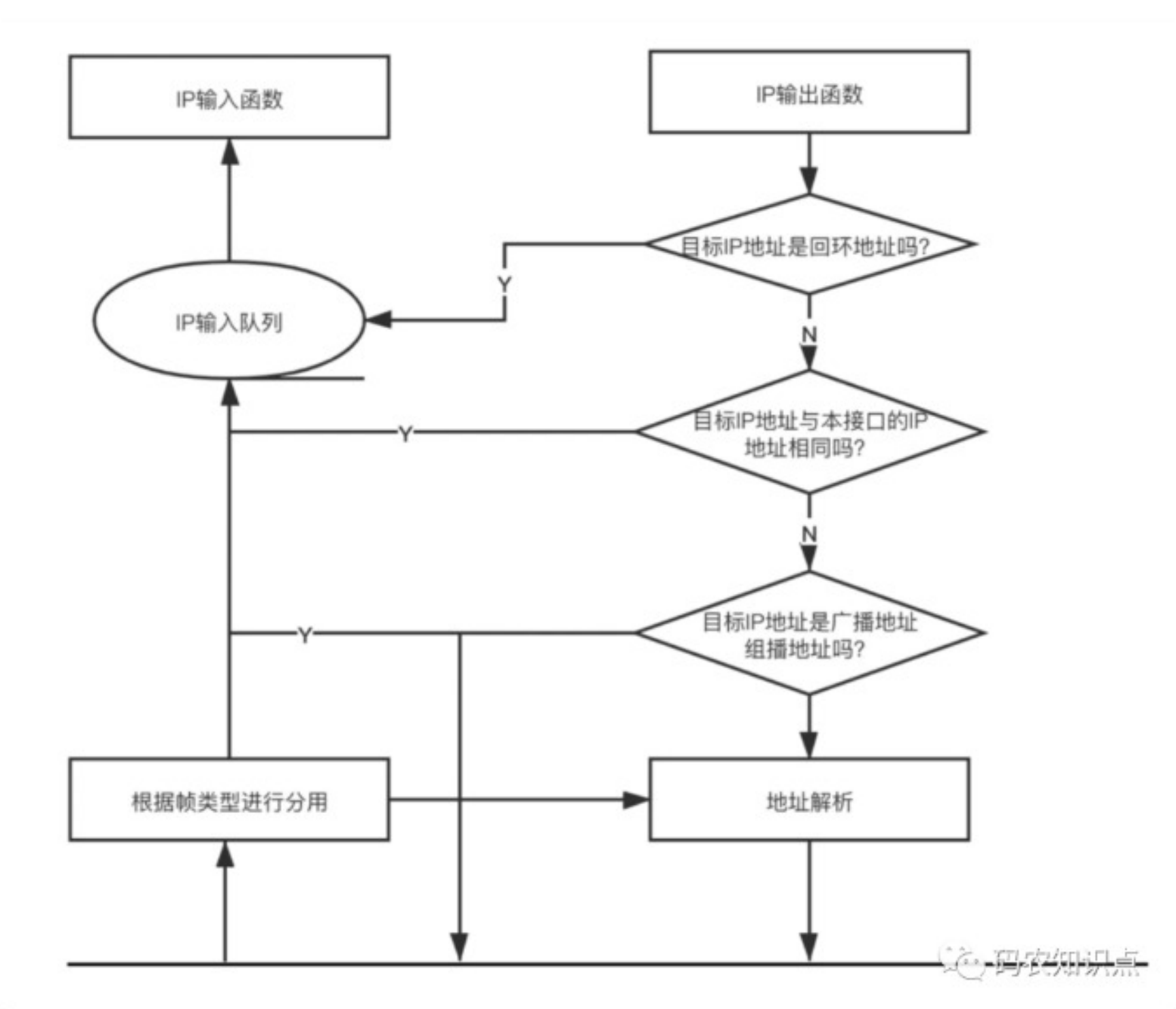
## 127.0.0.1 & 0.0.0.0

终于要切入正题了。。

**127.0.0.1**

从上面 特殊的IP网段中我们可以知道 `127.0.0.1` 表示的是 `回环IP地址（loopback address）` 。啥意思呢？所有发往目标IP为 `127.0.0.1` 的数据包都不会通过网卡发送到网络上，而是在数据离开网络层时将其回送给本机的有关进程。

形象些说就是因为发送数据包是从应用层 —>数据链路层自上而下一层层按照程序封装的，当到了网络层时，发现目标IP是127.0.0.1，就不会再往下封装数据链路层了，而是把包又丢给需要往上层解析的队列中了。



实际上 `localhost` 通常也代表127.0.0.1。这是因为通常在**本机Hosts文件**会把localhost映射为127.0.0.1 。此外 `以127开头` 的IP地址都是回环地址，只是我们通常使用127.0.0.1。所以这只能在本机来回收发包的地址有啥用呢？**本机测试用！！**

**0.0.0.0**

0.0.0.0这个IP地址指的是 `没有路由的元地址` ，通常被用来表示 `无效的，未知的` 或是 `没有指定目标IP的地址` 。看不懂没关系，它其实相当于Java中的 `this` ，真表示啥要放到实际所处环境中去考虑。**用处**主要有：

- 本机所有IP

  当考虑它在一台服务器中的作用时，它指代的就是 `这台机器上所有的IP` 。假如一台机器上有两个IP:203.16.20.5/24 和 203.16.24.4/24。如果我们把一个Java应用的IP绑定到了0.0.0.0:8080，那访问203.16.20.5:8080 和 203.16.24.4:8080都可以与这个Java应用建立连接。

- 默认路由

上面讲IP路由的时候我们提到了路由表。路由表就是一个 `记录数据包下一跳应该去哪` 的路由规则。每一条规则至少包含三项信息：

`网络ID：` 就是目标地址的网络ID。

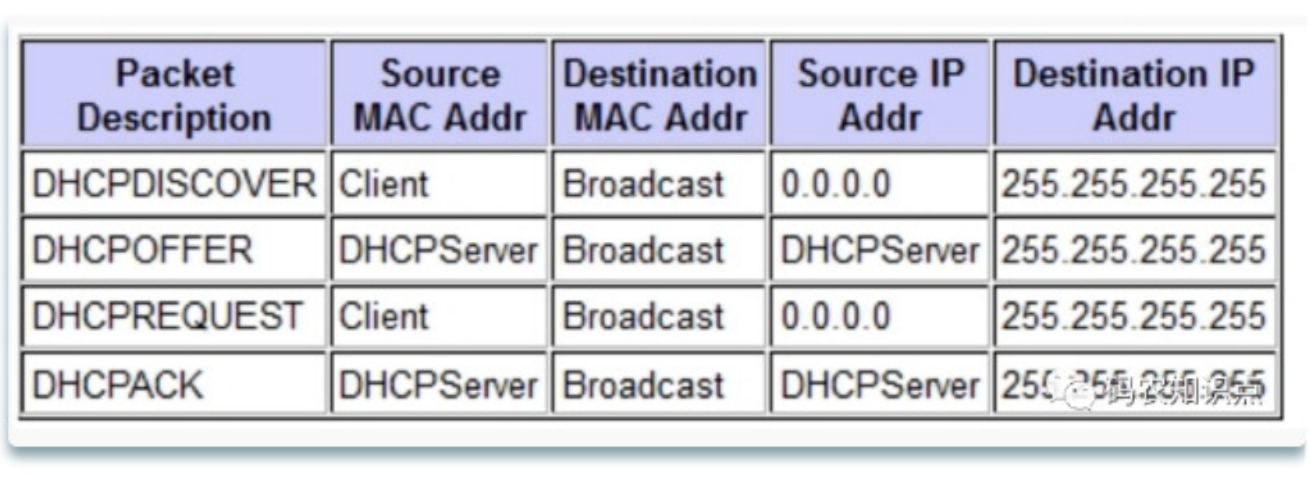
`子网掩码：` 用来判断IP所属网络。

`下一跳地址/接口` ：就是数据在发送到目标地址的旅途中下一站的地址。

假设一个IP匹配了多条路由规则，则子网越小的越优先，也就是/n这样的CIDR越大的越优先。而假如配置了0.0.0.0/0 -> via 111.222.1.254这样的路由策略,表示的是当解析不到任何精确的路由规则时，下一跳就统统跳往111.222.1.254。0.0.0.0在这里就是 `默认路由` 的意思。

- DHCP

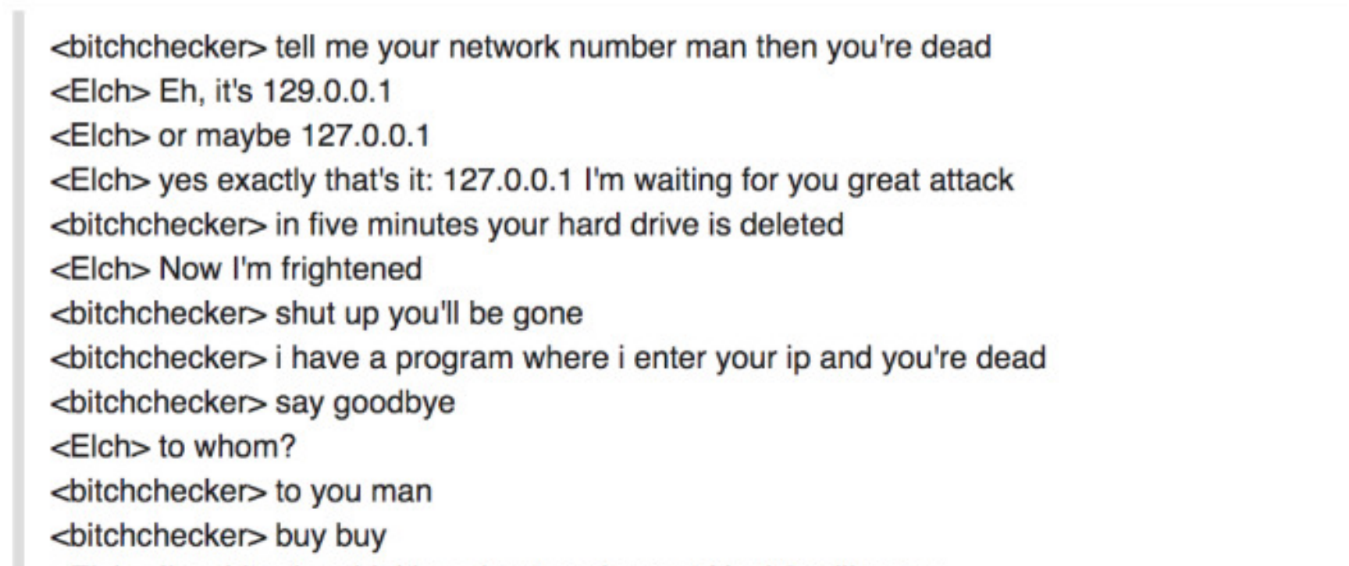
  当一个网络设备初次启动时，假如没有配置IP。它需要通过DHCP协议向所处局域网要一个IP。但DHCP是建立在UDP 上的协议，没IP咋发包要IP啊。这里 `0.0.0.0/0` 的作用就是在 `没要到IP时所使用的源IP` 。放张协议图，感兴趣的小伙伴可进一步了解：



| Packet Description | Source MAC Addr | Destination MAC Addr | Source IP Addr | Destination IP Addr |
|---|---|---|---|---|
| DHCPDISCOVER | Client | Broadcast | 0.0.0.0 | 255.255.255.255 |
| DHCPOFFER | DHCPServer | Broadcast | DHCPServer | 255.255.255.255 |
| DHCPREQUEST | Client | Broadcast | 0.0.0.0 | 255.255.255.255 |
| DHCPACK | DHCPServer | Broadcast | DHCPServer | 255.255.255.255 |

为了首尾呼应，小姐姐再讲一个网上看到的一个不那么冷的笑话吧。。

bitchcheker是一个扬言要攻击Elch的黑客：



```
<bitchchecker> tell me your network number man then you're dead
<Elch> Eh, it's 129.0.0.1
<Elch> or maybe 127.0.0.1
<Elch> yes exactly that's it: 127.0.0.1 I'm waiting for you great attack
<bitchchecker> in five minutes your hard drive is deleted
<Elch> Now I'm frightened
<bitchchecker> shut up you'll be gone
<bitchchecker> i have a program where i enter your ip and you're dead
<bitchchecker> say goodbye
<Elch> to whom?
<bitchchecker> to you man
<bitchchecker> buy buy
<Elch> I'm shivering thinking about such great Hack0rs like you
• bitchchecker (~java@euirc-61a2169c.dip.t-dialin.net) Quit (Ping timeout#)
```

至此你有没有更了解127.0.0.1呀～

参考资料：

https://www.zhihu.com/question/20717354
https://time.geekbang.org/column/article/8590 https://www.tech-faq.com/127-0-0-1.html
https://www.howtogeek.com/225487/what-is-the-difference-between-127.0.0.1-and-0.0.0.0/
https://blog.csdn.net/qq_38410730/article/details/80980749
https://knowyourmeme.com/memes/hack-127001