硬核图解！断网了，还能ping通 127.0.0.1 吗？为什么？

Original 小白 小白debug 2021-06-15 18:00

收录于合集
#图解网络

17个

你**女神爱不爱你**，你问她，她可能不会告诉你。

但**网通不通**，你 `ping` 一下就知道了。

可能看到标题，你就知道答案了，但是你了解背后的原因吗？那如果把 `127.0.0.1` 换成 `0.0.0.0` 或 `localhost` 会怎么样呢？你知道这几个 `IP` 有什么区别吗？

以前面试的时候就遇到过这个问题，大家看个动图了解下面试官和我当时的场景，求当时小白的心里阴影面积。

话不多说，我们直接开车。

拔掉网线，断网。

然后在控制台输入 `ping 127.0.0.1` 。

```
$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.080 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.093 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.079 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.079 ms
^C
--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.074/0.081/0.093/0.006 ms
```

说明，拔了网线， `ping 127.0.0.1` 是**能ping通的**。

其实这篇文章看到这里，标题前半个问题已经被回答了。但是我们可以再想深一点。

为什么断网了还能 `ping` 通 `127.0.0.1` 呢？

这能说明你不用交网费就能上网吗？

不能。

首先我们需要进入基础科普环节。

不懂的同学看了就懂了，懂的看了就当查漏补缺吧。

什么是127.0.0.1

首先，这是个 IPV4 地址。

IPV4 地址有 32 位，一个字节有 8 位，共 4 个字节。

其中**127 开头的都属于回环地址**，也是 IPV4 的特殊地址，没什么道理，就是人为规定的。

而 127.0.0.1 是**众多**回环地址中的一个。之所以不是 127.0.0.2 ，而是 127.0.0.1 ，是因为源码里就是这么定义的，也没什么道理。

```
/* Address to loopback in software to local host.  */
#define    INADDR_LOOPBACK     0x7f000001  /* 127.0.0.1   */
```

回环地址

IPv4 的地址是 32 位的，2的32次方，大概是 40+亿 。地球光人口就76亿了，40亿IP这点量，**塞牙缝都不够**，实际上**IP也确实用完**了。

所以就有了 IPV6 ， IPv6 的地址是 128 位的，大概是2的128次方≈**10的38次方**。据说地球的沙子数量大概是 **10的23次方**，所以IPV6的IP可以认为用不完。

IPV4以8位一组，每组之间用 . 号隔开。

IPV6就以16位为一组，每组之间用 : 号隔开。如果全是0，那么可以省略不写。

ipv6回环地址

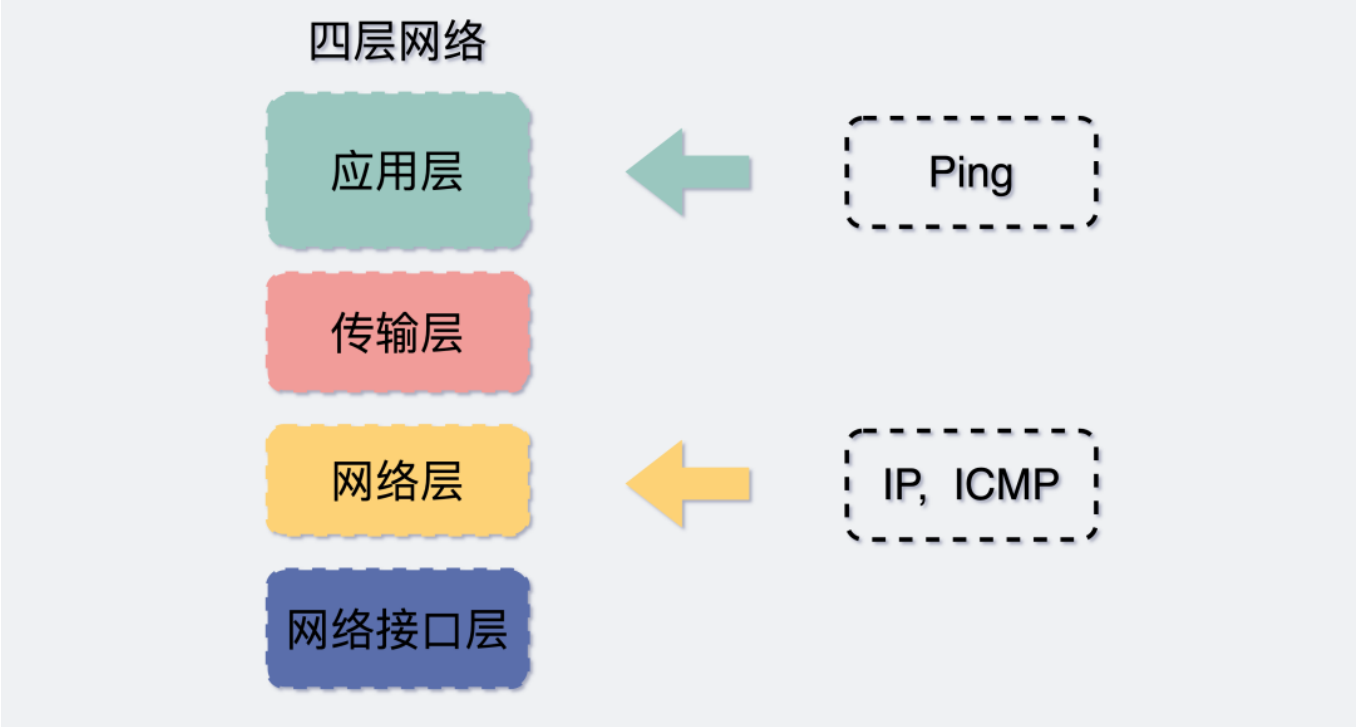
在IPV4下的回环地址是 `127.0.0.1`，在 `IPV6` 下，表达为 `::1` 。中间把**连续的0**给省略了，之所以不是**7个 冒号**，而是**2个冒号:** ， 是因为一个 IPV6 地址中**只允许出现一次两个连续的冒号**。

> 多说一句：在IPV4下用的是 **ping 127.0.0.1** 命令。在IPV6下用的是 **ping6 ::1** 命令。

什么是 ping

ping 是应用层命令，可以理解为它跟游戏或者聊天软件属于同一层。只不过聊天软件可以收发消息，还能点个赞什么的，有很多复杂的功能。而 ping 作为一个小软件，它的功能比较简单，就是**尝试**发送一个小小的消息到目标机器上，判断目的机器是否**可达**，其实也就是判断目标机器网络是否能连通。

ping应用的底层，用的是网络层的**ICMP协议**。



IP和ICMP和Ping所在分层

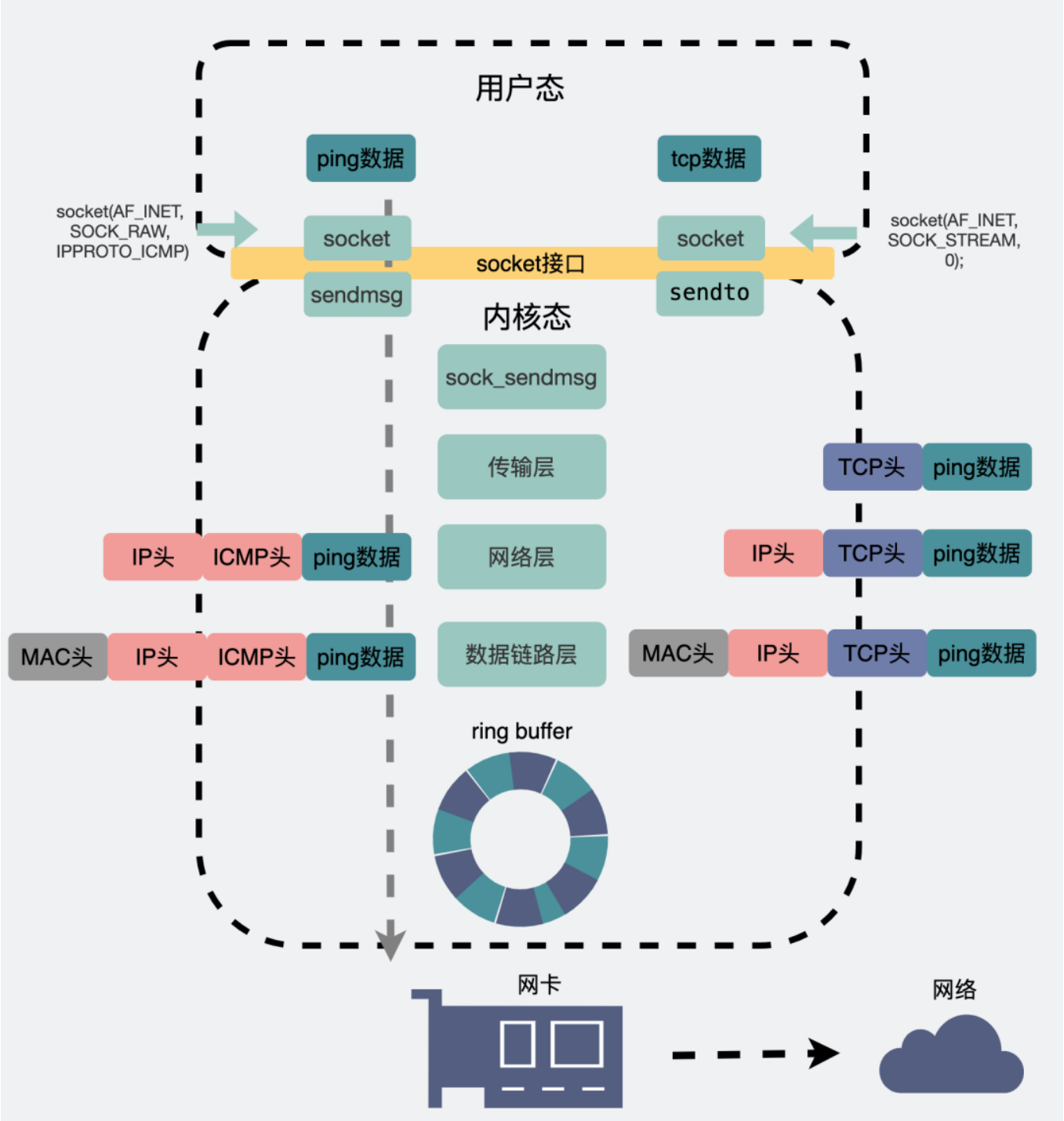
虽然ICMP协议和IP协议**都属于网络层协议**，但其实**ICMP也是利用了IP协议进行消息的传输**。

ip和icmp的关系

所以，大家在这里完全可以简单的理解为 ping 某个IP 就是往某个IP地址发个消息。

TCP发数据和ping的区别

一般情况下，我们会使用 TCP 进行网络数据传输，那么我们可以看下它和 ping 的区别。



ping和普通发消息的关系

ping和其他应用层软件都属于**应用层**。

那么我们横向对比一下，比方说聊天软件，如果用的是TCP的方式去发送消息。

为了发送消息，那就得先知道往哪发。linux里万物皆文件，那你要发消息的目的地，也是个文件，这里就引出了socket 的概念。

要使用 `socket` ，那么首先需要创建它。

在 TCP 传输中创建的方式是 `socket(AF_INET, SOCK_STREAM, 0);` ，其中 `AF_INET` 表示将使用 IPV4 里 **host:port** 的方式去解析待会你输入的网络地址。`SOCK_STREAM` 是指使用面向字节流的 TCP 协议，**工作在传输层**。

创建好了 `socket` 之后，就可以愉快的把要传输的数据写到这个文件里。调用 socket 的 `sendto` 接口的过程中进程会从**用户态进入到内核态**，最后会调用到 `sock_sendmsg` 方法。

然后进入传输层，带上 `TCP` 头。网络层带上 `IP` 头，数据链路层带上 `MAC` 头等一系列操作后。进入网卡的**发送队列 ring buffer** ，顺着网卡就发出去了。

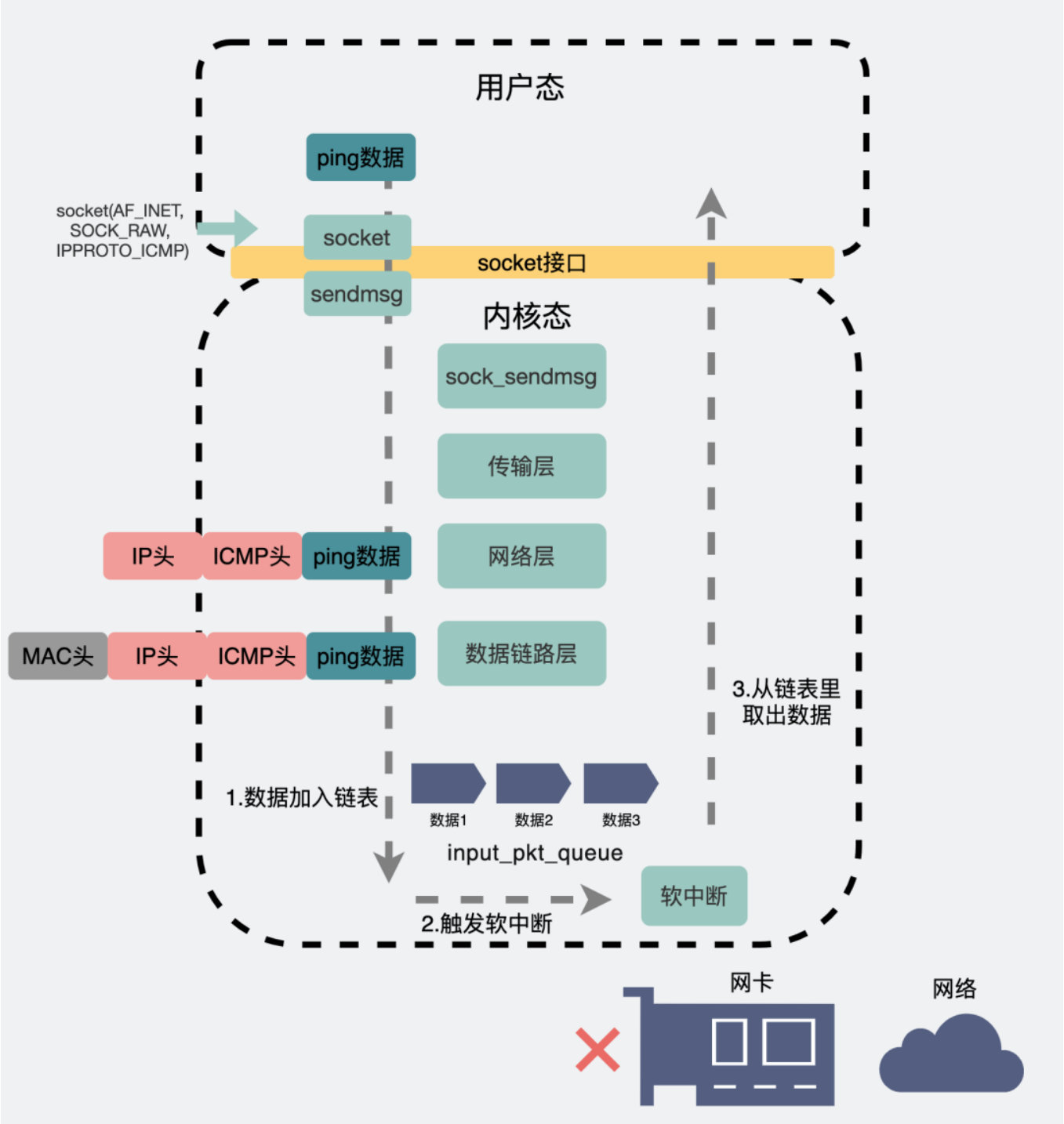
回到 `ping` ， 整个过程也基本跟 `TCP` 发数据类似，差异的地方主要在于，创建 `socket` 的时候用的是 `socket(AF_INET,SOCK_RAW,IPPROTO_ICMP)` ，`SOCK_RAW` 是原始套接字 ，**工作在网络层**， 所以构建 `ICMP` （网络层协议）的数据，是再合适不过了。ping 在进入内核态后最后也是调用的 `sock_sendmsg` 方法，进入到网络层后加上**ICMP和IP头**后，数据链路层加上**MAC头**，也是顺着网卡发出。因此 本质上ping 跟 普通应用发消息 在程序流程上没太大差别。

这也解释了**为什么当你发现怀疑网络有问题的时候，别人第一时间是问你能ping通吗？**因为可以简单理解为ping就是自己组了个数据包，让系统按着其他软件发送数据的路径往外发一遍，能通的话说明其他软件发的数据也能通。

为什么断网了还能 ping 通 127.0.0.1

前面提到，有网的情况下，ping 最后是**通过网卡**将数据发送出去的。

那么断网的情况下，网卡已经不工作了，ping 回环地址却一切正常，我们可以看下这种情况下的工作原理。



ping回环地址

从应用层到传输层再到网络层。这段路径跟ping外网的时候是几乎是一样的。到了网络层，系统会根据目的IP，在路由表中获取对应的**路由信息**，而这其中就包含选择**哪个网卡**把消息发出。

当发现**目标IP是外网IP**时，会从"真网卡"发出。

当发现**目标IP是回环地址**时，就会选择**本地网卡**。

本地网卡，其实就是个**"假网卡"**，它不像"真网卡"那样有个 `ring buffer` 什么的，"假网卡"会把数据推到一个叫 `input_pkt_queue` 的 **链表** 中。这个链表，其实是所有网卡共享的，上面挂着发给本机的各种消息。消息被发送到这个链表后，会再触发一个**软中断**。

专门处理软中断的工具人**"ksoftirqd"** （这是个**内核线程**），它在收到软中断后就会立马去链表里把消息取出，然后顺着数据链路层、网络层等层层往上传递最后给到应用程序。

工具人ksoftirqd

ping 回环地址和**通过TCP等各种协议发送数据到回环地址**都是走这条路径。整条路径从发到收，都没有经过"真网卡"。**之所以127.0.0.1叫本地回环地址，可以理解为，消息**

发出到这个地址上的话，就不会出网络，在本机打个转就又回来了。所以断网，依然能 `ping` 通 `127.0.0.1` 。

ping回环地址和ping本机地址有什么区别

我们在mac里执行 `ifconfig` 。

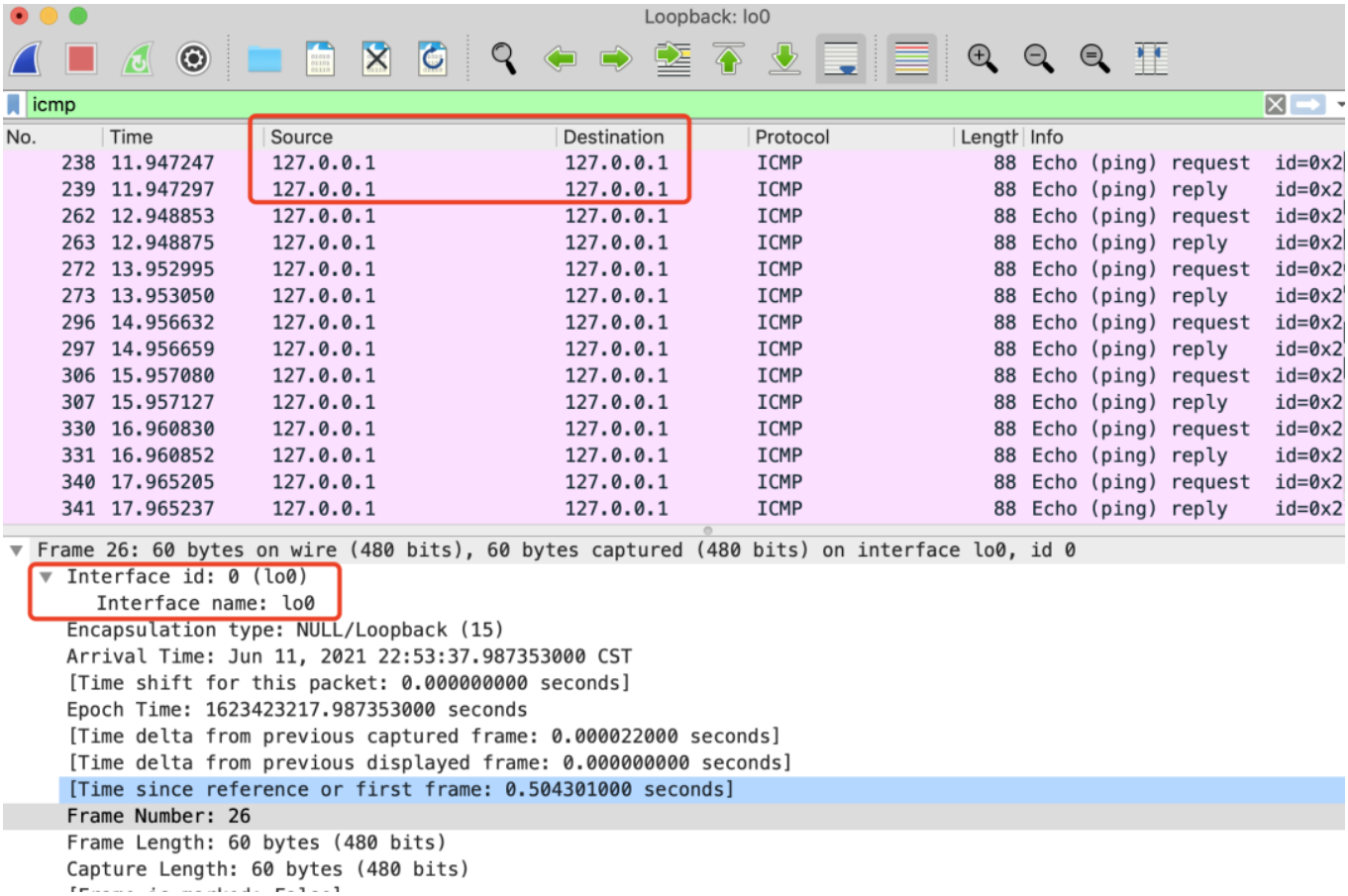
```
$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet 127.0.0.1 netmask 0xff000000
    ...
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.31.6 netmask 0xffffff00 broadcast 192.168.31.255
    ...
```

能看到 **lo0**，表示本地回环接口，对应的地址，就是我们前面提到的 **127.0.0.1** ，也就是**回环地址**。

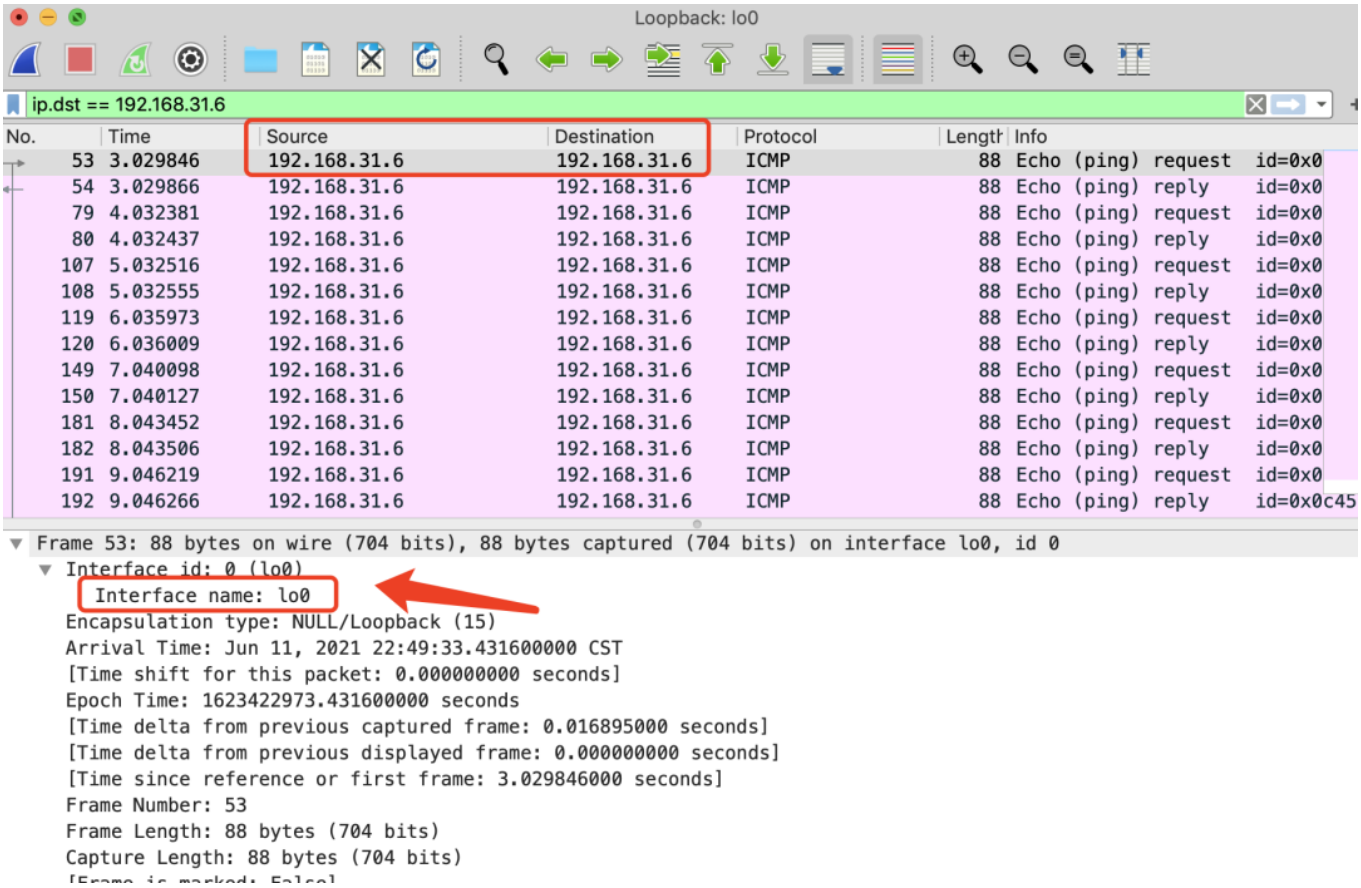
和 **eth0**，表示本机第一块网卡，对应的IP地址是**192.168.31.6**，管它叫**本机IP**。

之前一直认为ping本机IP的话会通过"真网卡"出去，然后遇到第一个路由器，再发回来到本机。

为了验证这个说法，可以进行抓包，但结果跟上面的说法并不相同。


Loopback: lo0
icmp
No. Time Source Destination Protocol Length Info
238 11.947247 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) request id=0x2
239 11.947297 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) reply id=0x2
262 12.948853 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) request id=0x2
263 12.948875 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) reply id=0x2
272 13.952995 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) request id=0x2
273 13.953050 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) reply id=0x2
296 14.956632 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) request id=0x2
297 14.956659 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) reply id=0x2
306 15.957080 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) request id=0x2
307 15.957127 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) reply id=0x2
330 16.960830 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) request id=0x2
331 16.960852 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) reply id=0x2
340 17.965205 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) request id=0x2
341 17.965237 127.0.0.1 127.0.0.1 ICMP 88 Echo (ping) reply id=0x2
Frame 26: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface lo0, id 0
Interface id: 0 (lo0)
Interface name: lo0
Encapsulation type: NULL/Loopback (15)
Arrival Time: Jun 11, 2021 22:53:37.987353000 CST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1623423217.987353000 seconds
[Time delta from previous captured frame: 0.000022000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.504301000 seconds]
Frame Number: 26
Frame Length: 60 bytes (480 bits)
Capture Length: 60 bytes (480 bits)

ping 127.0.0.1



ping 本机地址

可以看到 ping 本机IP 跟 ping 回环地址一样，相关的网络数据，都是走的 **lo0**，本地回环接口，也就是前面提到的**"假网卡"**。

只要走了本地回环接口，那数据都不会发送到网络中，在本机网络协议栈中兜一圈，就发回来了。因此 **ping回环地址和ping本机地址没有区别**。

127.0.0.1 和 localhost 以及 0.0.0.0 有区别吗

回到文章开头动图里的提问，算是面试八股文里的老常客了。

以前第一次用 `nginx` 的时候，发现用这几个 `IP`，都能正常访问到 `nginx` 的欢迎网页。一度认为这几个 `IP` 都是一样的。

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

访问127.0.0.1:80

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

访问localhost:80

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

访问0.0.0.0:80

ifconfig下的ip

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

访问本机的IP地址

但本质上还是有些区别的。

首先 `localhost` 就不叫 `IP`，它是一个域名，就跟 `"baidu.com"`,是一个形式的东西，只不过默认会把它解析为 `127.0.0.1` ，当然这可以在 `/etc/hosts` 文件下进行修改。

所以默认情况下，使用 `localhost` 跟使用 `127.0.0.1` 确实是没区别的。

其次就是 `0.0.0.0`，执行 ping 0.0.0.0 ，是会失败的，因为它在 `IPV4` 中表示的是无效的**目标地址**。

```
$ ping 0.0.0.0
PING 0.0.0.0 (0.0.0.0): 56 data bytes
ping: sendto: No route to host
ping: sendto: No route to host
```

但它还是很有用处的，回想下，我们启动服务器的时候，一般会 `listen` 一个 IP 和端口，等待客户端的连接。

如果此时 `listen` 的是本机的 `0.0.0.0` ，那么它表示本机上的**所有IPV4地址**。

```
/* Address to accept any incoming messages. */
#define    INADDR_ANY      ((unsigned long int) 0x00000000) /* 0.0.0.0   */
```

举个例子。刚刚提到的 `127.0.0.1` 和 `192.168.31.6` ，都是本机的IPV4地址，如果监听 `0.0.0.0` ，那么用上面两个地址，都能访问到这个服务器。

当然， 客户端 `connect` 时，不能使用 `0.0.0.0` 。必须指明要连接哪个服务器IP。

总结

- `127.0.0.1` 是**回环地址**。`localhost` 是**域名**，但默认等于 `127.0.0.1` 。
- `ping` 回环地址和 `ping` 本机地址，是一样的，走的是**lo0 "假网卡"**，都会经过网络层和数据链路层等逻辑，最后在快要出网卡前**狠狠拐了个弯**， 将数据插入到一个**链表**后就**软中断**通知 **ksoftirqd** 来进行**收数据**的逻辑，**压根就不出网络**。所以断网了也能 `ping` 通回环地址。

- 如果服务器 `listen` 的是 `0.0.0.0`，那么此时用 `127.0.0.1` 和本机地址**都可以**访问到服务。

最后

最近工作上的事情太忙，本来就黑的黑眼圈，就更黑了，鸽了大家这么久实在不好意思哈。

欢迎大家加我微信（公众号里右下角“联系我”），互相围观朋友圈砍一刀啥的哈哈。

如果文章对你有帮助，看下文章底部右下角，做点正能量的事情（**点两下**）支持一下。（**疯狂暗示，拜托拜托，这对我真的很重要！**）

我是小白，我们下期见。

参考资料

《127.0.0.1 之本机网络通信过程知多少 ?! 》—— 推荐关注飞哥的《开发内功修炼》

文章推荐：

- 动图图解！既然IP层会分片，为什么TCP层也还要分段？
- 动图图解！GMP模型里为什么要有P？背后的原因让人暖心
- i/o timeout，希望你不要踩到这个net/http包的坑
- 妙啊！程序猿的第一本互联网黑话指南
- 程序员防猝死指南
- 我感觉，我可能要拿图灵奖了。。。
- 给大家丢脸了，用了三年golang，我还是没答对这道内存泄漏题