# Shadowsocks 和服务器

# 1.购买阿里云服务器



客户端 SS 软件登陆公网 IP，不要搞错了搞成私网

点击管理查看端口


选择本地实例安全组

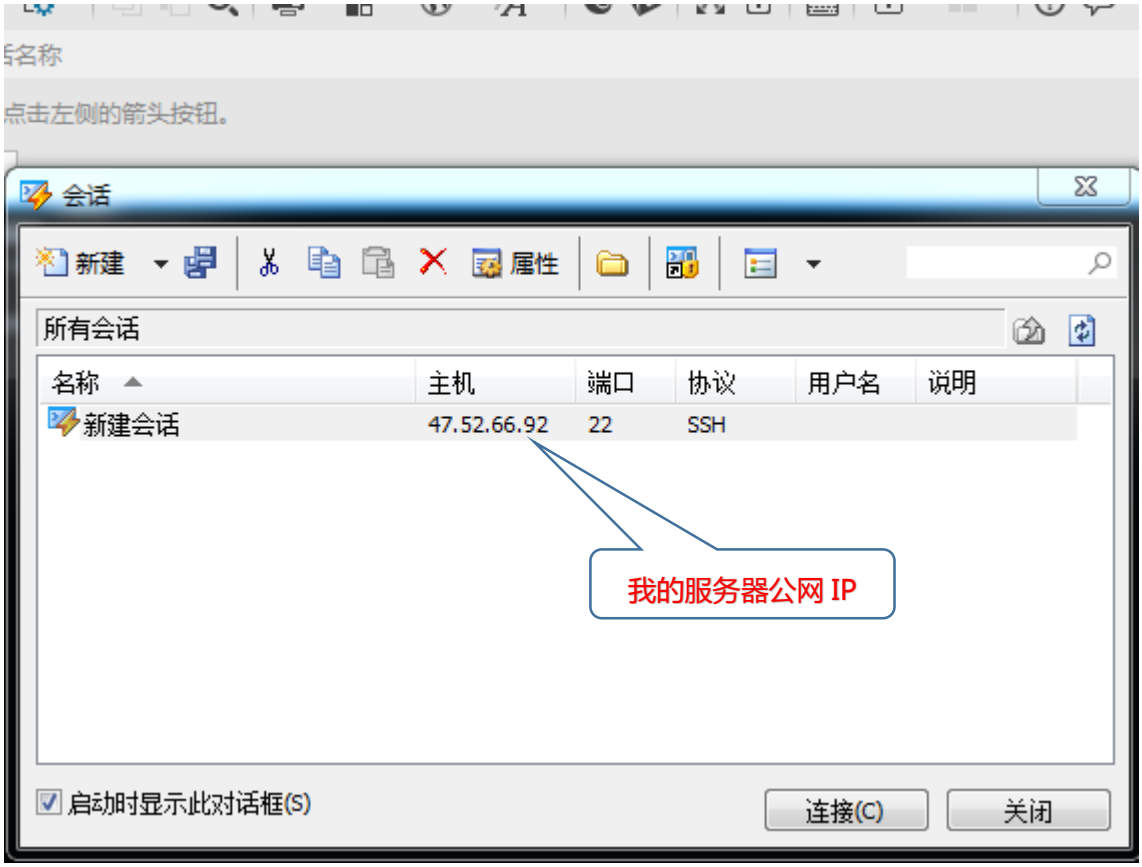
点击配置规则


端口号 88/88，意思不是服务器端口 8888，而是客户端 SS 可以访问的端口是 88 号，如果你 ss 软件有事无事想换起换来的访问端口，你可以设置 88/100，那么你客户端 ss 软件就可以访问 88 到 100 的 12 个端口其中一个，还有端口号要设置越大越好，小端口很容易被 linux 自己系统占用，到时候你端口就重了

阿里云的端口和 IP 地址我们搞懂了。现在用 ssh 登陆上阿里云服务器

我这次用的 Xshell 软件的 ssh

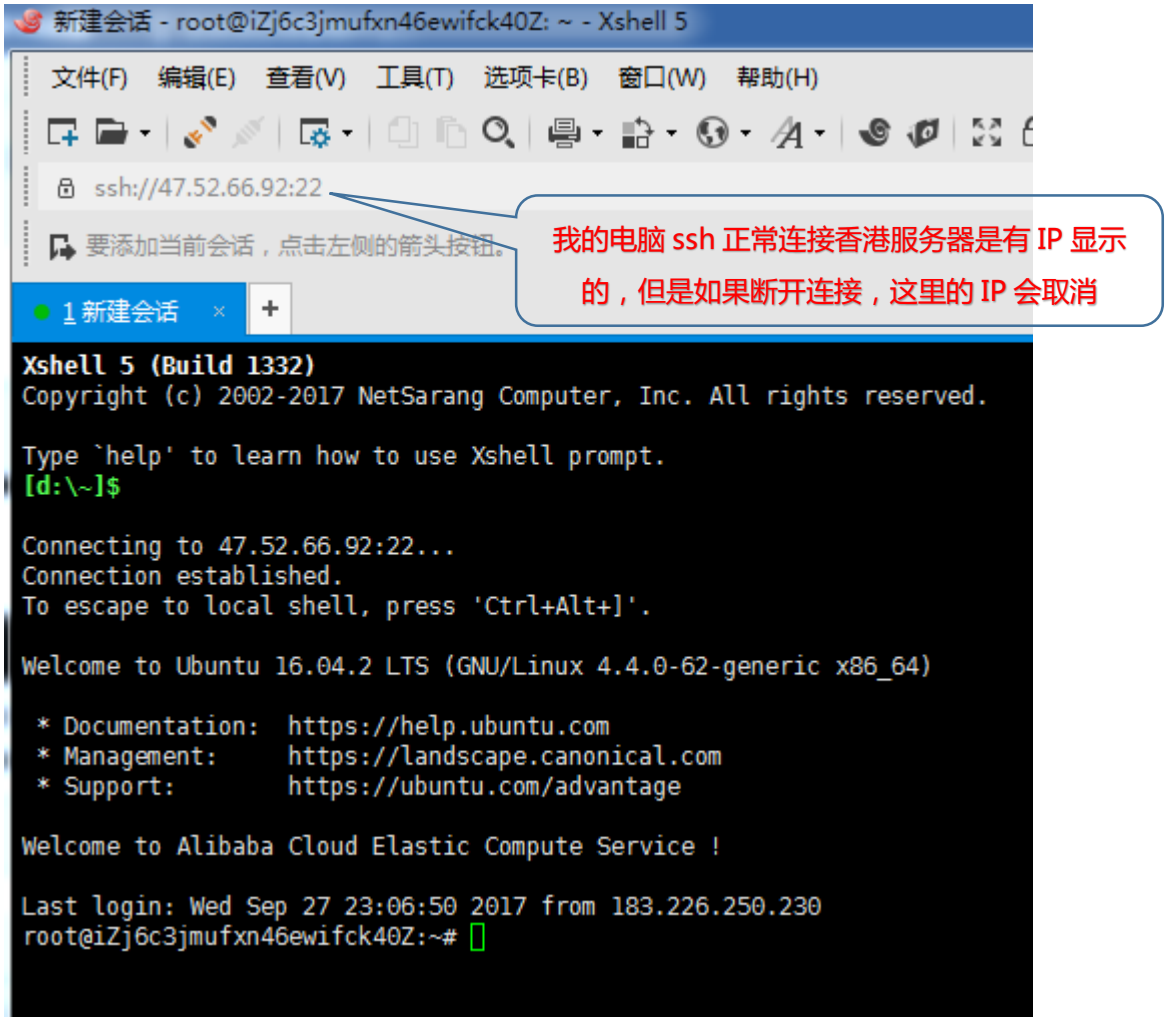
创建连接




点击连接


用户名写 root 不要写我创建阿里云的用户名，因为 linux 默认是认为 root 就等于我阿里的用户名。


密码就是你创建用户时候的密码




这就是成功登陆你在香港的阿里云服务器

```
apt-get install python-pip
pip install shadowsocks
```

有时 Ubuntu 会遇到第一个命令安装 python-pip 时找不到包的情况。pip 官方给出了一个安装脚本，可以自动安装 pip。先下载脚本，然后执行即可：

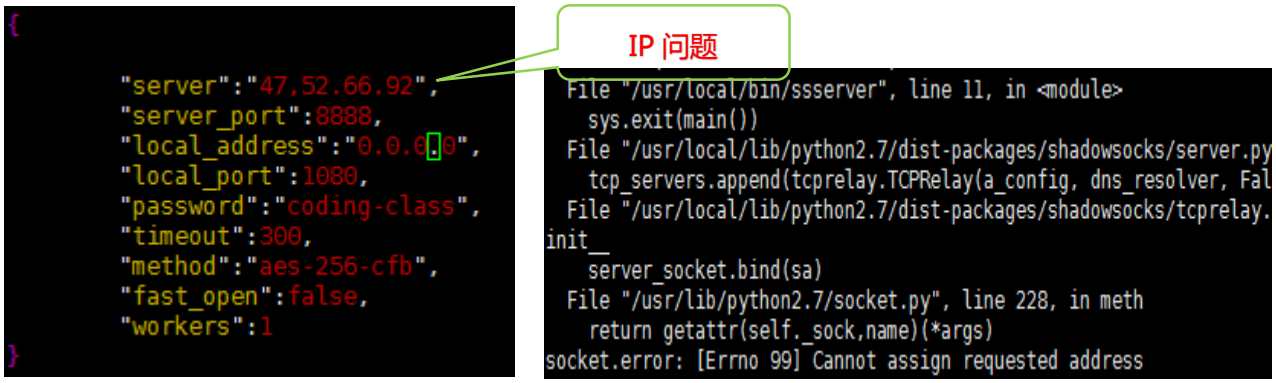
```
wget https://bootstrap.pypa.io/get-pip.py python get-pip.py
```

但是我没有遇到，因为阿里云给你安装了很多安装 pip 之前需要安装的东西，但是其他公司服务器就不一定咯

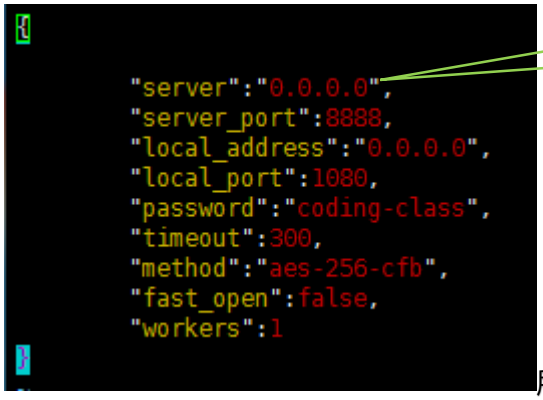
```
vi /etc/shadowsocks.json
```

新建一个 json 文件


这个 socket.gaierror 错误是因为你的 josn 文件 sever ip 地址里面有逗号

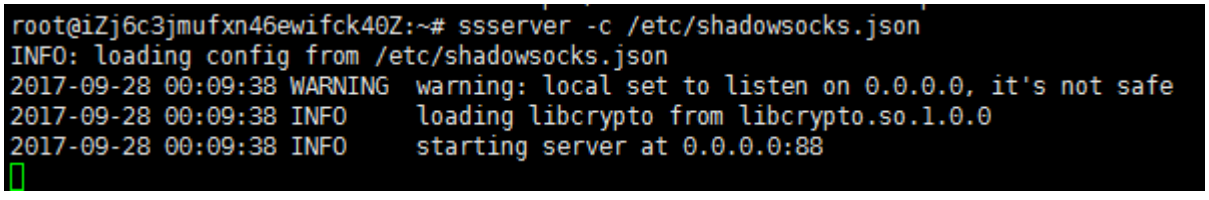



这个 socket.error 问题在有些服务器上面是不会出问题的，在这里出问题是因为阿里云对 IP 做了手脚，所以你在 josn 文件的 server 段不能填服务器的公网 ip，要填 0.0.0.0


所以你填这个就没有问题了

```
ssserver -c /etc/shadowsocks.json 启动阿里云 ss 服务器
```




这就是正常启动，我这里是前台打印

```
ssserver -c /etc/shadowsocks.json -d start  这是后台启动
```

```
ssserver -c /etc/shadowsocks.json -d stop    这是停止后台 ss 服务器
```

```
root@iZj6c3jmufxn46ewifck40Z:~# ssserver -c /etc/shadowsocks.json -d start
INFO: loading config from /etc/shadowsocks.json
2017-09-28 00:13:42 WARNING  warning: local set to listen on 0.0.0.0, it's not safe
2017-09-28 00:13:42 INFO     loading libcrypto from libcrypto.so.1.0.0
started
root@iZj6c3jmufxn46ewifck40Z:~#
```

SS 服务器启动后你可以用 ps aux 来查看

```
root      1331  0.0  0.9  44784  9452 ?        Ss   00:13   0:00 /usr/bin/python /usr/local/bin/ssserver -c /etc/shadowsocks.json
```

如果哪天服务器 ss 程序崩溃了，你就用 ps aux 查看有没有 ss 这个服务在运行

前面 josn 文件有个问题就是端口号是 88 而不是 8888，我没来得及修改，因为阿里服务器是写的 88/88 端口

```
root@iZj6c3jmufxn46ewifck40Z:~# ssserver -c /etc/shadowsocks.json
INFO: loading config from /etc/shadowsocks.json
2017-09-28 00:17:57 WARNING  warning: local set to listen on 0.0.0.0, it's
2017-09-28 00:17:57 INFO     loading libcrypto from libcrypto.so.1.0.0
2017-09-28 00:17:57 INFO     starting server at 0.0.0.0:88
Traceback (most recent call last):
  File "/usr/local/bin/ssserver", line 11, in <module>
    sys.exit(main())
  File "/usr/local/lib/python2.7/dist-packages/shadowsocks/server.py", lin
    tcp_servers.append(tcprelay.TCPRelay(a_config, dns_resolver, False))
  File "/usr/local/lib/python2.7/dist-packages/shadowsocks/tcprelay.py", l
    server_socket.bind(sa)
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 98] Address already in use
root@iZj6c3jmufxn46ewifck40Z:~#
```

这个问题是我的端口已经被后台启动的 ss 服务占用了，如果你再启动 ss 服务，是会报错的[Errno 98]

除非你把 ss 服务器关了，或者你修改 josn 文件里面的端口

虽然 SS 单个端口可以实行多人连接使用但是我想用多个端口实行多人使用，然给每个人的端口配置密码

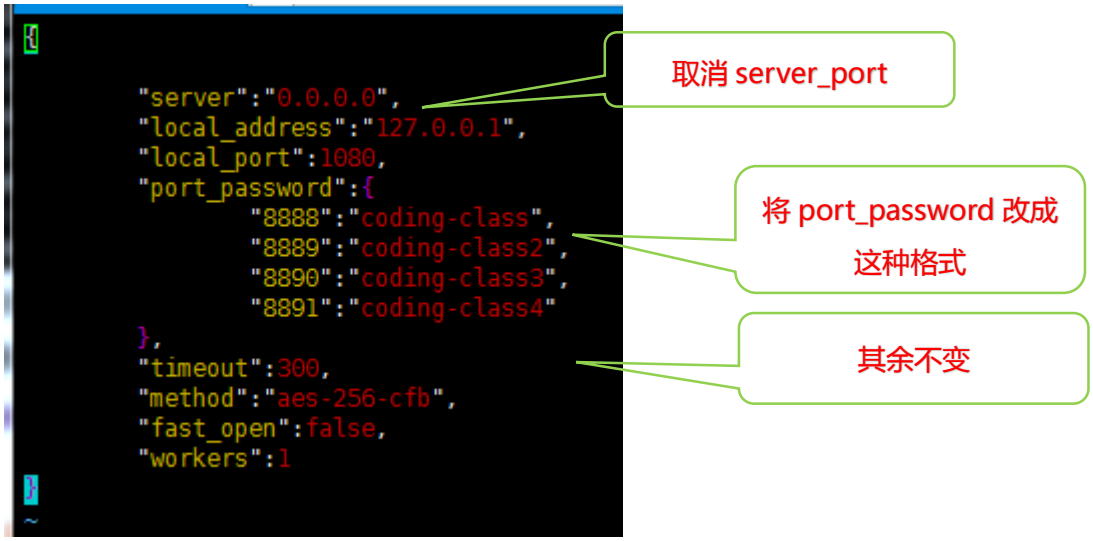
首先关闭 ss 服务器进程

```
root      16472  0.0  1.2  45044 12948 pts/0    S    00:18   0:00 /usr/bin/python /usr/local/bin/ssserver -c shadowsocks.json
```

用 kill -9 16472 将其杀死，或者其它什么命令杀死进程

```
root@iZj6c3jmufxn46ewifck40Z:/etc# vim shadowsocks.json
```

打开 SS 的 json 文件




4 个端口 8888 端口对应的密码是 coding-class。8889 端口对应的密码是 coding-class2，其余端口也是这样设置，还可以增加很多端口，但是你要保证你的阿里云端口范围是增加了的。

入方向 出方向

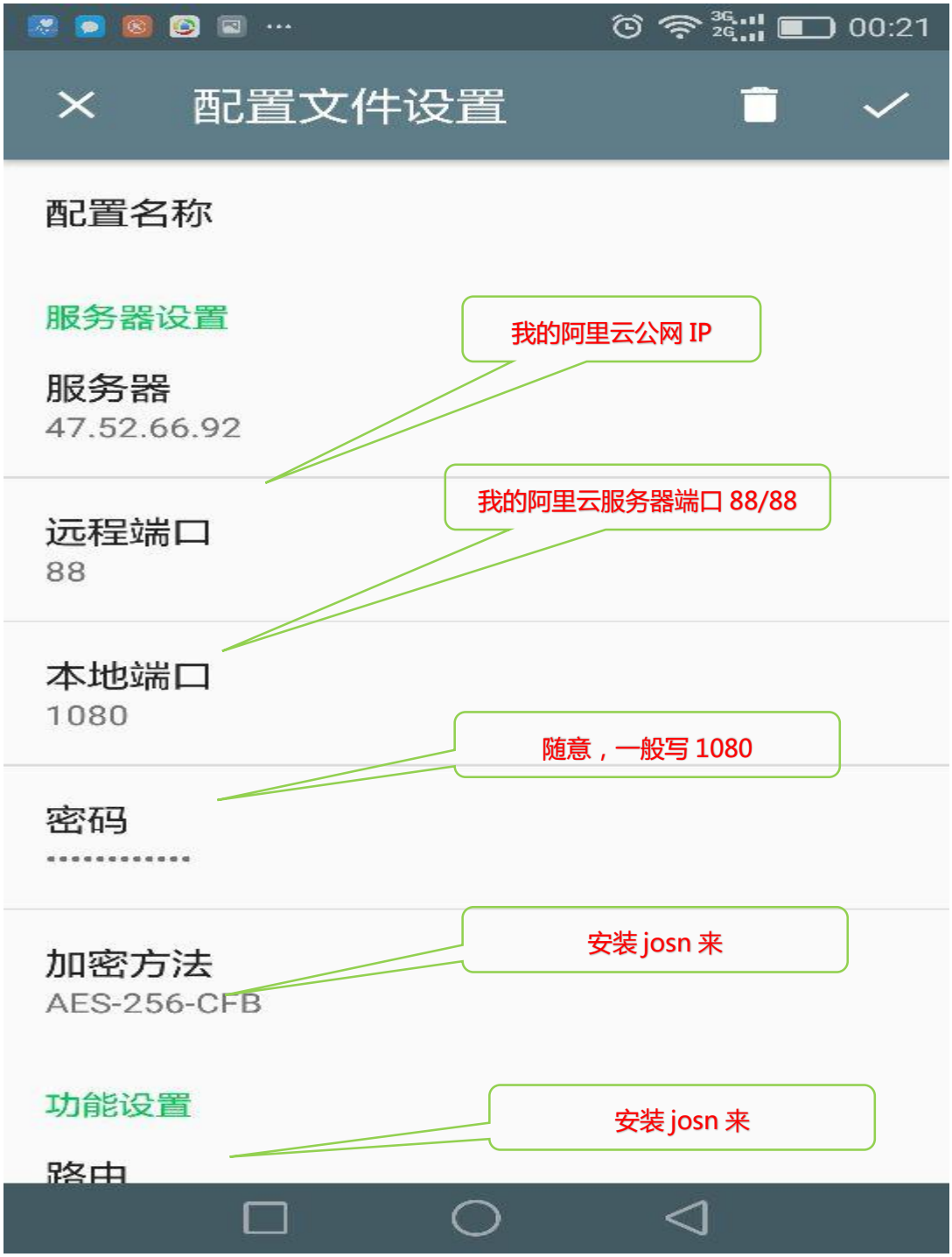
| 授权策略 | 协议类型 | 端口范围 | 授权类型 | 授权对象 | 描述 |
|---|---|---|---|---|---|
| 允许 | 自定义 UDP | 8888/9000 | 地址段访问 | 0.0.0.0/0 | www |
| 允许 | 自定义 TCP | 8888/9000 | 地址段访问 | 0.0.0.0/0 | www |

这是我阿里云端口修改的范围，从

8888~9000 端口都可以使用。

**客户端 ss 配置，我们以手机为例，电脑那个 ss 再另外一个文档里面**

打开传说中的影俊




现在你可以正常使用 ss 与国际接轨了。

# 如何查询有多少人的客户端连接了你的 SS 服务器

netstat -an | grep 'ESTABLISHED' 用这个命令来查询客户端连接数量

```
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED'
tcp        0     52 172.31.6.120:22         183.226.249.44:13718    ESTABLISHED
```

我启动了自己本地电脑的 ss 代理功能

```
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED'
tcp        0     52 172.31.6.120:22         183.226.249.44:13718    ESTABLISHED
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED' | wc -l
1
```

但是连接数还是 1，这是为什么呢？

然后我打开了一个网页

```
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED' | wc -l
3
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED'
tcp        0     52 172.31.6.120:22         183.226.249.44:13718    ESTABLISHED
tcp        0      0 172.31.6.120:41706      216.58.221.132:443      ESTABLISHED
tcp        0    115 172.31.6.120:88         183.226.249.44:15027    ESTABLISHED
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED'
tcp        0     52 172.31.6.120:22         183.226.249.44:13718    ESTABLISHED
tcp        0      0 172.31.6.120:41706      216.58.221.132:443      ESTABLISHED
tcp        0    115 172.31.6.120:88         183.226.249.44:15027    ESTABLISHED
tcp        0      0 172.31.6.120:88         183.226.249.44:15089    ESTABLISHED
tcp        0      0 172.31.6.120:39234      108.177.97.188:443      ESTABLISHED
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED' | wc -l
5
root@iZj6c3jmufxn46ewifck40Z:~# netstat -an | grep 'ESTABLISHED' | wc -l
```

连接数从 3 到 5 再到 3 来回变化，所以这个只能证明有个人连接了我的 SS 服务器但是这个 netstat 命令是用来检测端口的，我的端口是 88 号，并不是检测我的 ss 服务器本身，但是我可以通过端口来查看有没有人连接我的服务器。但是无法获取确定的人数，所以只能知道有人连接你的 ss

# 你买的服务器的网速测试

用 wget 下载工具下载 speedtest-cli 命令软件

wget https://raw.github.com/sivel/speedtest-cli/master/speedtest_cli.py

chmod a+rx speedtest_cli.py

mv speedtest_cli.py /usr/local/bin/speedtest-cli

chown root:root /usr/local/bin/speedtest-cli

直接执行[root@localhost temp]# speedtest-cli

```
root@iZj6c3jmufxn46ewifck40Z:~# speedtest-cli
Retrieving speedtest.net configuration...
Testing from Alicoloud Hk (47.52.66.92)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by STC (Hong Kong) [0.00 km]: 1.959 ms
Testing download speed................................................................................
Download: 343.38 Mbit/s
Testing upload speed................................................................................................
Upload: 1.46 Mbit/s
```

如果执行 speedtest-cli 失败,可能还缺少脚本什么的

```
wget https://github.com/sivel/speedtest-cli/archive/master.zip
```

unzip master.zip

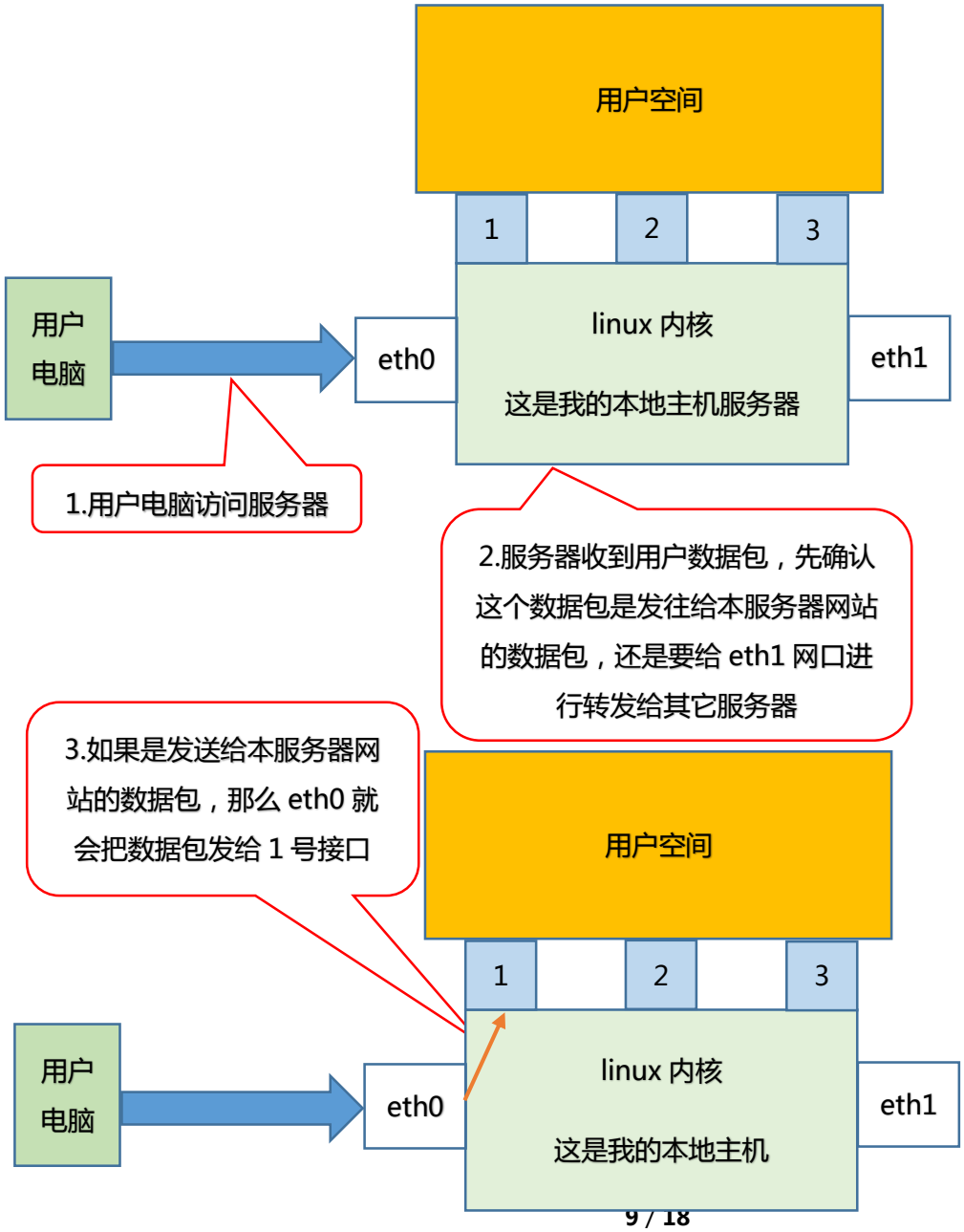
我解压失败，但是再去执行 speedtest-cli 居然成功了

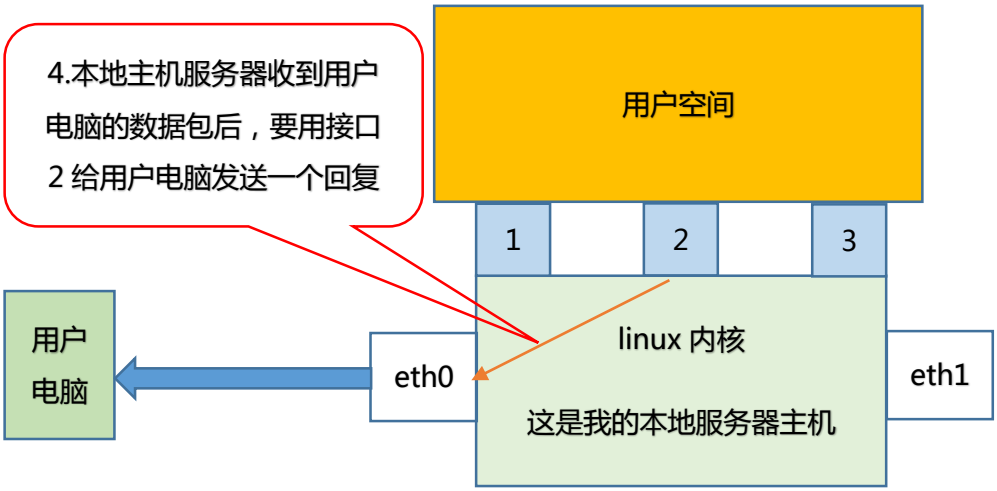
# 使用 iptables 建立防火墙

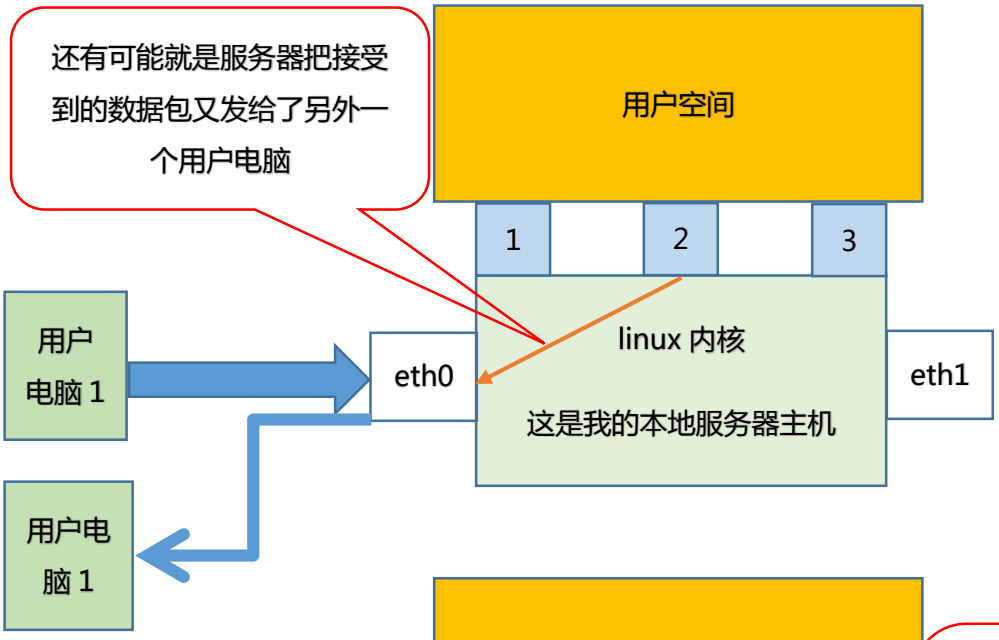
包过滤防火墙：就是对端口进行过滤，比如其他端口数据包不能过我服务器，80 端口允许过我的服务器。

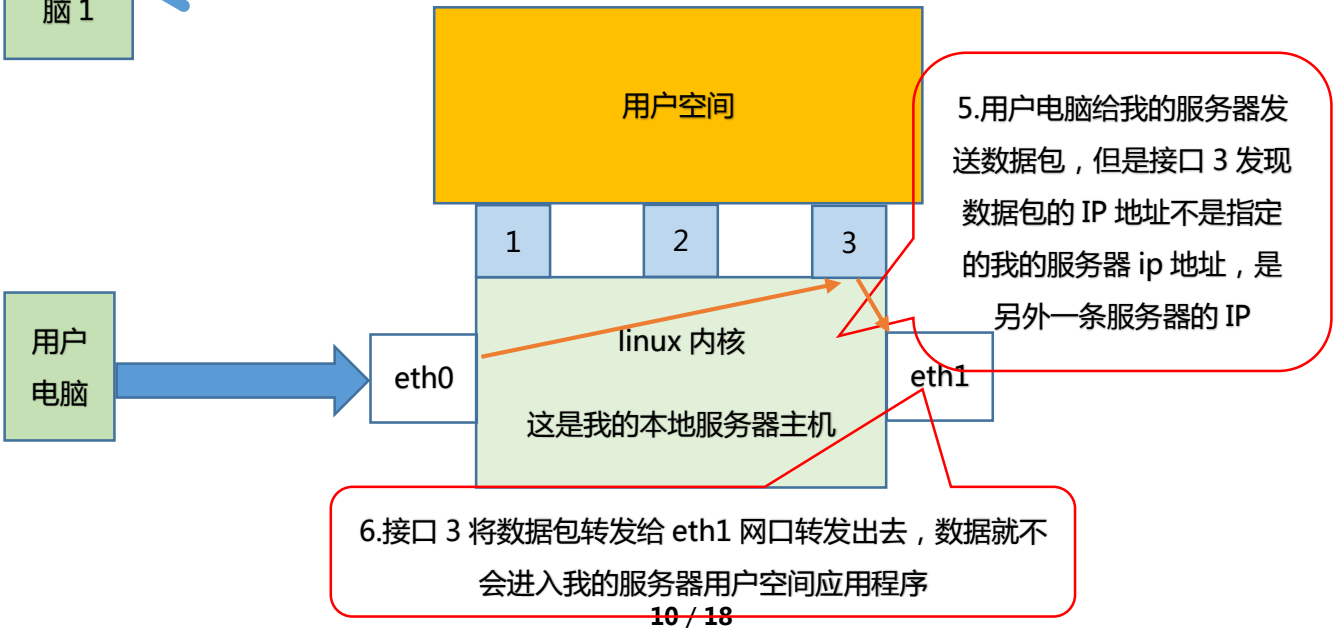
应用层防火墙：针对同一个端口，检查接受到的数据包 url 地址里面的内容，如果是敏感的 url 地址，就把它过滤掉，这个和包过滤防火墙的不同在于，包过滤防火墙一杆子全部打死，应用层防火墙选择性的打死。

## 网络走向图：

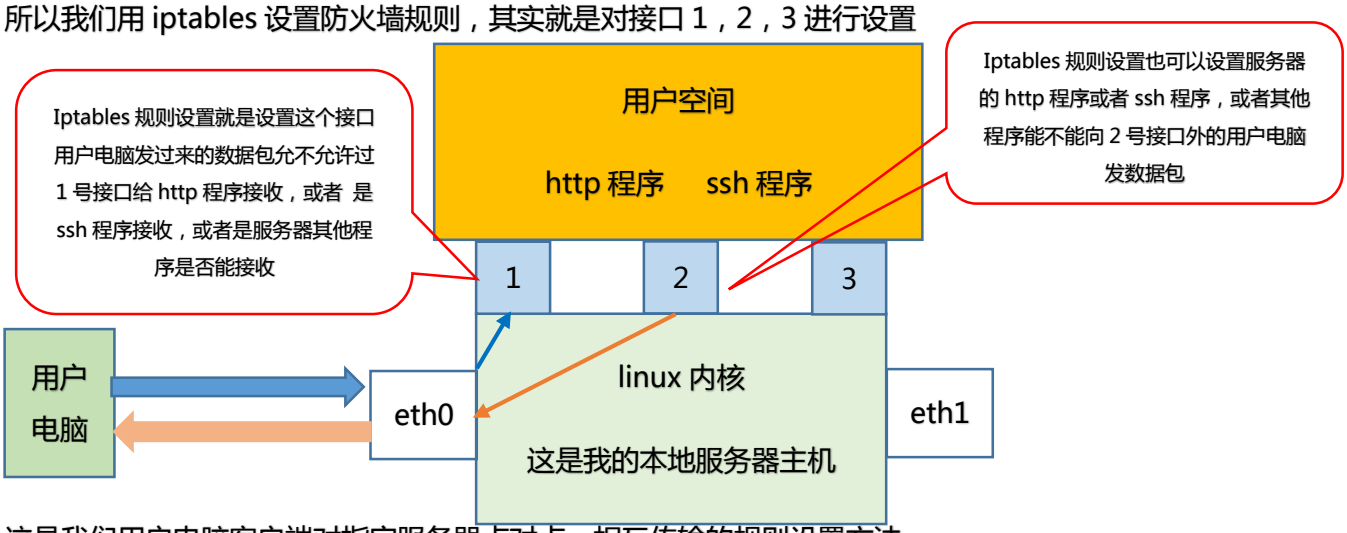



4.本地主机服务器收到用户电脑的数据包后，要用接口2 给用户电脑发送一个回复
用户空间
1
2
3
用户电脑
eth0
linux 内核
这是我的本地服务器主机
eth1


还有可能就是服务器把接受到的数据包又发给了另外一个用户电脑
用户空间
1
2
3
用户电脑 1
eth0
linux 内核
这是我的本地服务器主机
eth1
用户电脑 1

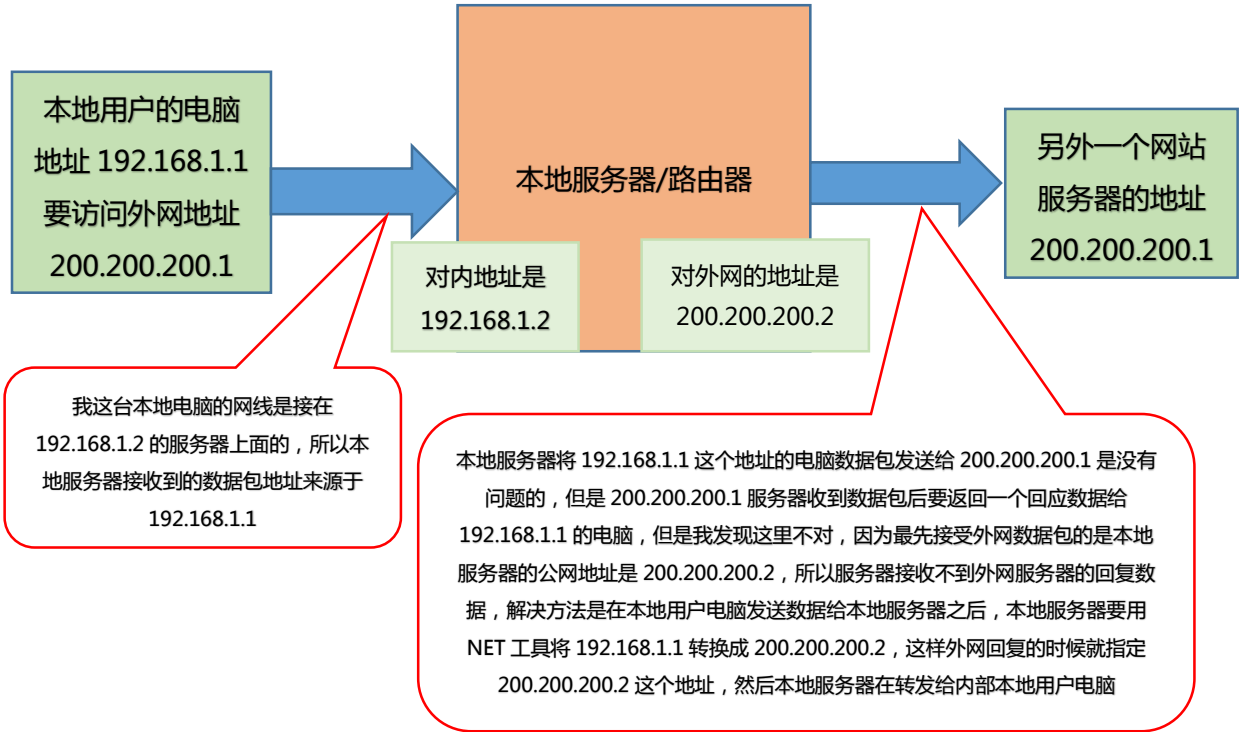

用户空间
5.用户电脑给我的服务器发送数据包，但是接口 3 发现数据包的 IP 地址不是指定的我的服务器 ip 地址，是另外一条服务器的 IP
1
2
3
用户电脑
eth0
linux 内核
这是我的本地服务器主机
eth1
6.接口 3 将数据包转发给 eth1 网口转发出去，数据就不会进入我的服务器用户空间应用程序

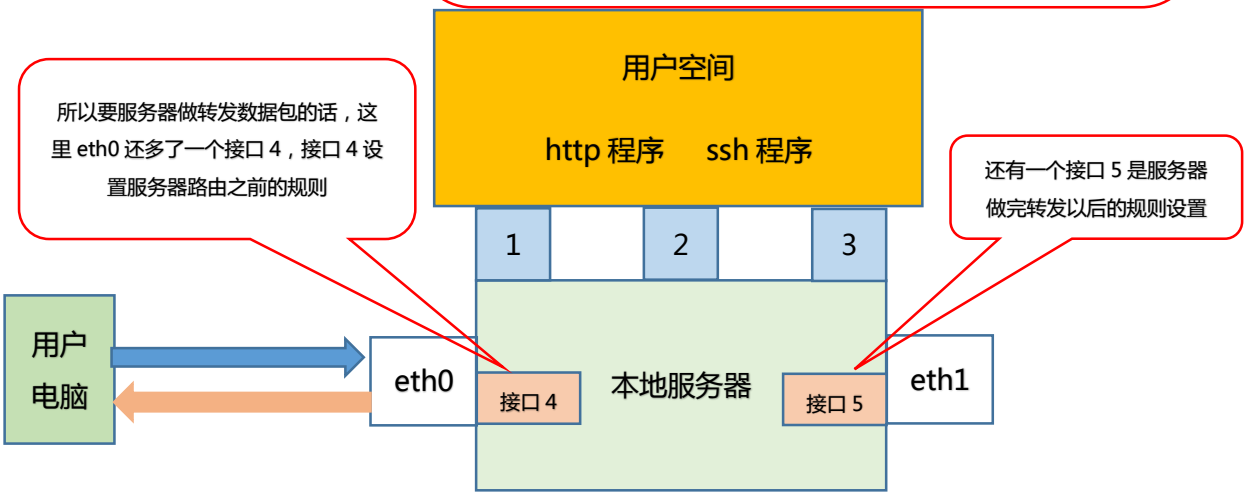
所以我们用 iptables 设置防火墙规则，其实就是对接口 1，2，3 进行设置




这是我们用户电脑客户端对指定服务器点对点，相互传输的规则设置方法。

如果我们电脑连接上自己的服务器，但是我电脑要求访问的地址是另外一台网络的服务器地址，那么这里就要用 NET 来设置路由转发功能，数据流向如下：



所以根据以上网络路径的示意图，我们知道了每个接口都是可以设置规则的，只是规则的意义不一样。

1 号接口：设置服务器入口规则

2 号接口：设置服务器出口规则

3 号接口：设置服务器转发规则

4 号接口：设置路由前规则

5 号接口：设置路由后规则

下面我们来操作一下 iptables 防火墙


这条命令就是查看 filter 表里面 INPUT 链有没有设置什么规则

表有四种：raw raw 表里面有 PREROUTING 链，OUTPUT。

mangle

nat

filter filter 表里面有 INPUT 链，FORWARD 链 ，OUTPUT 链

我们上面就是向 filter 表里面写入 INPUT 链，记住每个表都有固定的几个链可以选择，但是 raw 表不能选择 nat 表里面的链


我们查看了 filter 到底有哪些链可以设置规则。和上面一样只有三个链可以操作


这条规则的意思就是向 filter 表里面的 INPUT 链写入规则，规则是对 tcp 发来的数据进行接收

```
root@ubuntu:/home/xiang# iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

你再查看 filter 表 INPUT 链就多了一条规则

前面查看的时候是没有任何规则。

```
root@ubuntu:/home/xiang# iptables -t filter -D INPUT 1
root@ubuntu:/home/xiang# iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

删除 INPUT 链那一条呢？这里是第 1 条

-D 指定删除 filter 表 INPUT 链的规则

我们下面多加两条规则试试

```
root@ubuntu:/home/xiang# iptables -t filter -A INPUT -p tcp -j ACCEPT
root@ubuntu:/home/xiang# iptables -t filter -A INPUT -p udp -j ACCEPT
root@ubuntu:/home/xiang# iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere
ACCEPT     udp  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

```
root@ubuntu:/home/xiang# iptables -t filter -D INPUT 2
root@ubuntu:/home/xiang# iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:/home/xiang#
```

你看我 INPUT 后面跟的 2 就是删除 INPUT 链第 2 条规则，-D 就是删除规则的意思

```
root@ubuntu:/home/xiang# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:/home/xiang#
```

Ubuntu 系统开机后防火墙默认规则都是 ACCEPT

比如我想修改 filter 表里面的 FORWARD 链数据包转发的默认规则。

```
root@ubuntu:/home/xiang# iptables -P FORWARD DROP
root@ubuntu:/home/xiang# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:/home/xiang#
```

DROP 扔掉

-P 修改默认规则

你发现默认规则修改了

这句话就是将 FORWARD 链里面允许转发数据包的功能，改成丢掉转发的数据包。这样 FORWARD 就不具备转发数据包的功能了。

我们下面来设置一系列防火墙功能

```
root@ubuntu:/home/xiang# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:cf:c4:2e
          inet addr:192.168.14.38  Bcast:192.168.14.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fecf:c42e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13175 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3899 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7342163 (7.3 MB)  TX bytes:362945 (362.9 KB)
```

主机 IP

我主机的 IP 地址是 192.168.14.38

```
root@imx6qdlsolo:/# ssh 192.168.14.38
root@192.168.14.38's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

543 packages can be updated.
371 updates are security updates.

New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Oct 19 03:06:23 2017 from 192.168.14.43
root@ubuntu:~# ls
bin
root@ubuntu:~# cd /
root@ubuntu:/# ls
bin    dev   initrd.img  lib64       media  proc  sbin  tmp   vmlinuz
boot   etc   lib         libx32      mnt    root  srv   usr
cdrom  home  lib32       lost+found  opt    run   sys   var
```

开发板用 ssh 连接我主机

这是开发板上面看到的主机内容

我开发板用 ssh 连接我主机成功

```
root@ubuntu:/home/xiang# iptables -A INPUT -s 192.168.14.44 -j DROP
```

-s 是指定哪个 IP 地址的主机连接我的主机

我给主机加了一条防火墙规则

意思是 IP 地址 为 192.168.14.44 的客户机，不准连接我主机，DROP 就是丢掉 192.168.14.44 机器的数据包

```
root@ubuntu:/home/xiang# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  bogon                anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:/home/xiang#
```

已经增加了一条规则

```
boot   etc   lib    media        opt   run    sys    thress_boy
root@imx6qdlsolo:/# ssh 192.168.14.38
```

我开发板想再用 ssh 连接我主机就连接不上了。

```
root@ubuntu:/home/xiang# iptables -F
```

iptables –F 是清楚掉所有规则

```
root@ubuntu:/home/xiang# iptables -A INPUT -p tcp --dport 22 -j DROP
root@ubuntu:/home/xiang# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere            tcp dpt:ssh
```

-p 是指协议

指定端口

Tcp 协议

任意地址

这句话的意思就是任意地址的 tcp 协议数据包，发送到我主机 22 号端口，都将会被丢弃。

```
^C
root@imx6qdlsolo:/# ssh 192.168.14.38
```

你看我开发板还是连接不上主机，因为 ssh 端口就是 22

我下面修改一下

```
root@ubuntu:/home/xiang# iptables -A INPUT -p tcp --dport 23 -j DROP
root@ubuntu:/home/xiang# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere             tcp dpt:telnet

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

我把端口改成 23 号

```
root@imx6qdlsolo:/# ssh 192.168.14.38
root@192.168.14.38's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

543 packages can be updated.
371 updates are security updates.

Last login: Thu Oct 19 03:21:28 2017 from 192.168.14.44
root@ubuntu:~# cd /home/xiang/
root@ubuntu:/home/xiang# ls
AP6212                examples.desktop  makefilepractice   vim_chajian
bgku                  githubfile        Music              web_stdy_guide
Cdata1-fsl-linux      guide             Pictures           wh.sh
Cdata_IMX6system.tar  IMX6              Public             yello_http
Desktop               IMX6ul            STM32              yocto
Documents             libbbb            Templates
Downloads             linux_code_guide  video_driver_test
driver_test           linux_command     Videos
root@ubuntu:/home/xiang#
```

你看我开发板就连接上了。

## Iptables 多条匹配规则使用

```
root@ubuntu:/home/xiang# ping 192.168.14.46
PING 192.168.14.46 (192.168.14.46) 56(84) bytes of data.
64 bytes from 192.168.14.46: icmp_seq=1 ttl=64 time=2.34 ms
64 bytes from 192.168.14.46: icmp_seq=2 ttl=64 time=0.565 ms
64 bytes from 192.168.14.46: icmp_seq=3 ttl=64 time=1.35 ms
64 bytes from 192.168.14.46: icmp_seq=4 ttl=64 time=0.948 ms
^C
--- 192.168.14.46 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.565/1.302/2.340/0.662 ms
```

```
iptables-restore  iptables-test
root@imx6qdlsolo:~# ping 192.168.14.38
PING 192.168.14.38 (192.168.14.38): 56 data bytes
64 bytes from 192.168.14.38: seq=0 ttl=64 time=1.435 ms
64 bytes from 192.168.14.38: seq=1 ttl=64 time=0.724 ms
64 bytes from 192.168.14.38: seq=2 ttl=64 time=0.470 ms
64 bytes from 192.168.14.38: seq=3 ttl=64 time=1.128 ms
64 bytes from 192.168.14.38: seq=4 ttl=64 time=0.914 ms
^C
--- 192.168.14.38 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.470/0.934/1.435 ms
root@imx6qdlsolo:~#
```

现在我两边的主机都能相互 ping 通，我现在不用端口规则和 IP 地址规则，我们用协议规则来做防火墙

```
root@ubuntu:/home/xiang# iptables -A INPUT -p icmp -j DROP
root@ubuntu:/home/xiang# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere
```

这句表示所有 icmp 协议的数据包来到我主机，我主机都把他们丢弃

```
root@ubuntu:/home/xiang# ping 192.168.14.46
PING 192.168.14.46 (192.168.14.46) 56(84) bytes of data.
^C
--- 192.168.14.46 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3001m
```

```
root@imx6qdlsolo:~# ping 192.168.14.38
PING 192.168.14.38 (192.168.14.38): 56 data bytes
^C
--- 192.168.14.38 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
root@imx6qdlsolo:~#
```

开发板 ping 主机平不通是因为主机设置了 icmp 协议丢弃，但是主机怎么会 ping 不通开发板呢？开发板上面什么防火墙都没有，其实主机是 ping 通了开发板的，只是开发板返回的包被主机 icm 协议防火墙拦截了

```
root@ubuntu:/home/xiang# iptables -I INPUT -p icmp --icmp-type 8 -j DROP
root@ubuntu:/home/xiang# iptables -I INPUT -p icmp --icmp-type 3 -j ACCEPT
root@ubuntu:/home/xiang# iptables -I INPUT -p icmp --icmp-type 0 -j ACCEPT
```

我在原来的 iptables 规则上又加了新的规则

```
root@ubuntu:/home/xiang# ping 192.168.14.46
PING 192.168.14.46 (192.168.14.46) 56(84) bytes of data.
64 bytes from 192.168.14.46: icmp_seq=1 ttl=64 time=1.04 ms
64 bytes from 192.168.14.46: icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from 192.168.14.46: icmp_seq=3 ttl=64 time=0.958 ms
```

```
root@imx6qdlsolo:~# ping 192.168.14.38
PING 192.168.14.38 (192.168.14.38): 56 data bytes
^C
--- 192.168.14.38 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
root@imx6qdlsolo:~#
```

你看主机就可以 ping 通开发板了，开发板 ping 不通主机是正常的，因为我主机有防火墙。

```
root@ubuntu:/home/xiang# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             anywhere             icmp echo-reply
ACCEPT     icmp --  anywhere             anywhere             icmp destination-unreachable
DROP       icmp --  anywhere             anywhere             icmp echo-request
DROP       icmp --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
```

这句的意思是其它主机发给我的请求全部丢掉

这是最先加的一条规则，意思是拒绝所有 icmp 数据

我 ping 其他主机，不可接受其他主机的应答，我这里要求放行，就是可以接受其他主机的数据包回复

这就是允许我去 ping 别人，但是不允许别人 ping 我的规则

所以这个规则就是先一杆子打死

然后放行一些东西。

## 服务器查看网卡实时流量

apt-get install ifstat

报错：Unable to locate package ifstat

你要更新下 apt-get 源

执行 sudo apt-get update

再安装 apt-get install ifstat

问题解决

```
root@iZj6c3jmufxn46ewifck40Z:~# ifstat
       eth0
 KB/s in  KB/s out
    0.06      0.12
    0.06      0.15
    0.49      0.55
    0.06      0.05
    0.06      0.13
    0.06      0.13
    0.06      0.13
    0.06      0.27
```

这是 SS 没连接查看视频的流量

```
  0.12      0.13
  0.18      0.32
  0.06      0.06
  0.00      0.27
  8.26      8.47
 46.47     35.35
  9.95      9.98
 13.08     16.40
  2.44      2.53
  6.05      5.32
 18.54     18.62
  0.12      0.34
  0.61      0.66
  0.06      0.34
```

这是 SS 连接上我的服务器了产生的连接流量

```
KB/s In  KB/s Out
  2.37     51.81
125.12     79.97
121.20    186.89
111.64     60.85
109.25     83.78
 95.99    110.84
  2.22     46.29
106.03    110.56
  1.88     11.33
```

这是看 twiter 产生的流量，看直播视频流量就起来了。

## 现在我用 iptables 给端口限速试试

限速前先将端口全部设置成丢包

```
root@iZj6c3jmufxn46ewifck40Z:~# iptables -F
root@iZj6c3jmufxn46ewifck40Z:~# iptables -I INPUT -p tcp --dport 8889  -j DROP
root@iZj6c3jmufxn46ewifck40Z:~# iptables -I INPUT -p udp --dport 8889  -j DROP
root@iZj6c3jmufxn46ewifck40Z:~# iptables -I INPUT -p tcp --dport 8889 -m limit --limit 10/s -j ACCEPT
root@iZj6c3jmufxn46ewifck40Z:~# iptables -I INPUT -p udp --dport 8889 -m limit --limit 10/s -j ACCEPT
root@iZj6c3jmufxn46ewifck40Z:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     udp  --  anywhere             anywhere             udp dpt:8889 limit: avg 10/sec burst 5
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:8889 limit: avg 10/sec burst 5
DROP       udp  --  anywhere             anywhere             udp dpt:8889
DROP       tcp  --  anywhere             anywhere             tcp dpt:8889

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

然后再选择性限速

iptables -I INPUT 是对下载做设置

iptables -I OUTPUT 是对上传做设置

在 ubuntu 论坛查到的资料说限速的时候规则是成对使用的，第一条是-j ACCEPT，接受符合规则的数据包，第二条是-j DROP 抛弃不符合的数据包，有错误欢迎指正。

限速的时候--limit 60/s 根据查到的资料说是这个"60"不是直接的速度而是数据包的数量，60/s 是指每秒转发 60 个数据包，每个数据包是 1.5Kbyte（貌似有人说是 1480byte？），所以限制 60/s 的时候，被限制的 IP 平均速度是 90K/s 的样子