**回归测试：协议限速和QoS测试文档**

# 协议限速

## 测试方法

此过程中需要用到本地机，服务器240【使用自己的账号登录】以及交换机223。使用wireshark查看报文。

**前期配置环节：**

服务器每次只要重启，这些操作都是要做的

首先登录240服务器：ssh zhangben@10.250.0.240,执行如下操作：

**sudo docker exec -it testbed bash**

**cd sonic-mgmt/ansible**

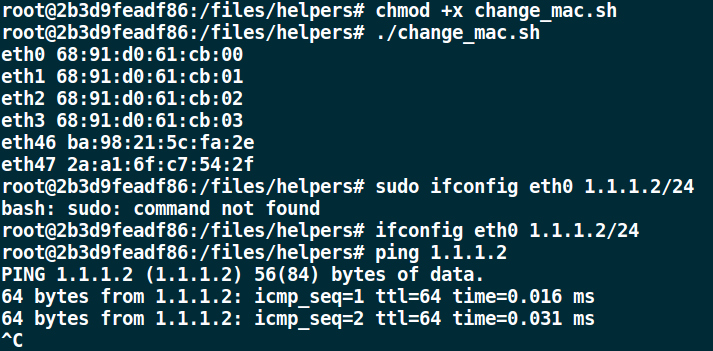
**./testbed-cli.sh start-vms Ttf-t02 password.txt**

**./testbed-cli.sh add-topo Ttf-t02 password.txt**

**exit**

**【退出testbed这个Docker，回到服务器240】**

如果过程中哪条语句有报错，就再执行一次那条语句。进入Docker ptf\_tf-t02中（**sudo docker exec -it ptf\_tf-t02 bash**）在**./files/helpers**中找到change\_mac.sh文件，运行这个文件（首先执行在这个路径下执行**chmod +x change\_mac.sh** 这似乎是一个赋权操作然后做**./change\_mac.sh**操作）

对端口配置，其实在./change\_mac.sh执行后会显示各个端口的MAC地址

目前似乎只能eth0能够使用，所以用**ifconfig eth0 ip地址/掩码位数** 给eth0配置了和交换机Ethernet0一个网段的IP地址。可以使用**ifconfig**命令查看端口配置信息。

**下面是抓包测试环节：**

在ptf\_tf-t02 Docker下ping交换机端口IP地址，目前是交换机Ethernet0的端口，另一边在交换机223中使用**sudo tcpdump -i 交换机端口名 -w 名字.pcap**【目前因为使用的是Ethernet0，所以端口号写的是这个，-w是说明写到这个.pcap文件中，这个名字可以自己起】wireshark文件的后缀名就是.pcap，然后将这个文件**scp**到本地机，在本地机中打开（这是因为wireshark是可视化页面，只装在了本地，所以必须先拷贝到本地查看），然后选取所要分析的报文类型单独导出一个.pcap文件【可以以**协议名\_pkt.pcap**命名】。

然后在将这个.pcap文件scp到服务器240上，但是这只是一个中转，还要进入240将这个文件拷贝到Docker ptf\_tf-t02中，这个操作使用**sudo docker cp lldp\_pkt.pcap ptf\_tf-t02:/（**直接复制到了Docker的根目录下,也可以自己创建一个新的文件夹，把路径写清楚就行),然后使用**sudo docker exec -it ptf\_tf-t02 bash**进入Docker下，开始模拟发包。

**模拟发包操作：**

root@2b3d9feadf86:/# **scapy**

>>> **f=rdpcap('/协议名\_pkt.pcap')**

>>> **f.display()**

>>> **pkt1 = f[0].copy()**

>>> **pkt1.display()**

>>> **sendp(pkt1,iface="eth0",loop=1)**

loop=1表示一直循环发包，然后再使用**sudo tcpdump -i Ethernet0 -w 名字.pcap**将数据抓取出来，再scp到本地机，使用wireshark查看，**计算每秒发了多少个包。和配置文件中所做规定的限速做比较。**

这个配置文件的位置是swss（在223中已经有，所以在223中操作）这个Docker中,所以对它进行修改时应该先**docker exec -it swss bash**进入swss，然后**vim /etc/swss/config.d/00-copp.config.json** 进行修改。

# 测试协议速度

承诺信息速率CIR 、峰值信息速率PIR、承诺突发尺寸CBS 和峰值突发尺寸PBS

其中对协议进行测试只需要改变CIR和PIR的值就可。

## ARP

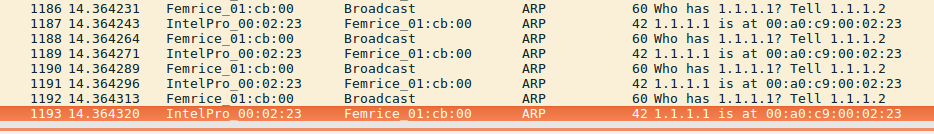
**使用上述测试方法测试**

地址解析协议，如果ping一个ARP表中没有目的地址的，会广播泛洪询问，然后目标地址会单播回自己的mac地址。

对arp的限速如下：将cir和pir限制在100次/s

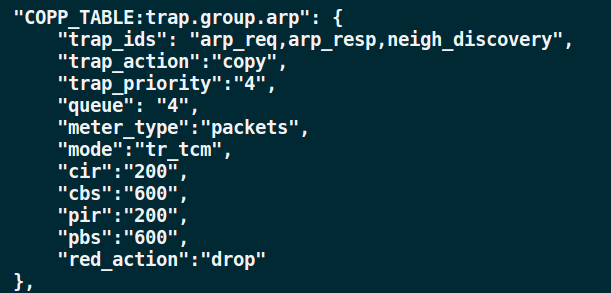


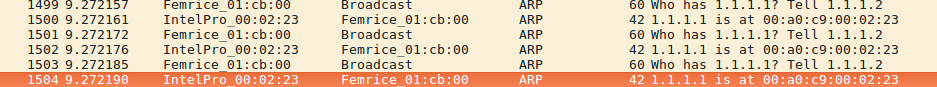
一波操作后



arp\_rate=1193/2/14.364320~=41.50次/s

将cir和pir限制在200次/s



一波操作

arp\_rate=1504/2/9.272190～=81.10次/s

通过限速100和200看出，确实存在限速的效果，但是并不接近在所限制的参数，导师滕飞说这可能是由此软件发包本身的不准确已经ARP报文只要64B造成的。

所以为了验证这个想法，决定测ICMP作为参考。

## ip2me(ICMP、UDP)

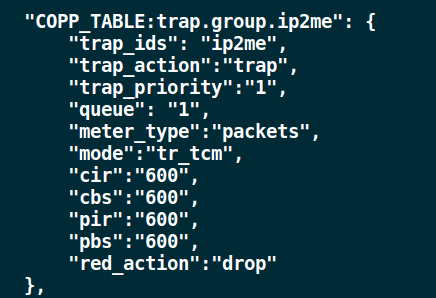
### ICMP使用scapy模拟发包测速

ICMP：Internet Control Message Protocol国际控制报文协议。

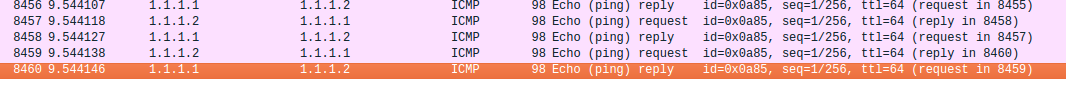
看了一波ICMP维基介绍，虽然没有完全理解，但是也知道了ICMP考研时的理解差了很多，简单来讲，ICMP就很像一个纠错的机器，会发送一些错误信息，告知主机错误信息。也可以存储一些重定向的信息等，改变或者调整路由路径。

Ping和tranceroute是ICMP的应用，所以ping发送的是ICMP报文。

先测试不同的包长的影响

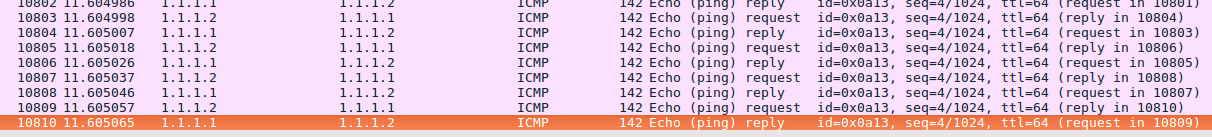
**限速设置是修改ip2me中的参数（cir和pir都设置为600）** 

**直接ping**



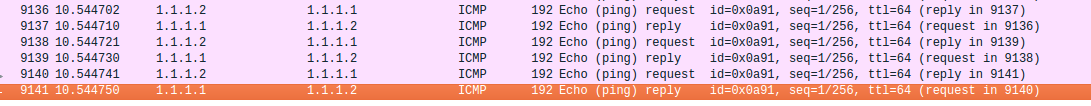
Rate=8460/2/9.544146～=443.20(次/s)

**使用了ping -s 100 1.1.1.1命令，即把包长设置成了100**



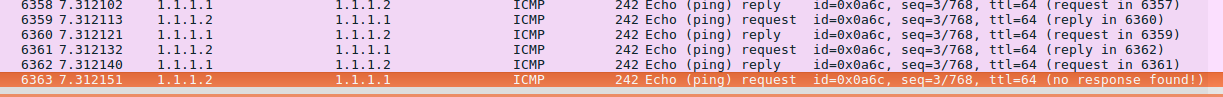
Rate=10810/2/11.605065=465.74(次/s)

**使用了ping -s 150 1.1.1.1命令，即把包长设置成了150**



Rate=9141/2/10.544750=433.44(次/s)

**使用了ping -s 200 1.1.1.1命令，即把包长设置成了200**



Rate=6363/2/7.312151=435.10（次/s)

**使用了ping -s 400 1.1.1.1命令，即把包长设置成了200**



Rate=10912/2/12.584801~=433.54(次/s)

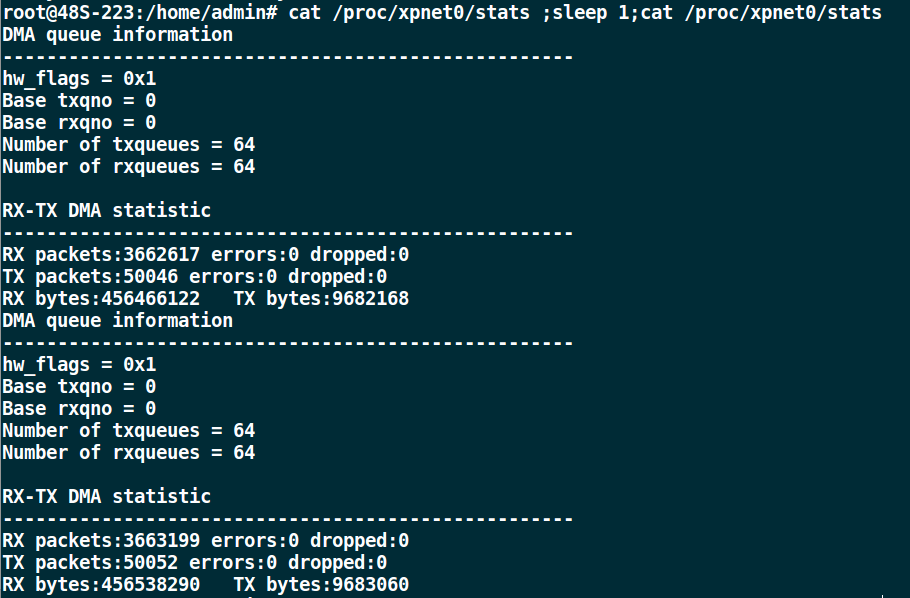
### **UDP使用仪表测试**

**用仪表发包，不过还是从223Ethernet0抓包看。因为仪表没有提供ICMP类型的包，所以使用UDP进行测试。**

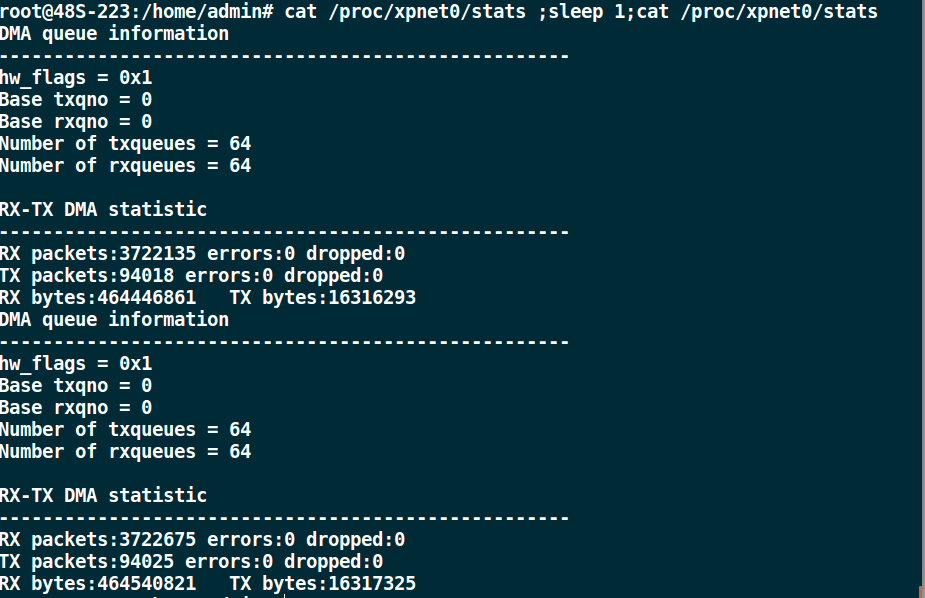
**限速600**

**使用cat /proc/xpnet0/stats; sleep 1;cat /proc/xpnet0/stats可以查看一秒内发了多少包。**

**Tx.size=100**

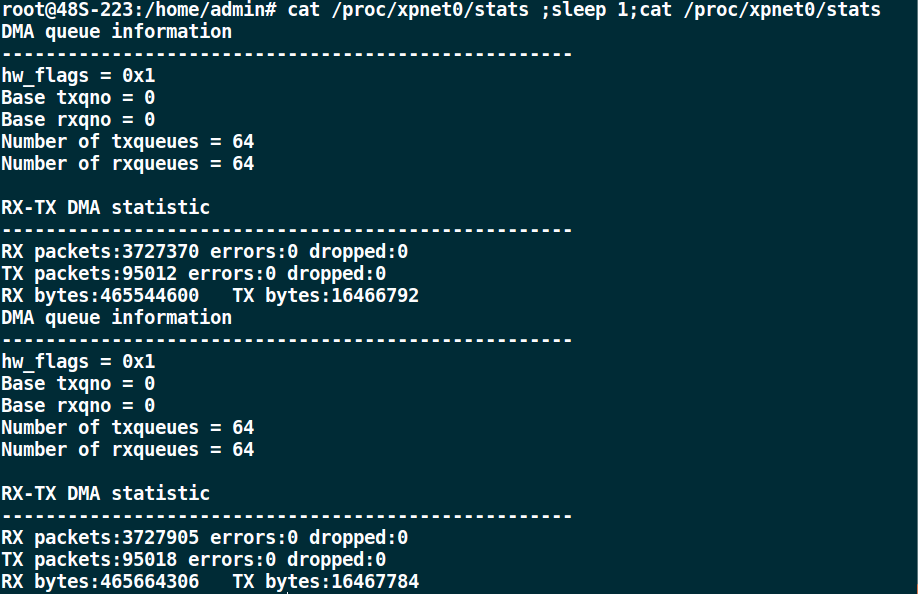


Rate=582包/s【此时限速仍旧是600】

**Tx.size=150** 

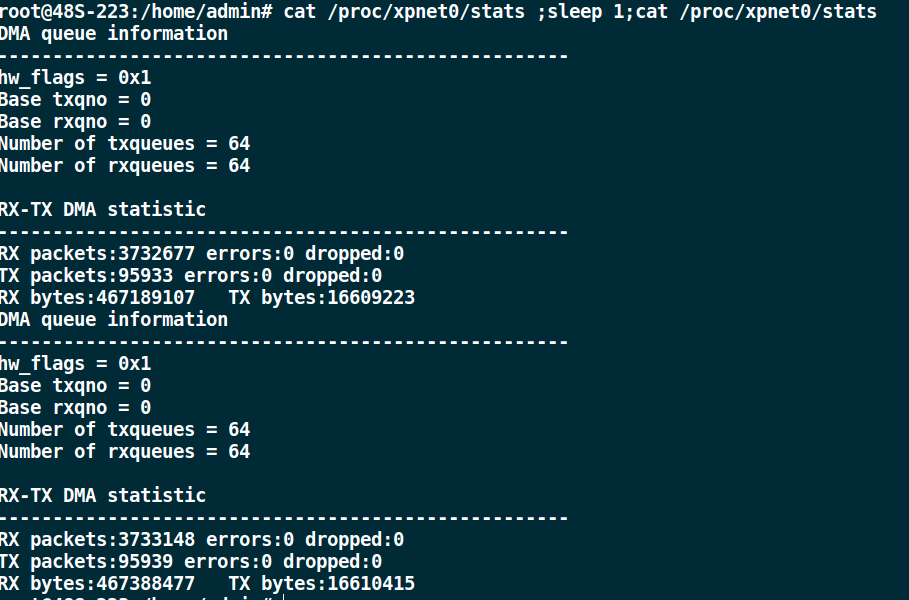
Rate=490包/s

**Tx.size=200**



Rate=535包/s

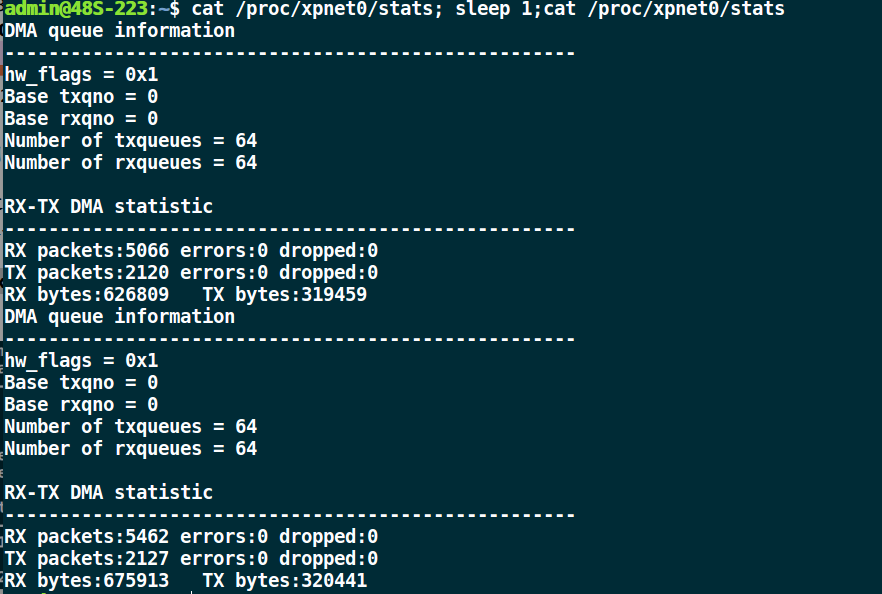
Tx.size=400

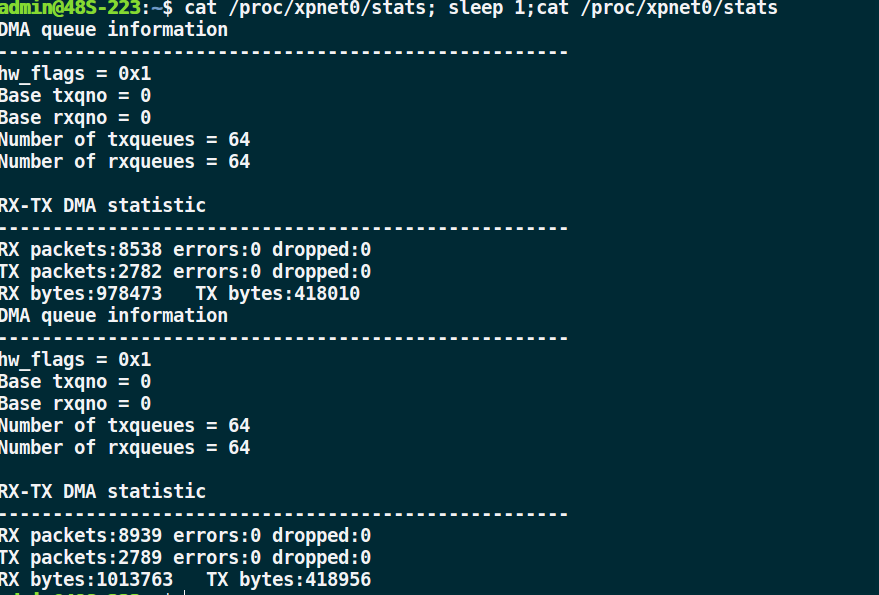


Rate=471

**Note:以上接近限速值是在将仪表的发包速度限制在了800包/s的速率，如果不限制tx.rate会出现很大的误差，这时候发包速率很快，结果抓包到的速率只有31包/s，误差很大。在这种情况下，应该限制发包速率，再进行测试。**

**限速400，包长100**

**Rate=396(包/s)**



Rate=401(包/s)