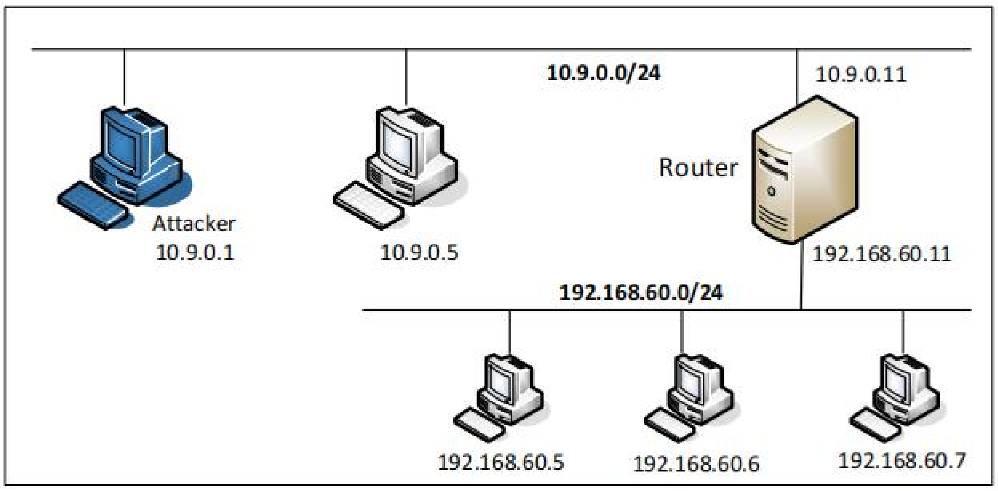
Firewall Exploration Lab

57118214 陈佳杰

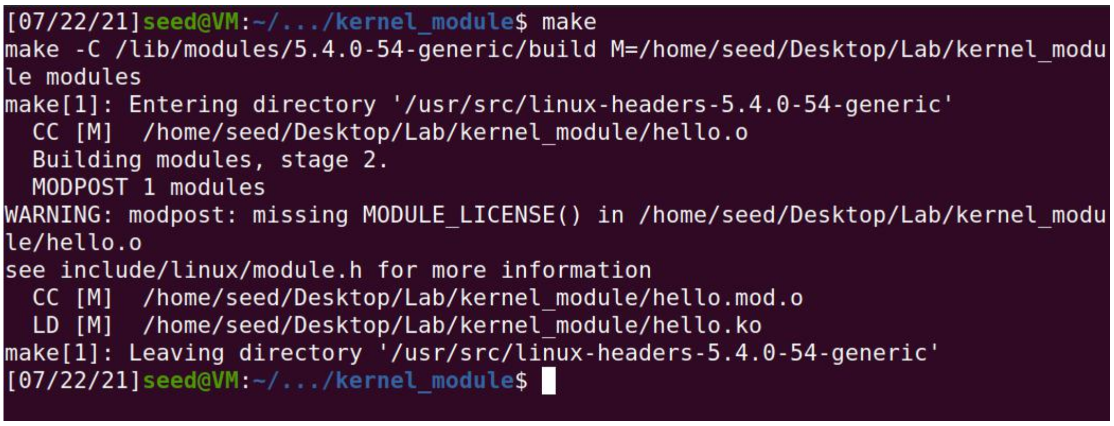
Task 1: Implementing a Simple Firewall

实验环境如下图所示



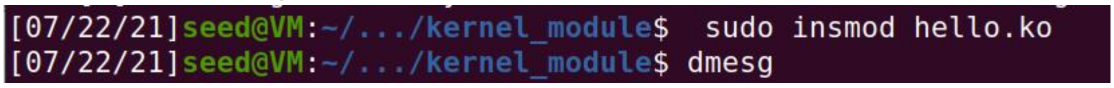
Task 1.A: Implement a Simple Kernel Module

编译内核

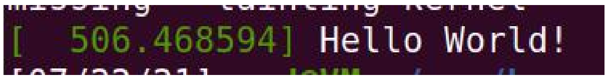


```
[07/22/21]seed@VM:~/.../kernel_module$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/Lab/kernel_modu
le modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/Lab/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/seed/Desktop/Lab/kernel_modu
le/hello.o
see include/linux/module.h for more information
  CC [M]  /home/seed/Desktop/Lab/kernel_module/hello.mod.o
  LD [M]  /home/seed/Desktop/Lab/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/22/21]seed@VM:~/.../kernel_module$
```

加载模块并用 dmesg 来查看信息

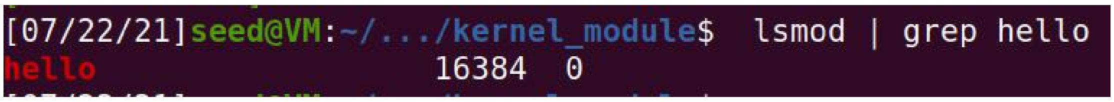


```
[07/22/21]seed@VM:~/.../kernel_module$  sudo insmod hello.ko
[07/22/21]seed@VM:~/.../kernel_module$ dmesg
```



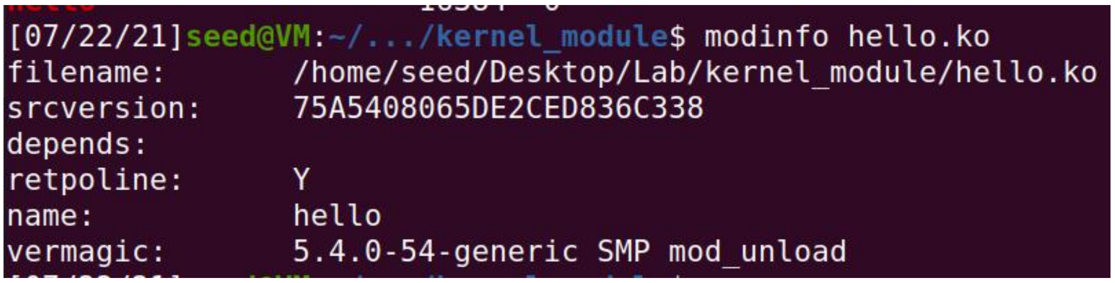
```
[  506.468594] Hello World!
```

列出模块



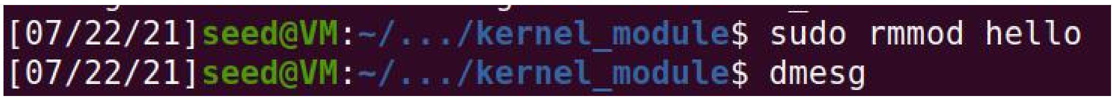
```
[07/22/21]seed@VM:~/.../kernel_module$  lsmod | grep hello
hello                  16384  0
```

查看模块相关信息

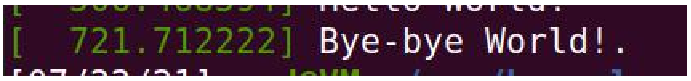


```
[07/22/21]seed@VM:~/.../kernel_module$ modinfo hello.ko
filename:       /home/seed/Desktop/Lab/kernel_module/hello.ko
srcversion:     75A5408065DE2CED836C338
depends:
retpoline:      Y
name:           hello
vermagic:       5.4.0-54-generic SMP mod_unload
```

删除模块并用 dmesg 查看信息



```
[07/22/21]seed@VM:~/.../kernel_module$ sudo rmmod hello
[07/22/21]seed@VM:~/.../kernel_module$ dmesg
```

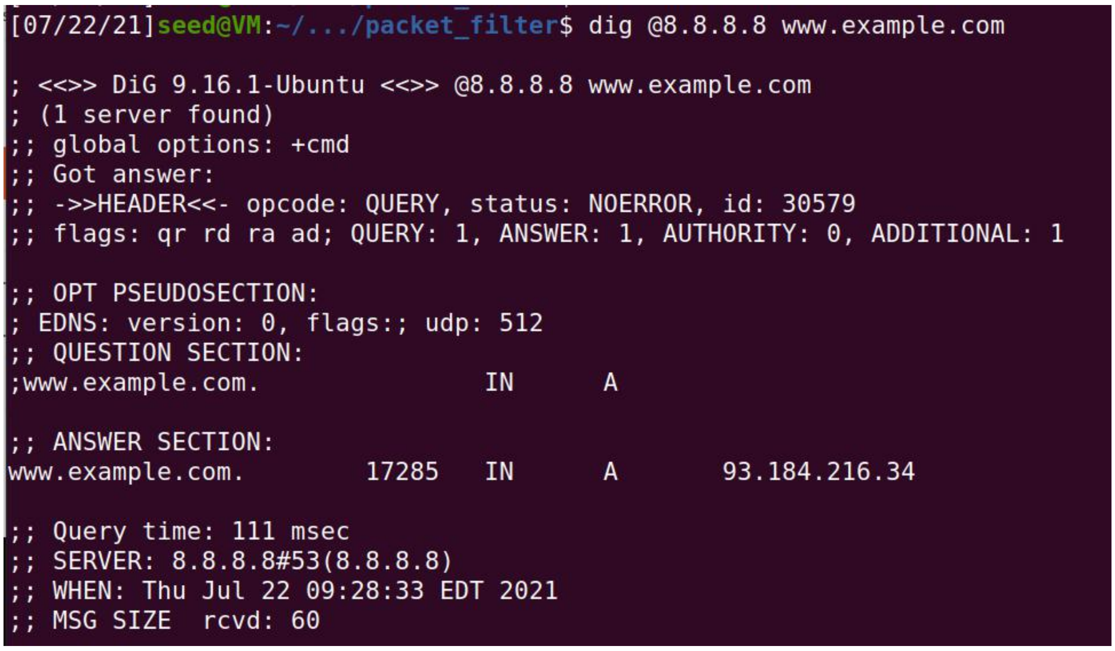


```
[  721.712222] Bye-bye World!.
```

Task 1.B: Implement a Simple Firewall Using Netfilter

1、

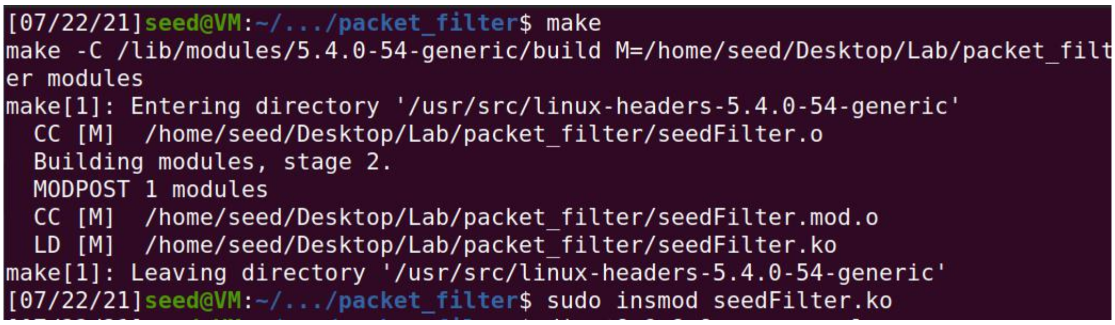
实验之前先对 8.8.8.8 进行请求尝试



```
[07/22/21]seed@VM:~/.../packet_filter$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30579
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        17285   IN      A       93.184.216.34

;; Query time: 111 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Jul 22 09:28:33 EDT 2021
;; MSG SIZE  rcvd: 60
```

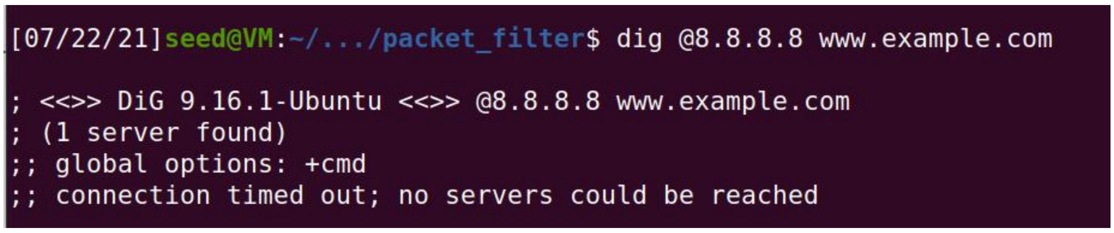
可以看到请求得到相应

编译并加载内核



```
[07/22/21]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/Lab/packet_filt
er modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/Lab/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Desktop/Lab/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/Desktop/Lab/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/22/21]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
```

加载完成后再次请求访问



```
[07/22/21]seed@VM:~/.../packet_filter$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

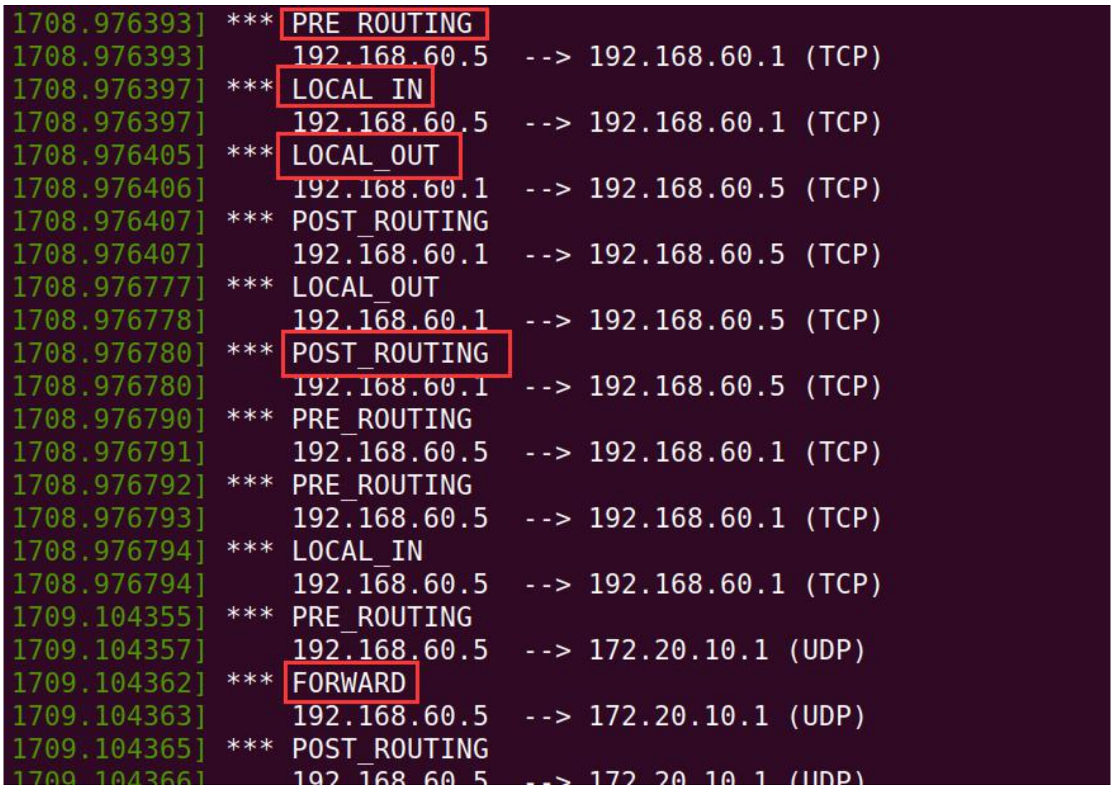
可以看到访问被拒绝

2、

定义五个钩子一一对应

```
hook1.hook = printInfo;
hook1.hooknum = NF_INET_PRE_ROUTING;
hook1.pf = PF_INET;
hook1.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook1);

hook2.hook = printInfo;
hook2.hooknum = NF_INET_LOCAL_IN;
hook2.pf = PF_INET;
hook2.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook2);

hook3.hook = printInfo;
hook3.hooknum = NF_INET_FORWARD;
hook3.pf = PF_INET;
hook3.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook3);

hook4.hook = printInfo;
hook4.hooknum = NF_INET_LOCAL_OUT;
hook4.pf = PF_INET;
hook4.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook4);

hook5.hook = printInfo;
hook5.hooknum = NF_INET_POST_ROUTING;
hook5.pf = PF_INET;
hook5.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook5);
```

编译并加载后 telnet 192.168.60.5，然后用 dmesg 查看信息



```
1708.976393] *** PRE_ROUTING
1708.976393]     192.168.60.5  --> 192.168.60.1 (TCP)
1708.976397] *** LOCAL_IN
1708.976397]     192.168.60.5  --> 192.168.60.1 (TCP)
1708.976405] *** LOCAL_OUT
1708.976406]     192.168.60.1  --> 192.168.60.5 (TCP)
1708.976407] *** POST_ROUTING
1708.976407]     192.168.60.1  --> 192.168.60.5 (TCP)
1708.976777] *** LOCAL_OUT
1708.976778]     192.168.60.1  --> 192.168.60.5 (TCP)
1708.976780] *** POST_ROUTING
1708.976780]     192.168.60.1  --> 192.168.60.5 (TCP)
1708.976790] *** PRE_ROUTING
1708.976791]     192.168.60.5  --> 192.168.60.1 (TCP)
1708.976792] *** PRE_ROUTING
1708.976793]     192.168.60.5  --> 192.168.60.1 (TCP)
1708.976794] *** LOCAL_IN
1708.976794]     192.168.60.5  --> 192.168.60.1 (TCP)
1709.104355] *** PRE_ROUTING
1709.104357]     192.168.60.5  --> 172.20.10.1 (UDP)
1709.104362] *** FORWARD
1709.104363]     192.168.60.5  --> 172.20.10.1 (UDP)
1709.104365] *** POST_ROUTING
1709.104366]     192.168.60.5  --> 172.20.10.1 (UDP)
```

从以上结果可以看出，LOCAL_IN 和 LOCAL_OUT 用于本地主机接收和发送数据包，POST_ROUTING 用于本地数据包产生后发送给目的地，PRE_ROUTING 用于来自目的地址数据包发给本地，FORWARD 用于路由器转发数据包。

NF_INET_PRE_ROUTING：除了混杂模式，所有数据包都将经过这个钩子点。它上面注册的钩子函数在路由判决之前被调用。
NF_INET_LOCAL_IN：数据包要进行路由判决，以决定需要被转发还是发往本机。前一种情况下，数据包将前往转发路径；后一种情况下，数据包将通过这个钩子点，之后被发送到网络协议栈，并最终被主机接收。
NF_INET_FORWARD：需要被转发的数据包会到达这个钩子点，这个钩子点对于实现一个防火墙是十分重要的。
NF_INET_LOCAL_OUT：这是本机产生的数据包到达的第一个钩子点。
NF_INET_POST_ROUTING：需要被转发或者由本机产生的数据包都会经过这个钩子点。源网络地址转换就是用这个钩子点实现的。

3、
阻断 ping 的函数如下

```c
unsigned int blockping(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   char ip[16] = "10.9.0.1";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_ICMP && iph->daddr == ip_addr)
   {
           printk(KERN_WARNING "*** Dropping %pI4 (ICMP)  \n", &(iph->daddr));
           return NF_DROP;
   }
   return NF_ACCEPT;
}
```

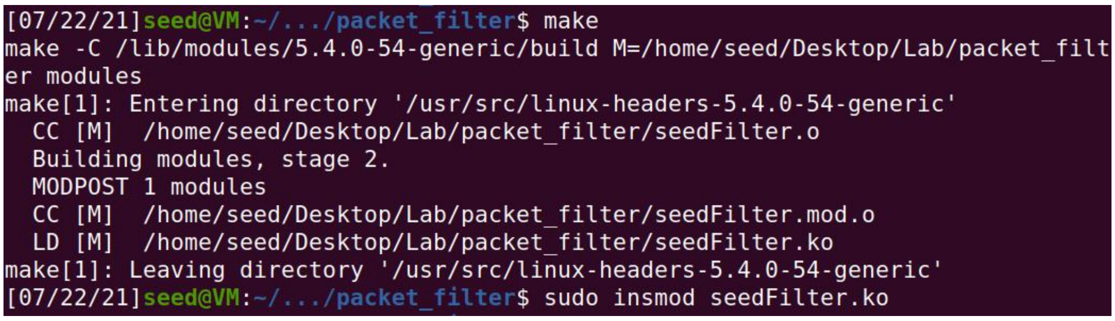
阻断 telnet 的函数如下

```c
unsigned int blocktelnet(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

   u16  port   = 23;
   char ip[16] = "10.9.0.1";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_TCP) {
       tcph = tcp_hdr(skb);
       if (iph->daddr == ip_addr && ntohs(tcph->dest) == port){
            printk(KERN_WARNING "*** Dropping %pI4 (UDP), port %d\n", &(iph->daddr), port);
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}
```

钩子定义如下

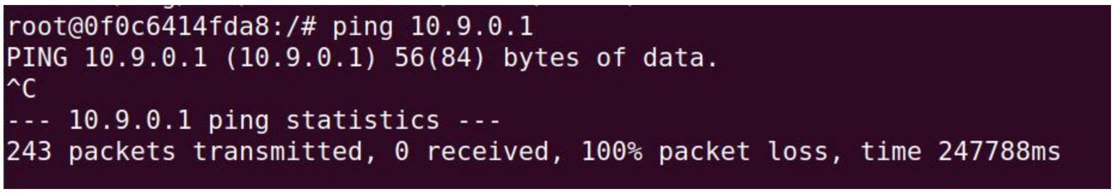
```c
hook3.hook = blockping;
hook3.hooknum = NF_INET_PRE_ROUTING;
hook3.pf = PF_INET;
hook3.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook3);

hook4.hook = blocktelnet;
hook4.hooknum = NF_INET_PRE_ROUTING;
hook4.pf = PF_INET;
hook4.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook4);
```

编译并加载

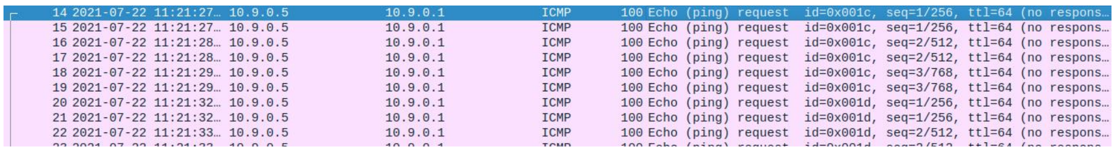


```
[07/22/21]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/Lab/packet_filt
er modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/Lab/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Desktop/Lab/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/Desktop/Lab/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[07/22/21]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
```

在 10.9.0.5 上 ping 10.9.0.1 发现无法连通

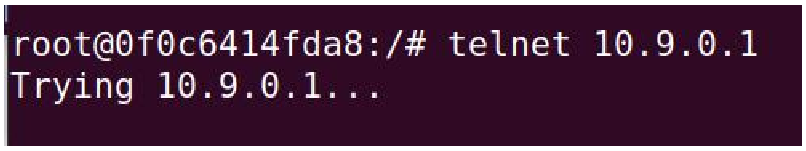


```
root@0f0c6414fda8:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
^C
--- 10.9.0.1 ping statistics ---
243 packets transmitted, 0 received, 100% packet loss, time 247788ms
```

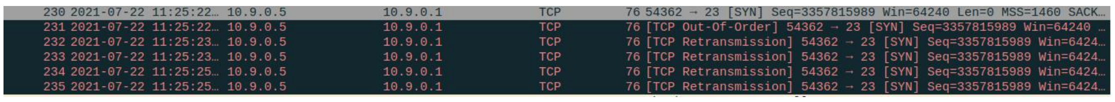


```
14 2021-07-22 11:21:27… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001c, seq=1/256, ttl=64 (no respons…
15 2021-07-22 11:21:27… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001c, seq=1/256, ttl=64 (no respons…
16 2021-07-22 11:21:28… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001c, seq=2/512, ttl=64 (no respons…
17 2021-07-22 11:21:28… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001c, seq=2/512, ttl=64 (no respons…
18 2021-07-22 11:21:29… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001c, seq=3/768, ttl=64 (no respons…
19 2021-07-22 11:21:29… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001c, seq=3/768, ttl=64 (no respons…
20 2021-07-22 11:21:32… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001d, seq=1/256, ttl=64 (no respons…
21 2021-07-22 11:21:32… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001d, seq=1/256, ttl=64 (no respons…
22 2021-07-22 11:21:33… 10.9.0.5    10.9.0.1    ICMP    100 Echo (ping) request  id=0x001d, seq=2/512, ttl=64 (no respons…
```

在 10.9.0.5 上 telnet 10.9.0.1 发现无法连通

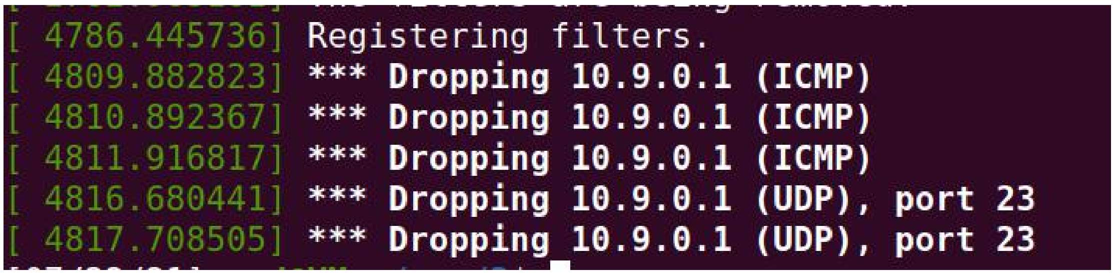


```
root@0f0c6414fda8:/# telnet 10.9.0.1
Trying 10.9.0.1...
```



```
230 2021-07-22 11:25:22… 10.9.0.5    10.9.0.1    TCP    76 54362 → 23 [SYN] Seq=3357815989 Win=64240 Len=0 MSS=1460 SACK…
231 2021-07-22 11:25:22… 10.9.0.5    10.9.0.1    TCP    76 [TCP Out-Of-Order] 54362 → 23 [SYN] Seq=3357815989 Win=64240 …
232 2021-07-22 11:25:23… 10.9.0.5    10.9.0.1    TCP    76 [TCP Retransmission] 54362 → 23 [SYN] Seq=3357815989 Win=6424…
233 2021-07-22 11:25:23… 10.9.0.5    10.9.0.1    TCP    76 [TCP Retransmission] 54362 → 23 [SYN] Seq=3357815989 Win=6424…
234 2021-07-22 11:25:25… 10.9.0.5    10.9.0.1    TCP    76 [TCP Retransmission] 54362 → 23 [SYN] Seq=3357815989 Win=6424…
235 2021-07-22 11:25:25… 10.9.0.5    10.9.0.1    TCP    76 [TCP Retransmission] 54362 → 23 [SYN] Seq=3357815989 Win=6424…
```

用 dmesg 查看信息



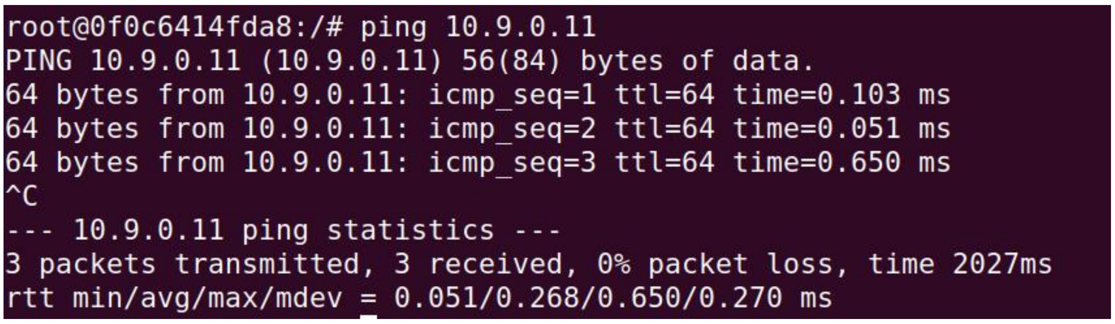
```
[ 4786.445736] Registering filters.
[ 4809.882823] *** Dropping 10.9.0.1 (ICMP)
[ 4810.892367] *** Dropping 10.9.0.1 (ICMP)
[ 4811.916817] *** Dropping 10.9.0.1 (ICMP)
[ 4816.680441] *** Dropping 10.9.0.1 (UDP), port 23
[ 4817.708505] *** Dropping 10.9.0.1 (UDP), port 23
```

说明阻断成功。

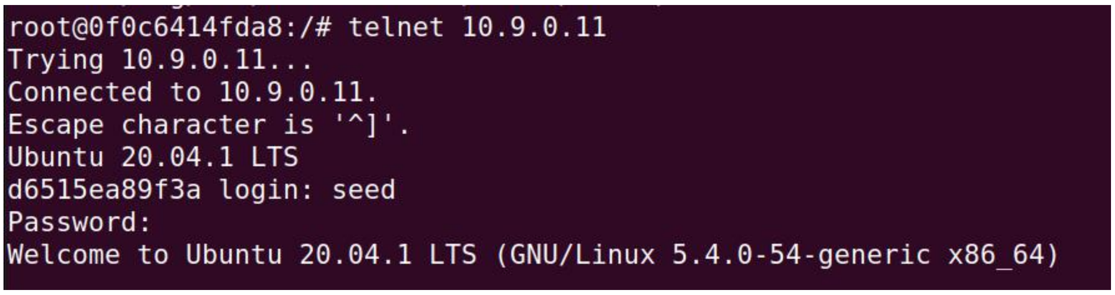
Task 2: Experimenting with Stateless Firewall Rules

Task 2.A: Protecting the Router

实验前用 10.9.0.5 去连接 10.9.0.11，可以 ping 到也可以 telnet 到

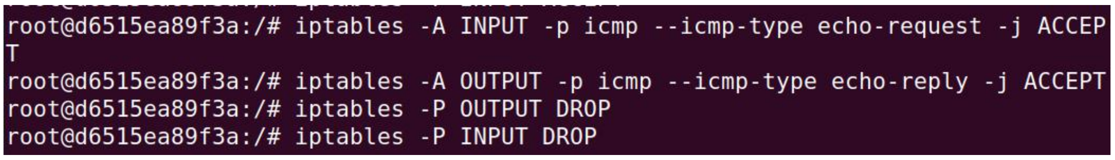


```
root@0f0c6414fda8:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.103 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.650 ms
^C
--- 10.9.0.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.051/0.268/0.650/0.270 ms
```



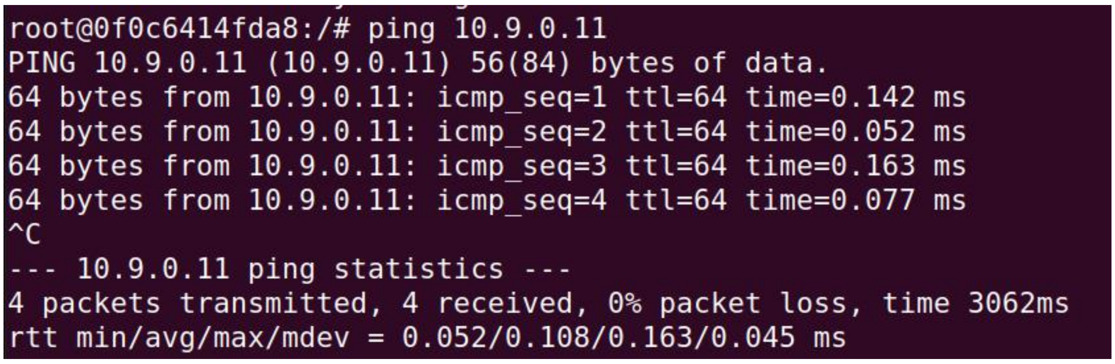
```
root@0f0c6414fda8:/# telnet 10.9.0.11
Trying 10.9.0.11...
Connected to 10.9.0.11.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d6515ea89f3a login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

在路由器中输入以下命令



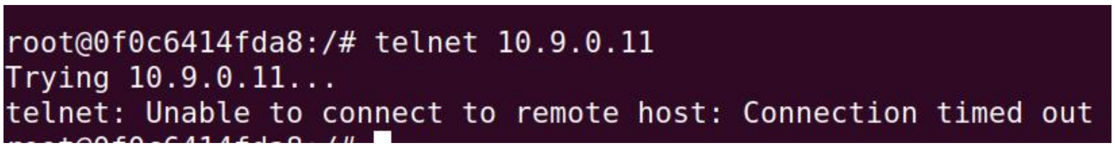
```
root@d6515ea89f3a:/# iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEP
T
root@d6515ea89f3a:/# iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@d6515ea89f3a:/# iptables -P OUTPUT DROP
root@d6515ea89f3a:/# iptables -P INPUT DROP
```

在 10.9.0.5 中 ping 路由器 10.9.0.11，ping 的到



```
root@0f0c6414fda8:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.142 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.163 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.077 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.052/0.108/0.163/0.045 ms
```

在 10.9.0.5 中 telnet 路由器，失败



```
root@0f0c6414fda8:/# telnet 10.9.0.11
Trying 10.9.0.11...
telnet: Unable to connect to remote host: Connection timed out
```

iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT 允许路由器接收请求报文

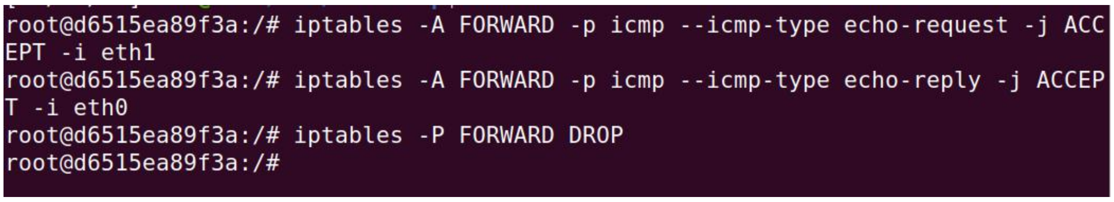
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT 允许路由器发出回答报文

iptables -P OUTPUT DROP 禁止路由器发出其他报文

iptables -P INPUT DROP 禁止路由器接收其他报文

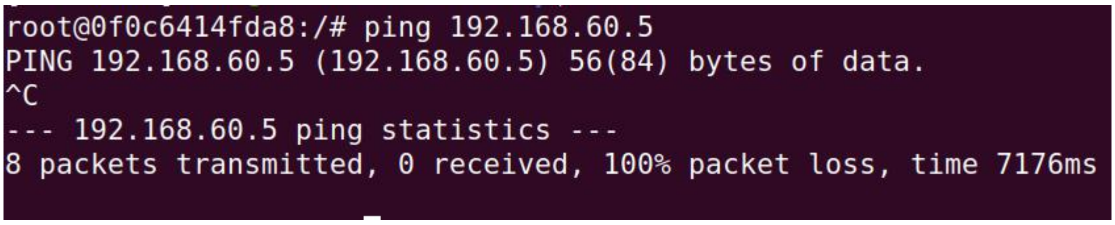
Task 2.B: Protecting the Internal Network

在路由器中设置如下规则：



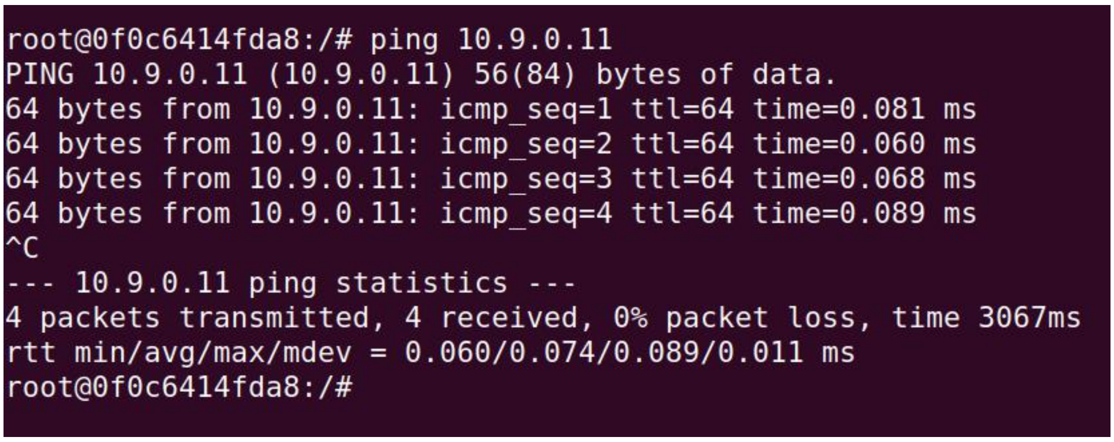
```
root@d6515ea89f3a:/# iptables -A FORWARD -p icmp --icmp-type echo-request -j ACC
EPT -i eth1
root@d6515ea89f3a:/# iptables -A FORWARD -p icmp --icmp-type echo-reply -j ACCEP
T -i eth0
root@d6515ea89f3a:/# iptables -P FORWARD DROP
root@d6515ea89f3a:/#
```

1、外部主机无法 ping 内部主机



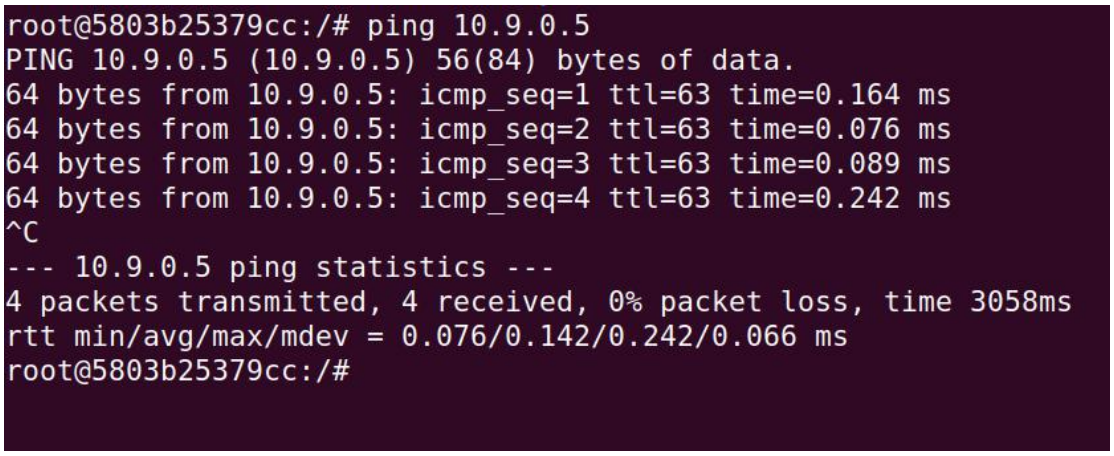
```
root@0f0c6414fda8:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7176ms
```

2、外部主机可以 ping 路由器



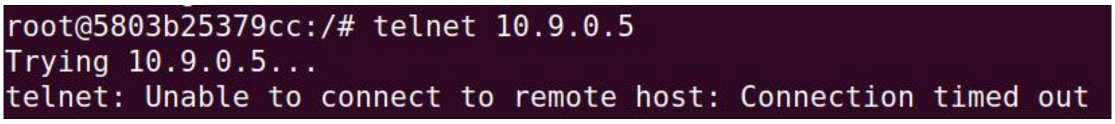
```
root@0f0c6414fda8:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.081 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.089 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.060/0.074/0.089/0.011 ms
root@0f0c6414fda8:/#
```

3、内部主机可以 ping 外部主机

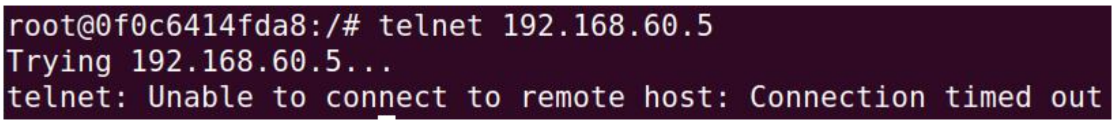


```
root@5803b25379cc:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.164 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.076 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.089 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.242 ms
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.076/0.142/0.242/0.066 ms
root@5803b25379cc:/#
```

4、其他数据包内外不能通



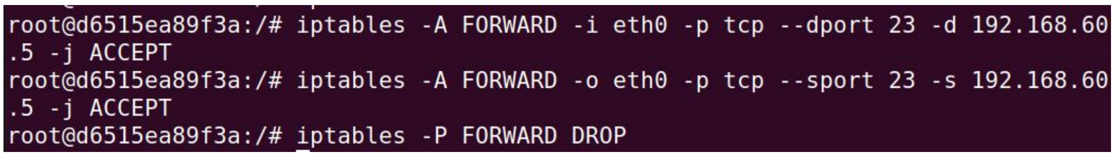
```
root@5803b25379cc:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```



```
root@0f0c6414fda8:/# telnet 192.168.60.5
Trying 192.168.60.5...
telnet: Unable to connect to remote host: Connection timed out
```

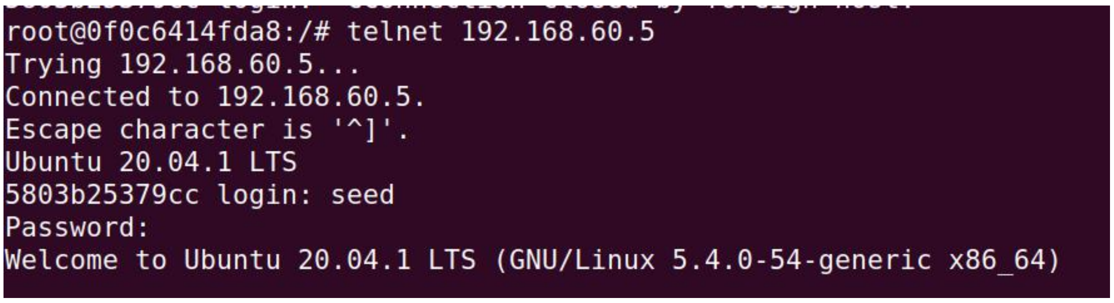
Task 2.C: Protecting Internal Servers

在路由器中设置如下规则



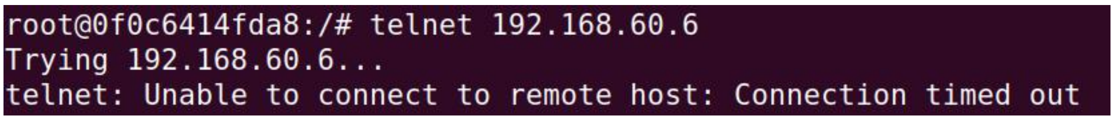
```
root@d6515ea89f3a:/# iptables -A FORWARD -i eth0 -p tcp --dport 23 -d 192.168.60
.5 -j ACCEPT
root@d6515ea89f3a:/# iptables -A FORWARD -o eth0 -p tcp --sport 23 -s 192.168.60
.5 -j ACCEPT
root@d6515ea89f3a:/# iptables -P FORWARD DROP
```

1、外部主机可以 telnet 到 192.168.60.5



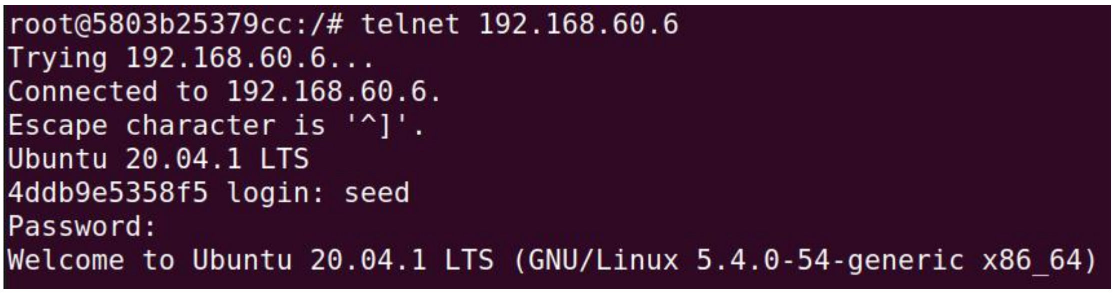
```
root@0f0c6414fda8:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5803b25379cc login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

2、外部主机无法 telnet 到其他内部主机



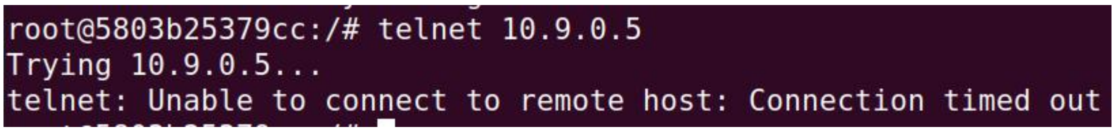
```
root@0f0c6414fda8:/# telnet 192.168.60.6
Trying 192.168.60.6...
telnet: Unable to connect to remote host: Connection timed out
```

3、内部主机可以 telnet 到其他内部主机



```
root@5803b25379cc:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4ddb9e5358f5 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

4、内部主机不能 telnet 到外部主机

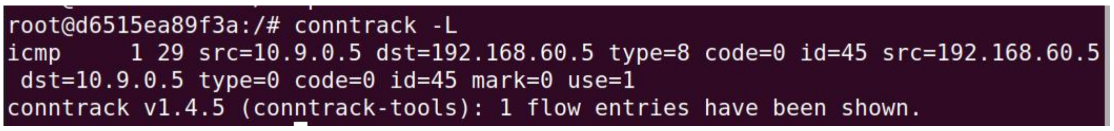


```
root@5803b25379cc:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

Task 3: Connection Tracking and Stateful Firewall

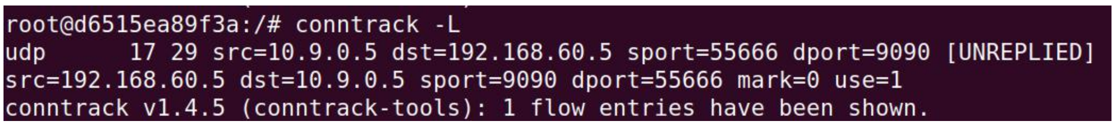
Task 3.A: Experiment with the Connection Tracking

ICMP 连接持续时间约为 30s



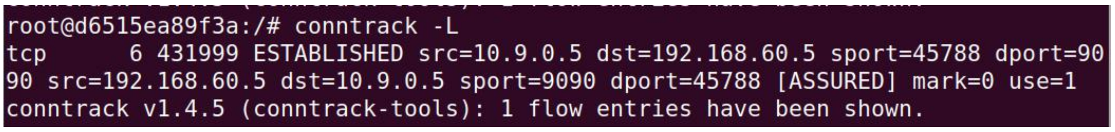
```
root@d6515ea89f3a:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=45 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=45 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

UDP 连接持续时间约为 30s



```
root@d6515ea89f3a:/# conntrack -L
udp      17 29 src=10.9.0.5 dst=192.168.60.5 sport=55666 dport=9090 [UNREPLIED]
src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=55666 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

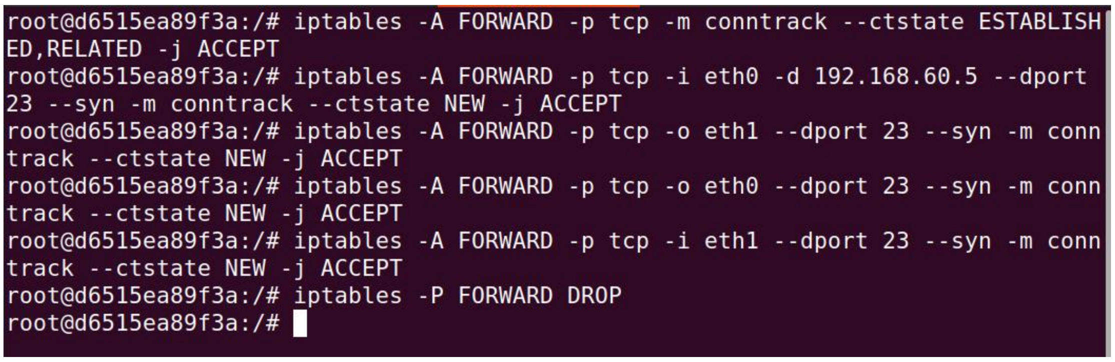
TCP 连接持续时间约为 120 小时



```
root@d6515ea89f3a:/# conntrack -L
tcp      6 431999 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=45788 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=45788 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

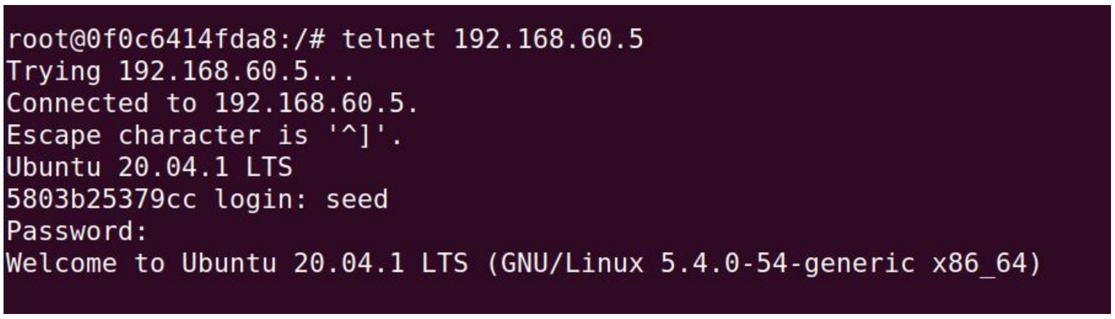
Task 3.B: Setting Up a Stateful Firewall

路由器配置如下规则



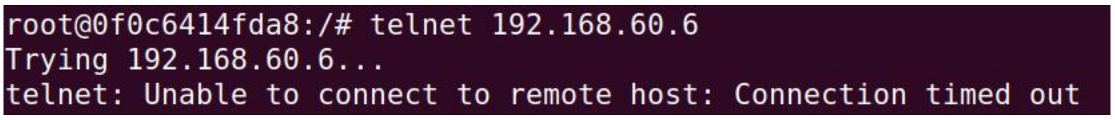
```
root@d6515ea89f3a:/# iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISH
ED,RELATED -j ACCEPT
root@d6515ea89f3a:/# iptables -A FORWARD -p tcp -i eth0 -d 192.168.60.5 --dport
23 --syn -m conntrack --ctstate NEW -j ACCEPT
root@d6515ea89f3a:/# iptables -A FORWARD -p tcp -o eth1 --dport 23 --syn -m conn
track --ctstate NEW -j ACCEPT
root@d6515ea89f3a:/# iptables -A FORWARD -p tcp -o eth0 --dport 23 --syn -m conn
track --ctstate NEW -j ACCEPT
root@d6515ea89f3a:/# iptables -A FORWARD -p tcp -i eth1 --dport 23 --syn -m conn
track --ctstate NEW -j ACCEPT
root@d6515ea89f3a:/# iptables -P FORWARD DROP
root@d6515ea89f3a:/# 
```

外部主机可以 telnet 到内部主机



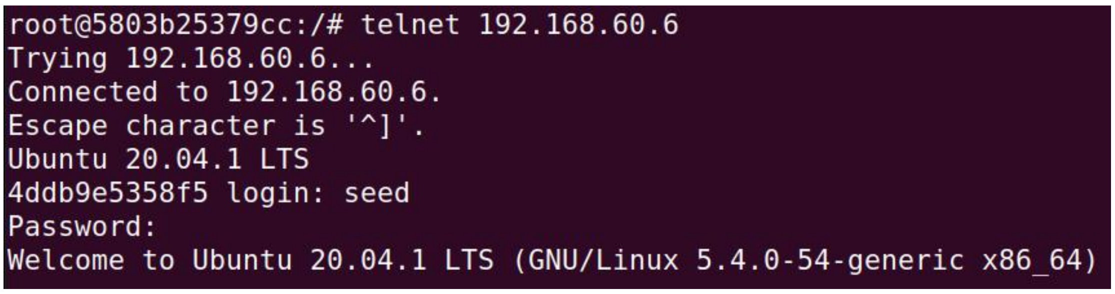
```
root@0f0c6414fda8:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5803b25379cc login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

外部主机无法 telnet 到其他内部主机



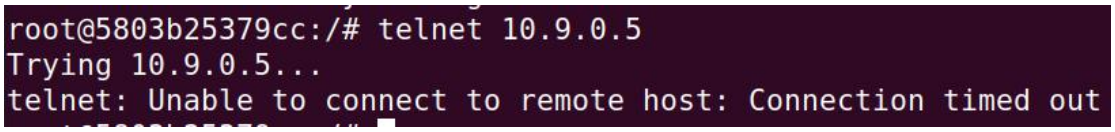
```
root@0f0c6414fda8:/# telnet 192.168.60.6
Trying 192.168.60.6...
telnet: Unable to connect to remote host: Connection timed out
```

内部主机可以 telnet 到其他内部主机



```
root@5803b25379cc:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4ddb9e5358f5 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

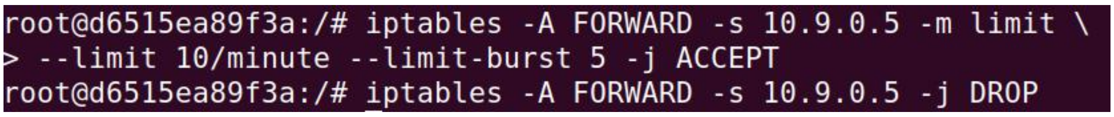
内部主机不能 telnet 到外部主机



```
root@5803b25379cc:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

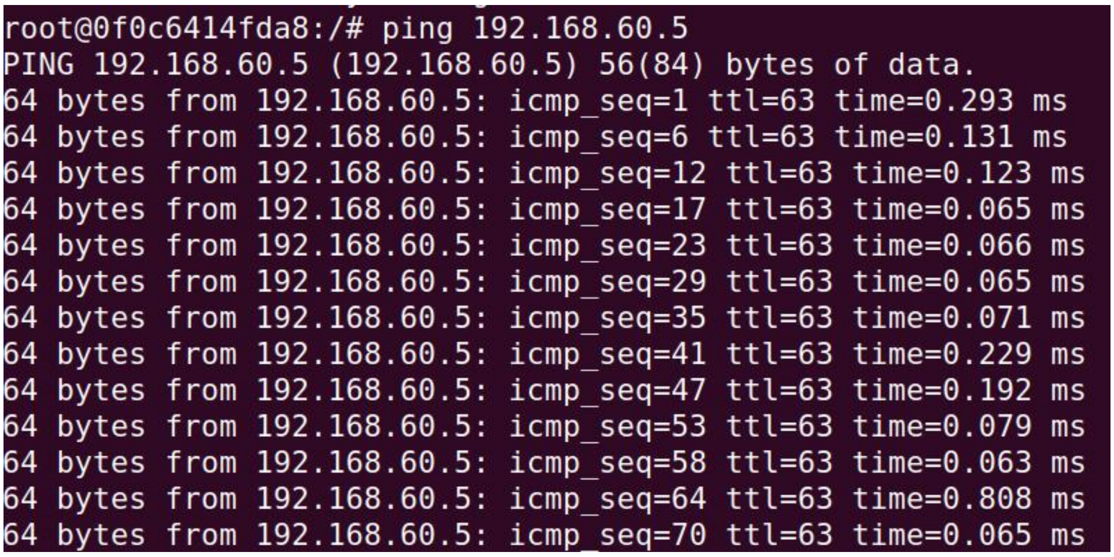
两者区别：基于跟踪连接的防火墙只需要在建立连接的时候判定是否合法，之后的报文只需要判定是否建立连接即可，而不使用跟踪连接的防火墙则需要对所有报文进行判定是否合法。

Task 4: Limiting Network Traffific



```
root@d6515ea89f3a:/# iptables -A FORWARD -s 10.9.0.5 -m limit \
> --limit 10/minute --limit-burst 5 -j ACCEPT
root@d6515ea89f3a:/# iptables -A FORWARD -s 10.9.0.5 -j DROP
```

在路由器配置上述两条命令之后，从主机 10.9.0.5 去 ping 主机 192.168.60.5



```
root@0f0c6414fda8:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.293 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.131 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.123 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.065 ms
64 bytes from 192.168.60.5: icmp_seq=23 ttl=63 time=0.066 ms
64 bytes from 192.168.60.5: icmp_seq=29 ttl=63 time=0.065 ms
64 bytes from 192.168.60.5: icmp_seq=35 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=41 ttl=63 time=0.229 ms
64 bytes from 192.168.60.5: icmp_seq=47 ttl=63 time=0.192 ms
64 bytes from 192.168.60.5: icmp_seq=53 ttl=63 time=0.079 ms
64 bytes from 192.168.60.5: icmp_seq=58 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=64 ttl=63 time=0.808 ms
64 bytes from 192.168.60.5: icmp_seq=70 ttl=63 time=0.065 ms
```

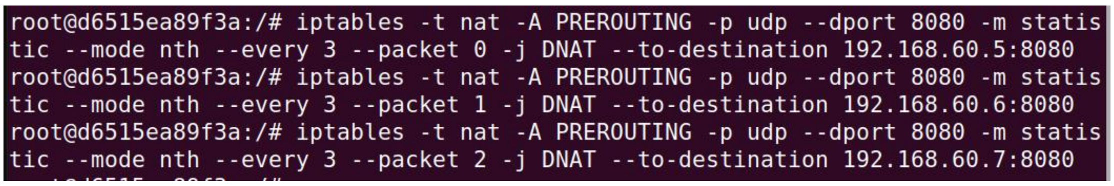
一开始会收到几个回复比较快的报文，然后就需要等一段时间才能收到下一个回复的报文。

在去掉第二条命令之后，就会收到连续的回复报文。所以没有第二条命令的话，第一条命令的限制就不会起到作用。原因是超出限制的报文是由第二条命令处理的，没有第二条命令的话路由器还是会接受超出限制的报文。

Task 5: Load Balancing

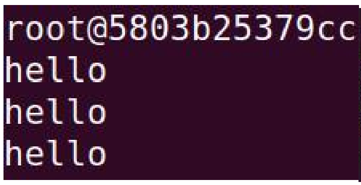
Using the nth mode (round-robin)

在路由器配置如下命令

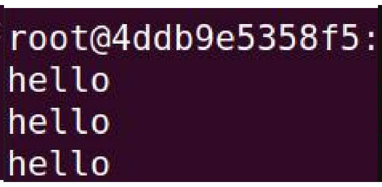


```
root@d6515ea89f3a:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
root@d6515ea89f3a:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 3 --packet 1 -j DNAT --to-destination 192.168.60.6:8080
root@d6515ea89f3a:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 3 --packet 2 -j DNAT --to-destination 192.168.60.7:8080
```

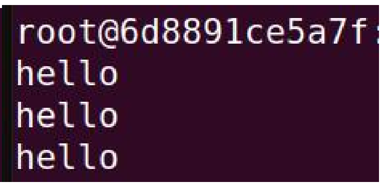
在路由器和 192.168.60.0 各主机中输入 nc -luk 8080，然后用 10.9.0.5 不断向 10.9.0.11 发送 hello，此时 192.168.60.0 按顺序依次收到 hello



```
root@5803b25379cc
hello
hello
hello
```



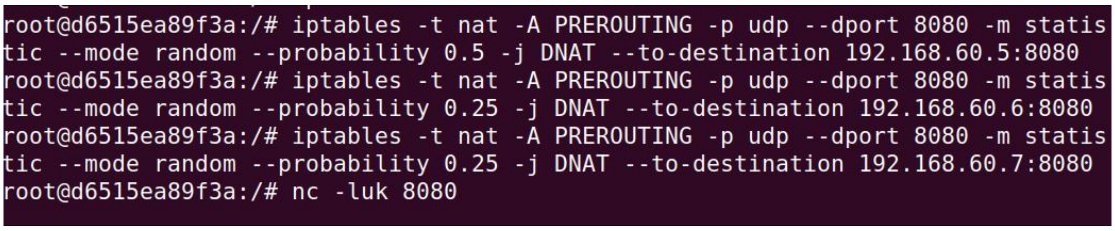
```
root@4ddb9e5358f5:
hello
hello
hello
```



```
root@6d8891ce5a7f
hello
hello
hello
```

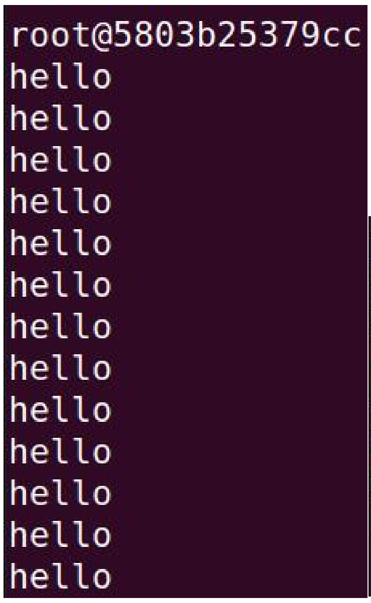
Using the random mode

在路由器配置如下命令

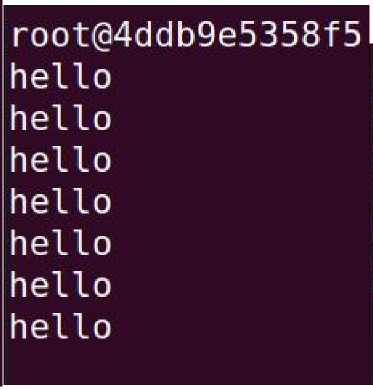


```
root@d6515ea89f3a:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.5 -j DNAT --to-destination 192.168.60.5:8080
root@d6515ea89f3a:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.25 -j DNAT --to-destination 192.168.60.6:8080
root@d6515ea89f3a:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.25 -j DNAT --to-destination 192.168.60.7:8080
root@d6515ea89f3a:/# nc -luk 8080
```

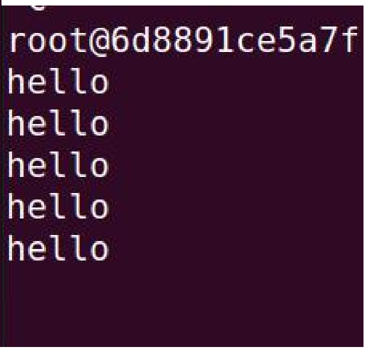
在这个情况下，发送的 hello 有 50%的概率发向 192.168.60.5，有 25%的概率发向 192.168.60.6，有 25%的概率发向 192.168.60.7



```
root@5803b25379cc
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```



```
root@4ddb9e5358f5
hello
hello
hello
hello
hello
hello
hello
```



```
root@6d8891ce5a7f
hello
hello
hello
hello
hello
```

从实验结果可以看出，hello 的数量大致为 2:1:1 的关系