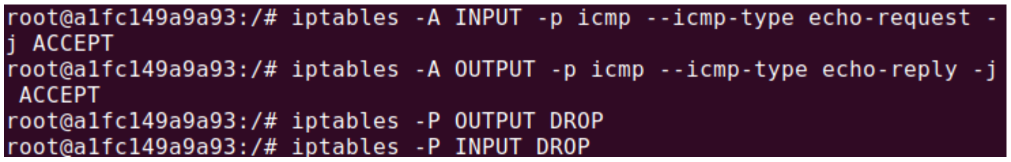
Task 2: Experimenting with Stateless Firewall Rules

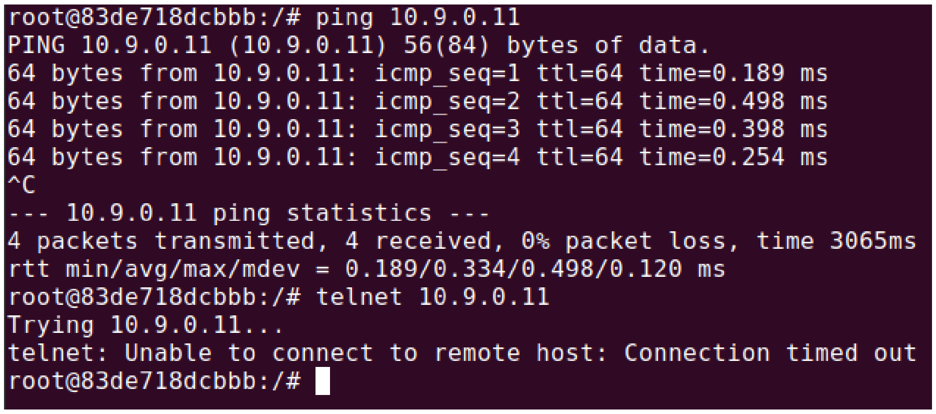
Task 2.A: Protecting the Router

On seed-router (10.9.0.11):



```
root@a1fc149a9a93:/# iptables -A INPUT -p icmp --icmp-type echo-request -
j ACCEPT
root@a1fc149a9a93:/# iptables -A OUTPUT -p icmp --icmp-type echo-reply -j
 ACCEPT
root@a1fc149a9a93:/# iptables -P OUTPUT DROP
root@a1fc149a9a93:/# iptables -P INPUT DROP
```

On 10.9.0.5:



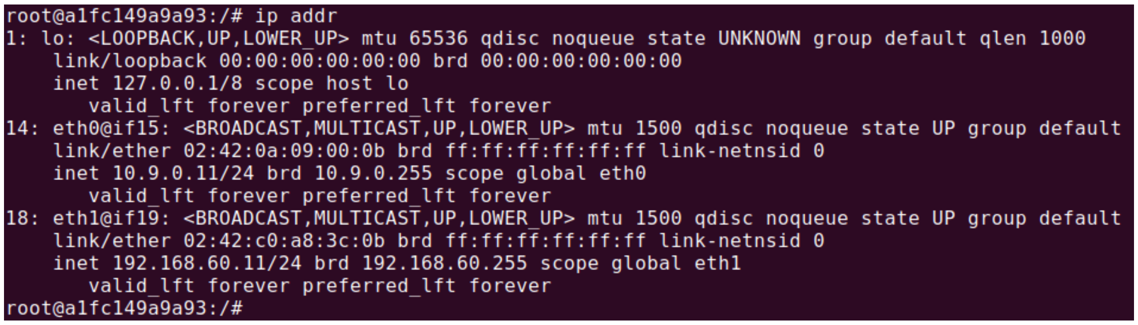
```
root@83de718dcbbb:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.189 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.498 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.398 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.254 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3065ms
rtt min/avg/max/mdev = 0.189/0.334/0.498/0.120 ms
root@83de718dcbbb:/# telnet 10.9.0.11
Trying 10.9.0.11...
telnet: Unable to connect to remote host: Connection timed out
root@83de718dcbbb:/#
```

We can see that I can successfully ping the router (10.9.0.11) from 10.9.0.5, but unable to telnet into the router as the connection timed out.
Based on the 4 rules, every incoming packet to 10.9.0.11 is dropped, except for ICMP echo requests and reply, hence pings was successful, and telnet was not.

- "iptables -A INPUT -p icmp –icmp-type echo-request -j ACCEPT" – accepts incoming ICMP echo request packets (ping) to seed-router
- "iptables -A OUTPUT -p icmp –icmp-type echo-reply -j ACCEPT" – accepts outgoing ICMP echo reply packets (ping) from seed-router
- "iptables -P OUTPUT DROP" – sets default policy for outgoing packets to be dropped except the echo-reply rule stated above
- "iptables -P INPUT DROP" – sets default policy for incoming packets to be dropped except the echo-request rule stated above

Task 2.B: Protecting the Internal Network

The interfaces on seed-router:

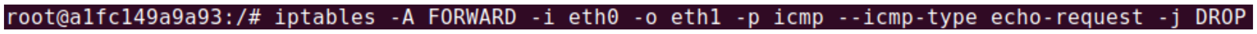


```
root@a1fc149a9a93:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
14: eth0@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.11/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
18: eth1@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a8:3c:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.60.11/24 brd 192.168.60.255 scope global eth1
       valid_lft forever preferred_lft forever
root@a1fc149a9a93:/#
```

We can see that the external interface is eth0(10.9.0.11/24), while the internal interface is eth1 (192.168.60.11/24).

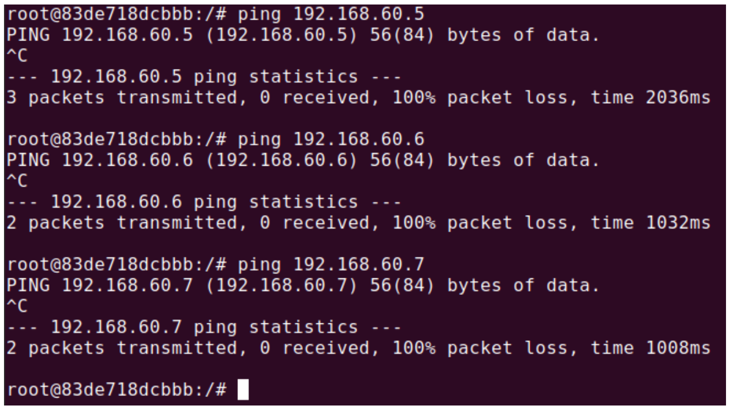
1. Outside hosts cannot ping internal hosts

To prevent outside hosts from pinging the internal hosts, we need to drop incoming ICMP echo requests that are coming from the external network interface, eth0 and entering to the internal network interface, eth1.



```
root@a1fc149a9a93:/# iptables -A FORWARD -i eth0 -o eth1 -p icmp --icmp-type echo-request -j DROP
```

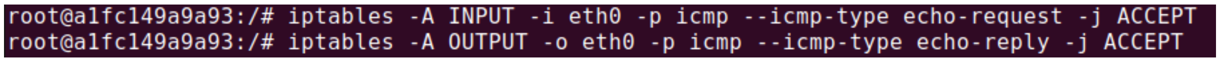
Now, we are not able to ping host1, host2, and host3 from hostA as seen from the picture below.



```
root@83de718dcbbb:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2036ms

root@83de718dcbbb:/# ping 192.168.60.6
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
^C
--- 192.168.60.6 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1032ms

root@83de718dcbbb:/# ping 192.168.60.7
PING 192.168.60.7 (192.168.60.7) 56(84) bytes of data.
^C
--- 192.168.60.7 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms

root@83de718dcbbb:/#
```

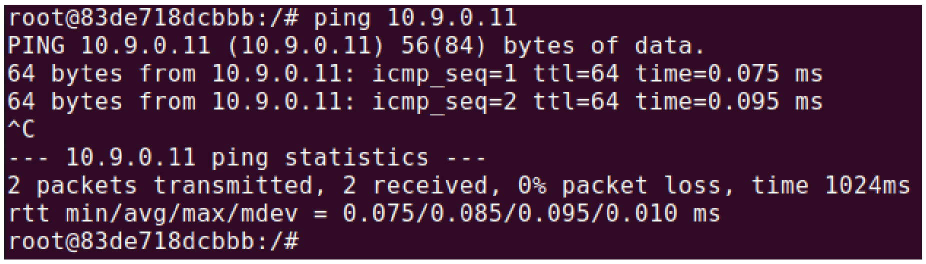
2. Outside hosts can ping the router

We need to accept incoming ICMP echo requests that goes to the router itself (from eth0), while also accepting outgoing ICMP echo replies from the router to the external network interface (eth0).



```
root@a1fc149a9a93:/# iptables -A INPUT -i eth0 -p icmp --icmp-type echo-request -j ACCEPT
root@a1fc149a9a93:/# iptables -A OUTPUT -o eth0 -p icmp --icmp-type echo-reply -j ACCEPT
```

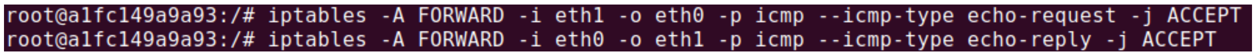
Now, the outside hosts can successfully ping the router.



```
root@83de718dcbbb:/# 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=0.075 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.095 ms
^C
--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1024ms
rtt min/avg/max/mdev = 0.075/0.085/0.095/0.010 ms
root@83de718dcbbb:/#
```

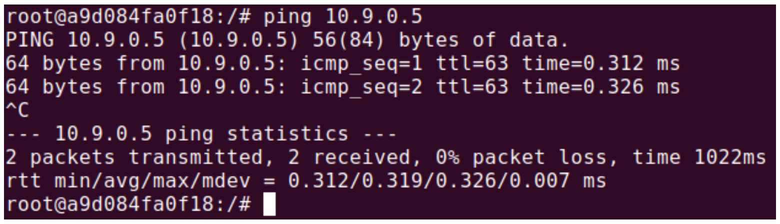
3. Internal hosts can ping outside hosts

We need to use the FORWARD chain, where we accept ICMP echo requests from the internal network interface, eth1, to the external network interface, eth0. We also need to accept ICMP echo replies from the external network interface, eth0, to the internal network interface, eth1.



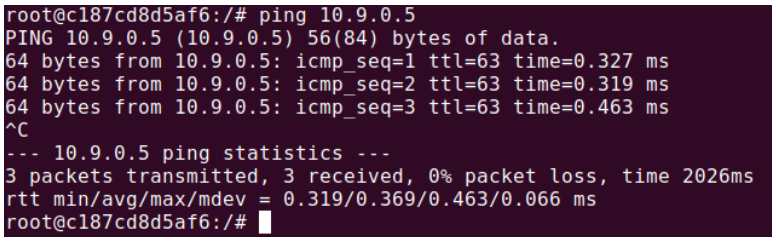
```
root@a1fc149a9a93:/# iptables -A FORWARD -i eth1 -o eth0 -p icmp --icmp-type echo-request -j ACCEPT
root@a1fc149a9a93:/# iptables -A FORWARD -i eth0 -o eth1 -p icmp --icmp-type echo-reply -j ACCEPT
```

On host1, we can successfully ping the outside host, 10.9.0.5.



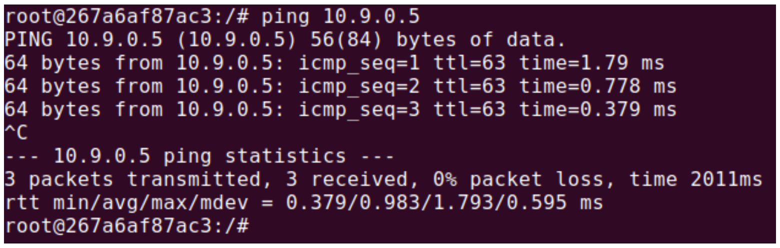
```
root@a9d084fa0f18:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.312 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.326 ms
^C
--- 10.9.0.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1022ms
rtt min/avg/max/mdev = 0.312/0.319/0.326/0.007 ms
root@a9d084fa0f18:/#
```

On host2, we can successfully ping the outside host, 10.9.0.5.



```
root@c187cd8d5af6:/# 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=0.327 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.319 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.463 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.319/0.369/0.463/0.066 ms
root@c187cd8d5af6:/#
```

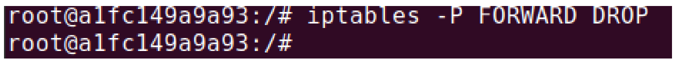
On host3, we can successfully ping the outside host, 10.9.0.5.



```
root@267a6af87ac3:/# 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=1.79 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.778 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.379 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 0.379/0.983/1.793/0.595 ms
root@267a6af87ac3:/#
```

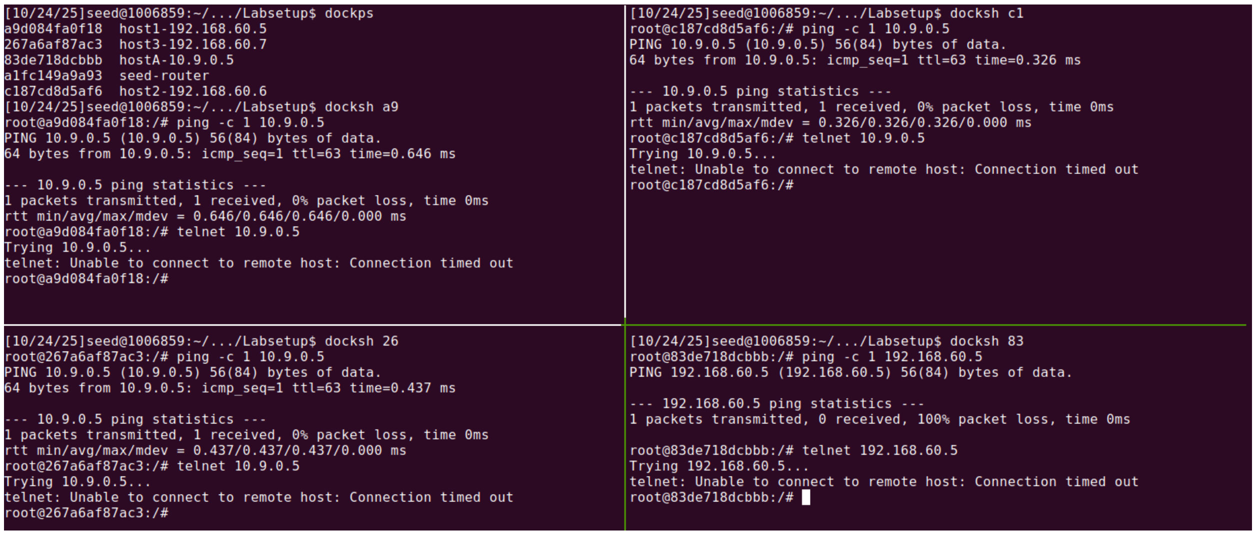
4. All other packets between the internal and external networks should be blocked.

We can set the default policy for the FORWARD chain to drop all packets unless stated otherwise.



```
root@a1fc149a9a93:/# iptables -P FORWARD DROP
root@a1fc149a9a93:/#
```

From the picture below, we can see that we can still ping from internal hosts to external hosts, but we cannot ping from an external host to the internal network. We are also unable to telnet regardless of which network we are working from, as all other packets between the internal and external networks are blocked.



```
[10/24/25]seed@1006859:~/.../Labsetup$ dockps
a9d084fa0f18  host1-192.168.60.5
267a6af87ac3  host3-192.168.60.7
83de718dcbbb  hostA-10.9.0.5
a1fc149a9a93  seed-router
c187cd8d5af6  host2-192.168.60.6
[10/24/25]seed@1006859:~/.../Labsetup$ docksh a9
root@a9d084fa0f18:/# ping -c 1 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=0.646 ms

--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.646/0.646/0.646/0.000 ms
root@a9d084fa0f18:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@a9d084fa0f18:/#
```

```
[10/24/25]seed@1006859:~/.../Labsetup$ docksh c1
root@c187cd8d5af6:/# ping -c 1 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=0.326 ms

--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.326/0.326/0.326/0.000 ms
root@c187cd8d5af6:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@c187cd8d5af6:/#
```

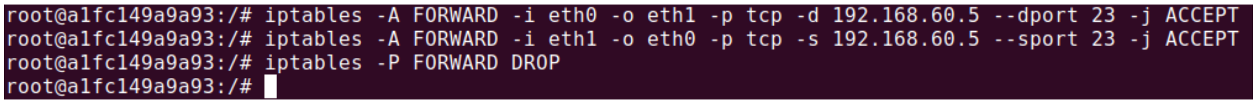
```
[10/24/25]seed@1006859:~/.../Labsetup$ docksh 26
root@267a6af87ac3:/# ping -c 1 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.437 ms

--- 10.9.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.437/0.437/0.437/0.000 ms
root@267a6af87ac3:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@267a6af87ac3:/#
```

```
[10/24/25]seed@1006859:~/.../Labsetup$ docksh 83
root@83de718dcbbb:/# ping -c 1 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

root@83de718dcbbb:/# telnet 192.168.60.5
Trying 192.168.60.5...
telnet: Unable to connect to remote host: Connection timed out
root@83de718dcbbb:/#
```

The top-left picture is host1, top-right picture is host2, bottom-left picture is host3, bottom-right picture is hostA.

Task 2.C: Protecting Internal Servers

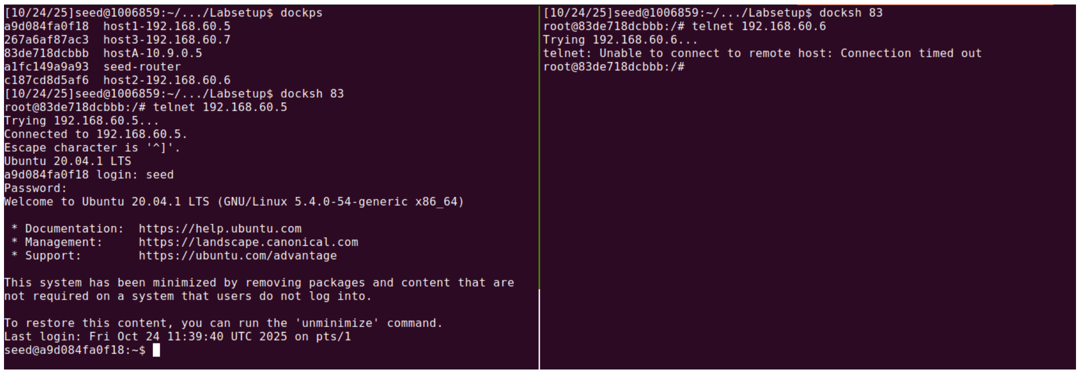
To satisfy all the objectives, we should only accept tcp packets coming from eth0 to eth1, with destination IP 192.168.60.5, and destination port 23. We should also only accept tcp packets coming from eth1 to eth0, with source IP 192.168.60.5, and source port 23.
The first rule will allow outside hosts to only access the telnet server on 192.168.60.5, and not the other internal servers.
The second rule will ensure that other internal hosts cannot access the external servers. Internal hosts can access all the internal servers since the default policy for INPUT and OUTPUT is ACCEPT.



```
root@a1fc149a9a93:/# iptables -A FORWARD -i eth0 -o eth1 -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT
root@a1fc149a9a93:/# iptables -A FORWARD -i eth1 -o eth0 -p tcp -s 192.168.60.5 --sport 23 -j ACCEPT
root@a1fc149a9a93:/# iptables -P FORWARD DROP
root@a1fc149a9a93:/#
```

The third rule sets the default policy for the FORWARD chain to DROP packets apart from the rules we explicitly stated so that the internal and external servers cannot access one another.

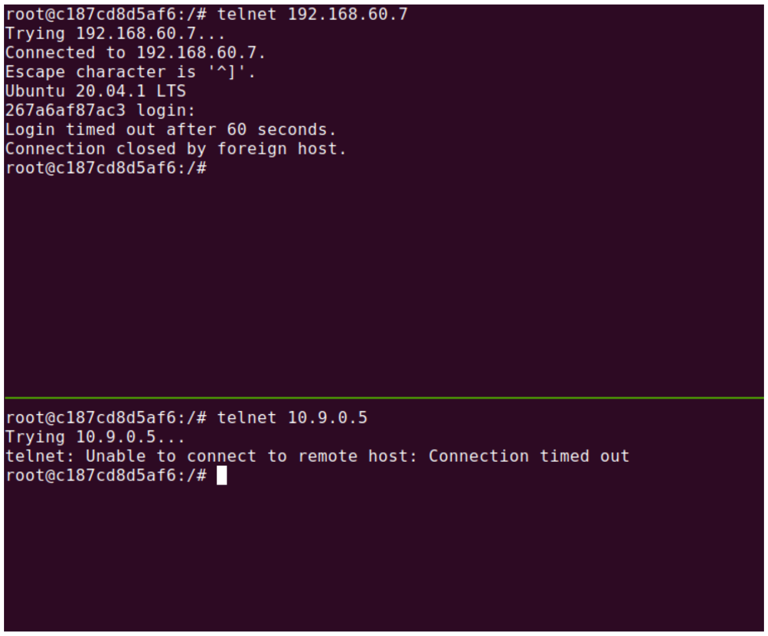
From the picture below, we can see that hostA (external) can access the telnet server on 192.168.60.5, but unable to access other internal servers.



```
[10/24/25]seed@1006859:~/.../Labsetup$ dockps
a9d084fa0f18  host1-192.168.60.5
267a6af87ac3  host3-192.168.60.7
83de718dcbbb  hostA-10.9.0.5
a1fc149a9a93  seed-router
c187cd8d5af6  host2-192.168.60.6
[10/24/25]seed@1006859:~/.../Labsetup$ docksh 83
root@83de718dcbbb:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
a9d084fa0f18 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Oct 24 11:39:40 UTC 2025 on pts/1
seed@a9d084fa0f18:~$
```

```
[10/24/25]seed@1006859:~/.../Labsetup$ docksh 83
root@83de718dcbbb:/# telnet 192.168.60.6
Trying 192.168.60.6...
telnet: Unable to connect to remote host: Connection timed out
root@83de718dcbbb:/#
```

From the picture below, we can see that host2 (internal server) can access other internal servers like host3, but is unable to access external servers.



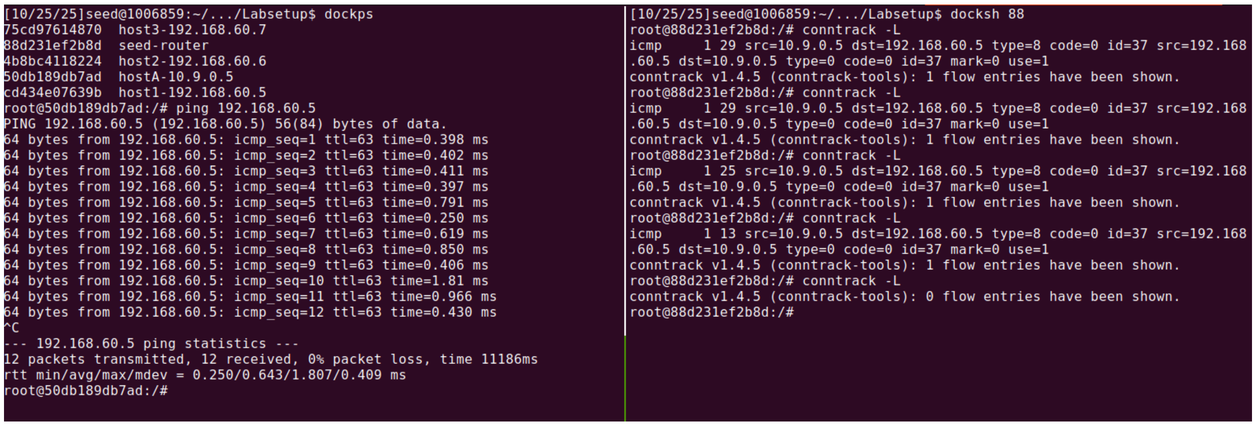
```
root@c187cd8d5af6:/# telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
267a6af87ac3 login:
Login timed out after 60 seconds.
Connection closed by foreign host.
root@c187cd8d5af6:/#
```

```
root@c187cd8d5af6:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@c187cd8d5af6:/#
```

Task 3: Connection Tracking and Stateful Firewall

Task 3.A: Experiment with the Connection Tracking

ICMP experiment

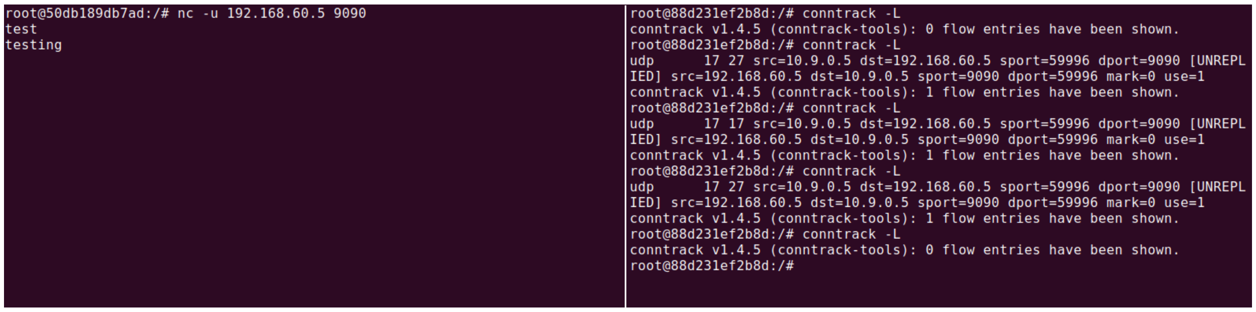


```
[10/25/25]seed@1006859:~/.../Labsetup$ dockps
75cd97614870  host3-192.168.60.7
88d231ef2b8d  seed-router
4b8bc4118224  host2-192.168.60.6
50db189db7ad  hostA-10.9.0.5
cd434e07639b  host1-192.168.60.5
root@50db189db7ad:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.398 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.402 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.411 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.397 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.791 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.250 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.619 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.850 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.406 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=1.81 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.966 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.430 ms
^C
--- 192.168.60.5 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11186ms
rtt min/avg/max/mdev = 0.250/0.643/1.807/0.409 ms
root@50db189db7ad:/#
```

```
[10/25/25]seed@1006859:~/.../Labsetup$ docksh 88
root@88d231ef2b8d:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=37 src=192.168
.60.5 dst=10.9.0.5 type=0 code=0 id=37 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=37 src=192.168
.60.5 dst=10.9.0.5 type=0 code=0 id=37 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
icmp     1 25 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=37 src=192.168
.60.5 dst=10.9.0.5 type=0 code=0 id=37 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
icmp     1 13 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=37 src=192.168
.60.5 dst=10.9.0.5 type=0 code=0 id=37 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
root@88d231ef2b8d:/#
```

When 10.9.0.5 is pinged, there is a temporary connection based on the connection tracking information. It is an ICMP connection that sends ICMP echo requests from 10.9.0.5 to 192.168.60.5, and sends ICMP echo replies from 192.168.60.5 to 10.9.0.5.
The first “conntrack -L” has a 29 seconds timeout, and continues to remain at 29 seconds when I continue pinging 192.168.60.5. After the ping was stopped, the timeout decreases from 29 to 25 to 13, then the connection closes. So, the ICMP connection state was kept for 29 seconds.

UDP experiment

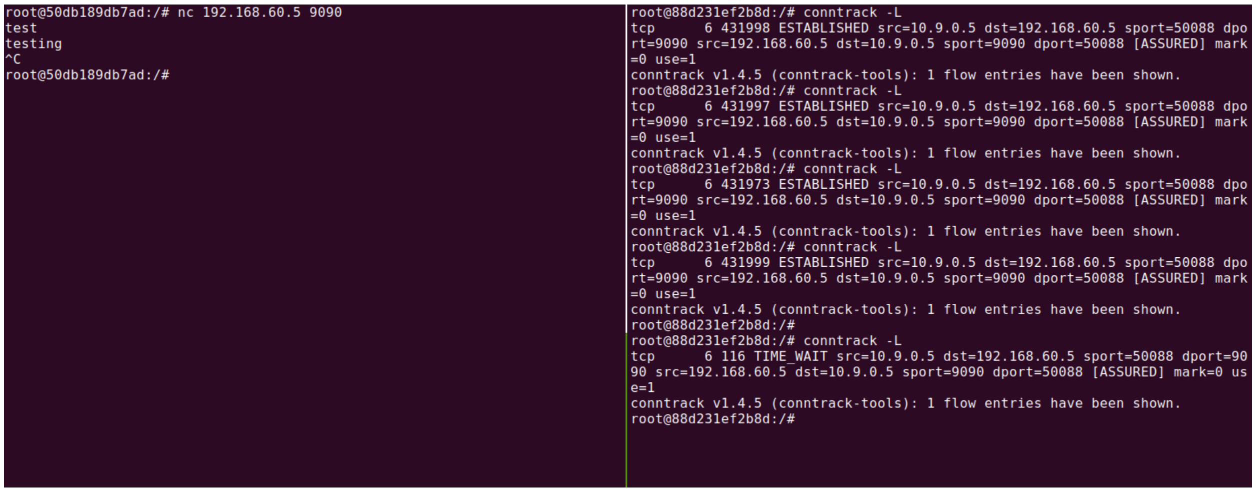


```
root@50db189db7ad:/# nc -u 192.168.60.5 9090
test
testing
```

```
root@88d231ef2b8d:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
udp      17 27 src=10.9.0.5 dst=192.168.60.5 sport=59996 dport=9090 [UNREPL
IED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=59996 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
udp      17 17 src=10.9.0.5 dst=192.168.60.5 sport=59996 dport=9090 [UNREPL
IED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=59996 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
udp      17 27 src=10.9.0.5 dst=192.168.60.5 sport=59996 dport=9090 [UNREPL
IED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=59996 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
root@88d231ef2b8d:/#
```

When I connected to 192.168.60.5 without sending any message, there is no connection between 10.9.0.5 and 192.168.60.5. After I sent a UDP packet, there is a UDP connection found on the router from 10.9.0.5 to 192.168.60.5, from port 59996 to port 9090. Although, there is no reply from 192.168.60.5.
The first conntrack -L shows that the timeout was 27 seconds for the UDP connection, which refreshes after I send another UDP packet to 192.168.60.5, else it is removed when it hits 0 seconds. So, the UDP connection state is kept for 27 seconds.

TCP experiment

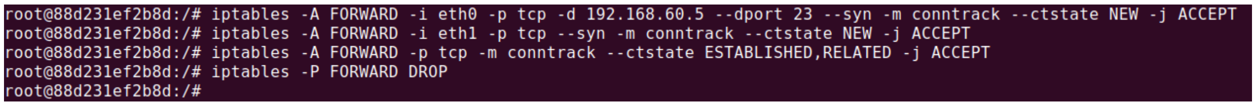


```
root@50db189db7ad:/# nc 192.168.60.5 9090
test
testing
^C
root@50db189db7ad:/#
```

```
root@88d231ef2b8d:/# conntrack -L
tcp      6 431998 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=50088 dpo
rt=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=50088 [ASSURED] mark
=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
tcp      6 431997 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=50088 dpo
rt=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=50088 [ASSURED] mark
=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
tcp      6 431973 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=50088 dpo
rt=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=50088 [ASSURED] mark
=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/# conntrack -L
tcp      6 431999 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=50088 dpo
rt=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=50088 [ASSURED] mark
=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/#
root@88d231ef2b8d:/# conntrack -L
tcp      6 116 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=50088 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=50088 [ASSURED] mark=0 us
e=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@88d231ef2b8d:/#
```

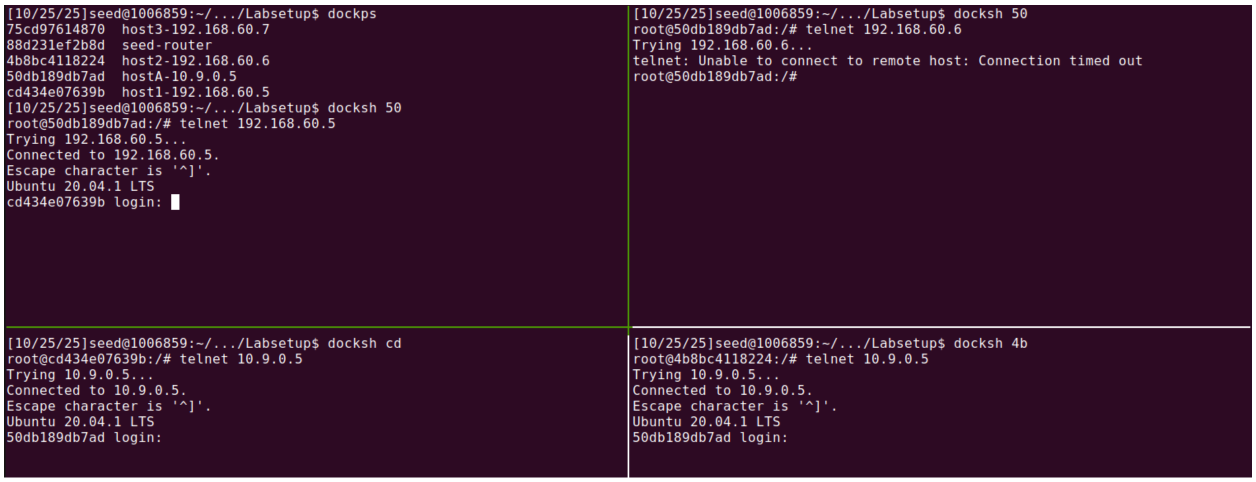
When I connected to 192.168.60.5, there is a TCP connection set up between 10.9.0.5:50088 and 192.168.60.5:9090, with state ESTABLISHED. It begins with a TCP connection state of timeout 431999 seconds, and refreshes every time a message is sent.
After the connection is closed, "conntrack -L" shows that there is still an existing TCP connection that is in state TIME_WAIT between both hosts, with a timeout of about 116 seconds. After the TIME_WAIT timeout, the connection is finally closed between both hosts.

Task 3.B: Setting Up a Stateful Firewall



```
root@88d231ef2b8d:/# iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport 23 --syn -m conntrack --ctstate NEW -j ACCEPT
root@88d231ef2b8d:/# iptables -A FORWARD -i eth1 -p tcp --syn -m conntrack --ctstate NEW -j ACCEPT
root@88d231ef2b8d:/# iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
root@88d231ef2b8d:/# iptables -P FORWARD DROP
root@88d231ef2b8d:/#
```

The first command allows incoming SYN packets from external hosts, only if they are headed towards the telnet server on 192.168.60.5 on port 23.
The second command allows incoming SYN packets from internal hosts regardless of their destination to external hosts, so internal hosts can form TCP connections with any external server.
The third command allows TCP packets belonging to an existing successful connection to pass through, while the fourth command ensures that the default policy for the FORWARD chain is dropped apart from the 3 rules stated above.



```
[10/25/25]seed@1006859:~/.../Labsetup$ dockps
75cd97614870  host3-192.168.60.7
88d231ef2b8d  seed-router
4b8bc4118224  host2-192.168.60.6
50db189db7ad  hostA-10.9.0.5
cd434e07639b  host1-192.168.60.5
[10/25/25]seed@1006859:~/.../Labsetup$ docksh 50
root@50db189db7ad:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
cd434e07639b login:
```

```
[10/25/25]seed@1006859:~/.../Labsetup$ docksh 50
root@50db189db7ad:/# telnet 192.168.60.6
Trying 192.168.60.6...
telnet: Unable to connect to remote host: Connection timed out
root@50db189db7ad:/#
```

```
[10/25/25]seed@1006859:~/.../Labsetup$ docksh cd
root@cd434e07639b:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
50db189db7ad login:
```

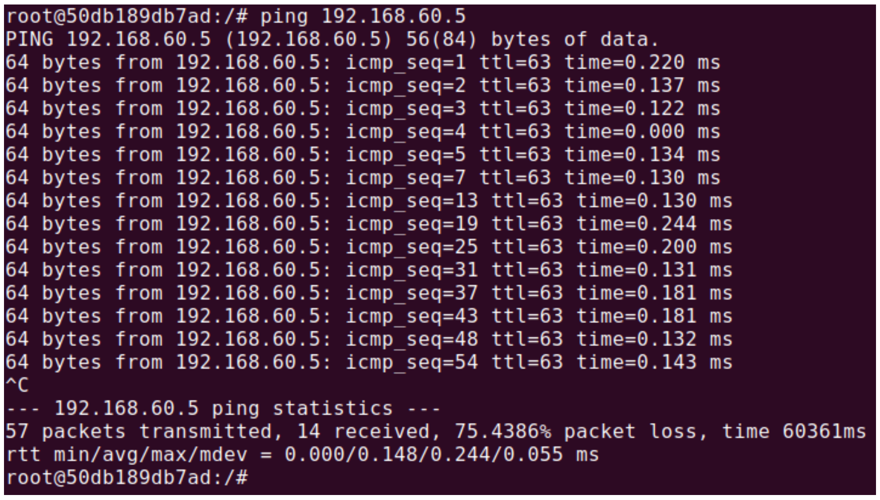
```
[10/25/25]seed@1006859:~/.../Labsetup$ docksh 4b
root@4b8bc4118224:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
50db189db7ad login:
```

The top-left terminal tries to telnet from external hostA to internal host1, which is successful (from Task 2.C). The top-right terminal tries to telnet from external hostA to other internal host2, which fails. The bottom-left terminal tries to telnet from internal host1 to external hostA, which is successful. The bottom-right terminal tries to telnet from internal host2 to external hostA, which is also successful.

Without the connection tracking mechanism, we will have to explicitly state the ports and details that we are going to allow, e.g. we need to explicitly state that a rule where packets from port 80/443 from internal hosts are allowed to be forwarded to the external interface.
With connection tracking:
Advantage: Do not need to predict which ports are being used, so it will work well with dynamic ports, like DNS.
Disadvantage: The kernel must maintain the connection table, which can result in large overhead when there are many connections.
Without connection tracking:
Advantage: There is no overhead for maintaining the connection table and tracking of states, so it will be fast.
Disadvantage: All valid traffic patterns need to be defined manually, so it is difficult to handle dynamic port usage like DNS.

Task 4: Limiting Network Traffic

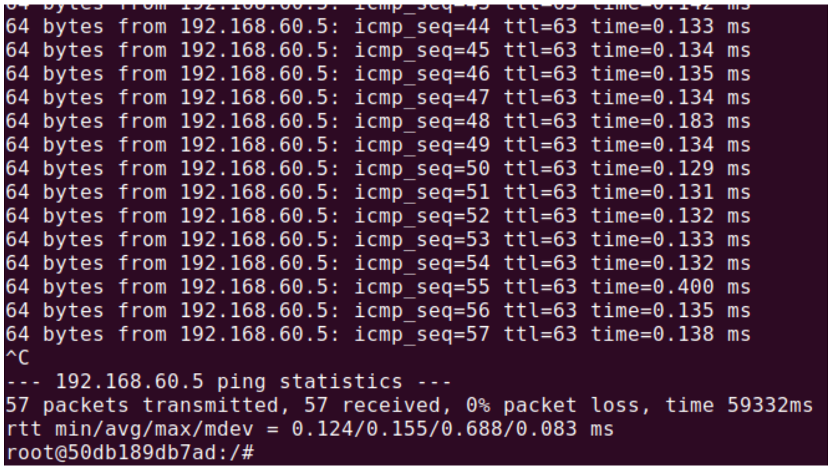
After running both rules, pinging 192.168.60.5 from 10.9.0.5 obtains the following results:



```
root@50db189db7ad:/# 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=0.220 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.137 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.122 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.000 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.130 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.130 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.244 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.200 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.131 ms
64 bytes from 192.168.60.5: icmp_seq=37 ttl=63 time=0.181 ms
64 bytes from 192.168.60.5: icmp_seq=43 ttl=63 time=0.181 ms
64 bytes from 192.168.60.5: icmp_seq=48 ttl=63 time=0.132 ms
64 bytes from 192.168.60.5: icmp_seq=54 ttl=63 time=0.143 ms
^C
--- 192.168.60.5 ping statistics ---
57 packets transmitted, 14 received, 75.4386% packet loss, time 60361ms
rtt min/avg/max/mdev = 0.000/0.148/0.244/0.055 ms
root@50db189db7ad:/#
```

We can see that the longest consecutive ICMP replies received were 5 (icmp_seq 1 to 5), and many packets were lost in the meantime. In about a span of a minute, only 14 packets was allowed by the router to be forwarded from 10.9.0.5 to 192.168.60.5.

Without running the second rule, pinging 192.168.60.5 from 10.9.0.5 obtains the following results:



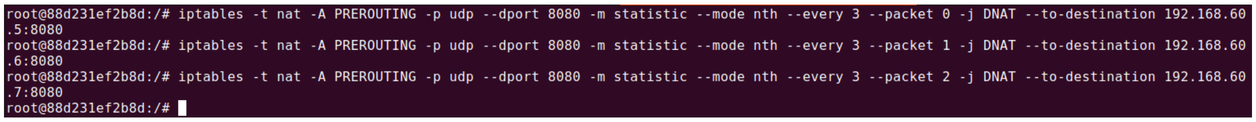
```
64 bytes from 192.168.60.5: icmp_seq=44 ttl=63 time=0.133 ms
64 bytes from 192.168.60.5: icmp_seq=45 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=46 ttl=63 time=0.135 ms
64 bytes from 192.168.60.5: icmp_seq=47 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=48 ttl=63 time=0.183 ms
64 bytes from 192.168.60.5: icmp_seq=49 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=50 ttl=63 time=0.129 ms
64 bytes from 192.168.60.5: icmp_seq=51 ttl=63 time=0.131 ms
64 bytes from 192.168.60.5: icmp_seq=52 ttl=63 time=0.132 ms
64 bytes from 192.168.60.5: icmp_seq=53 ttl=63 time=0.133 ms
64 bytes from 192.168.60.5: icmp_seq=54 ttl=63 time=0.132 ms
64 bytes from 192.168.60.5: icmp_seq=55 ttl=63 time=0.400 ms
64 bytes from 192.168.60.5: icmp_seq=56 ttl=63 time=0.135 ms
64 bytes from 192.168.60.5: icmp_seq=57 ttl=63 time=0.138 ms
^C
--- 192.168.60.5 ping statistics ---
57 packets transmitted, 57 received, 0% packet loss, time 59332ms
rtt min/avg/max/mdev = 0.124/0.155/0.688/0.083 ms
root@50db189db7ad:/#
```

We can see that all ICMP packets were forwarded from 10.9.0.5 to 192.168.60.5 successfully. The second rule is needed to limit network traffic because the default policy for FORWARD is set to ACCEPT. Without the second rule, ICMP packets that do not satisfy the first rule will just be accepted by default. The second rule ensures that the ICMP packets that do not satisfy the first rule (limit) will be dropped by the router, which helps in limiting the network traffic.

Task 5: Load Balancing

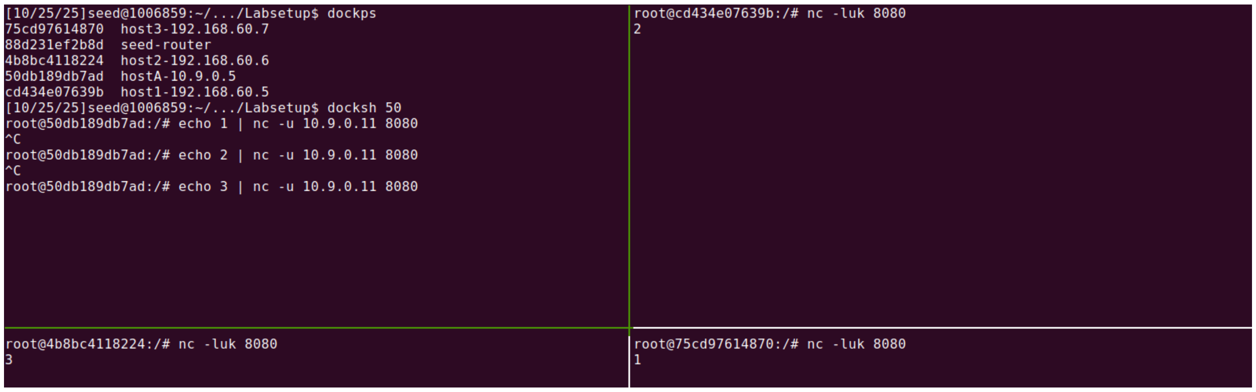
nth mode (round-robin)

First, we run “nc -luk 8080” on all 3 hosts, 192.168.60.5, 192.168.60.6, and 192.168.60.7. Then, we set up the rules as follows:



```
root@88d231ef2b8d:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 0 -j DNAT --to-destination 192.168.60
.5:8080
root@88d231ef2b8d:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 1 -j DNAT --to-destination 192.168.60
.6:8080
root@88d231ef2b8d:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 2 -j DNAT --to-destination 192.168.60
.7:8080
root@88d231ef2b8d:/# 
```

For every 3 packets, the first packet will be sent to 192.168.60.5:8080, while the second packet will be sent to 192.168.60.6:8080, and the last packet will be sent to 192.168.60.7:8080.



```
[10/25/25]seed@1006859:~/.../Labsetup$ dockps
75cd97614870  host3-192.168.60.7
88d231ef2b8d  seed-router
4b8bc4118224  host2-192.168.60.6
50db189db7ad  hostA-10.9.0.5
cd434e07639b  host1-192.168.60.5
[10/25/25]seed@1006859:~/.../Labsetup$ docksh 50
root@50db189db7ad:/# echo 1 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 2 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 3 | nc -u 10.9.0.11 8080
```

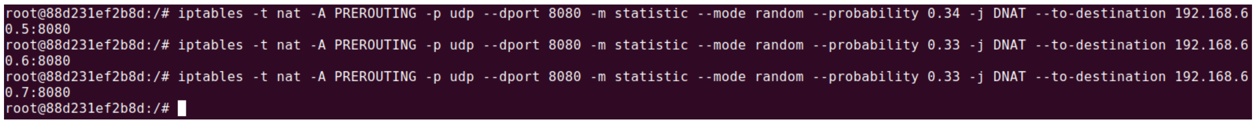
```
root@cd434e07639b:/# nc -luk 8080
2
```

```
root@4b8bc4118224:/# nc -luk 8080
3
```

```
root@75cd97614870:/# nc -luk 8080
1
```

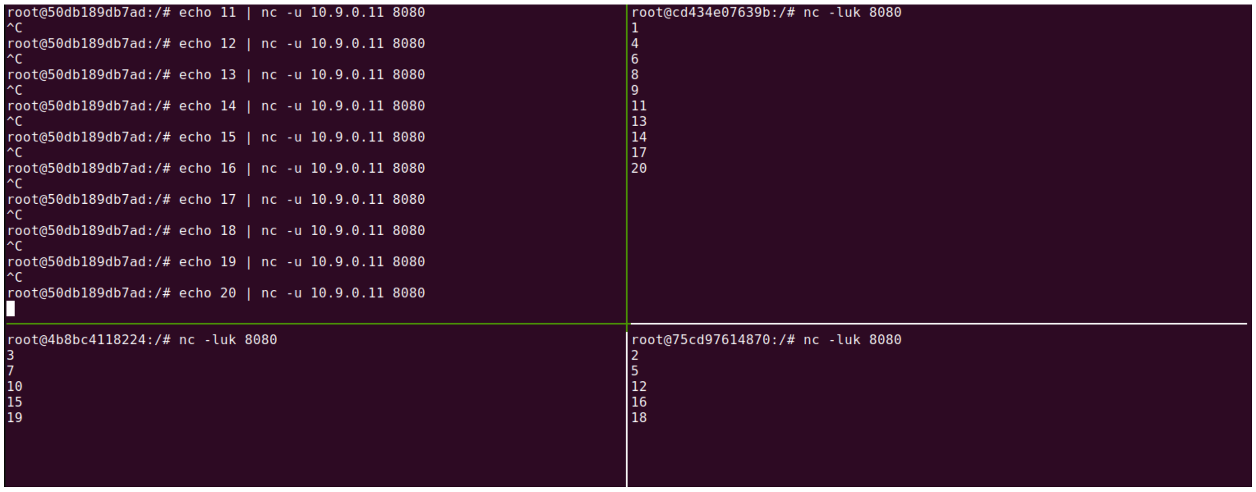
We can see that the 3 packets are each being sent to the 3 different internal servers individually and each internal server gets an equal number of packets.

random mode



```
root@88d231ef2b8d:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.34 -j DNAT --to-destination 192.168.6
0.5:8080
root@88d231ef2b8d:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.33 -j DNAT --to-destination 192.168.6
0.6:8080
root@88d231ef2b8d:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.33 -j DNAT --to-destination 192.168.6
0.7:8080
root@88d231ef2b8d:/# 
```

With the rules above, every packet has a probability of 0.34 to be delivered to 192.168.60.5:8080, and a probability of 0.33 to be delivered to 192.168.60.6:8080 and 192.168.60.7:8080. The sum of the 3 probabilities should be equal to 1.



```
root@50db189db7ad:/# echo 11 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 12 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 13 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 14 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 15 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 16 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 17 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 18 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 19 | nc -u 10.9.0.11 8080
^C
root@50db189db7ad:/# echo 20 | nc -u 10.9.0.11 8080
```

```
root@cd434e07639b:/# nc -luk 8080
1
4
6
8
9
11
13
14
17
20
```

```
root@4b8bc4118224:/# nc -luk 8080
3
7
10
15
19
```

```
root@75cd97614870:/# nc -luk 8080
2
5
12
16
18
```

From the results obtained, we can see that there is a random probability that each internal server gets the UDP packet, compared to nth mode, where the UDP packet is sent in a round robin manner.