CS6903-Network Security

Assignment 4: Firewall using Raw Socket

- Sai Balaram(ES18BTECH11011)
- Sai Mahesh(CS18BTECH11001)
- Satwik Reddy(CS18BTECH11014)
- Srikar Perugu(CS18BTECH11034)

Task 1:

We initially configured the Host1, firewall and Host2 as shown in the screenshot below.

Host1:

```
       valid_lft forever preferred_lft forever
3: ens1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 52:54:00:43:4c:42 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global ens1
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe43:4c42/64 scope link
       valid_lft forever preferred_lft forever
balaram@balaram6712:~$
```

Host2:

```
3: ens1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
 group default qlen 1000
    link/ether 52:54:00:2e:4e:cb brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.253/24 brd 10.0.0.255 scope global ens1
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe2e:4ecb/64 scope link
       valid_lft forever preferred_lft forever
```

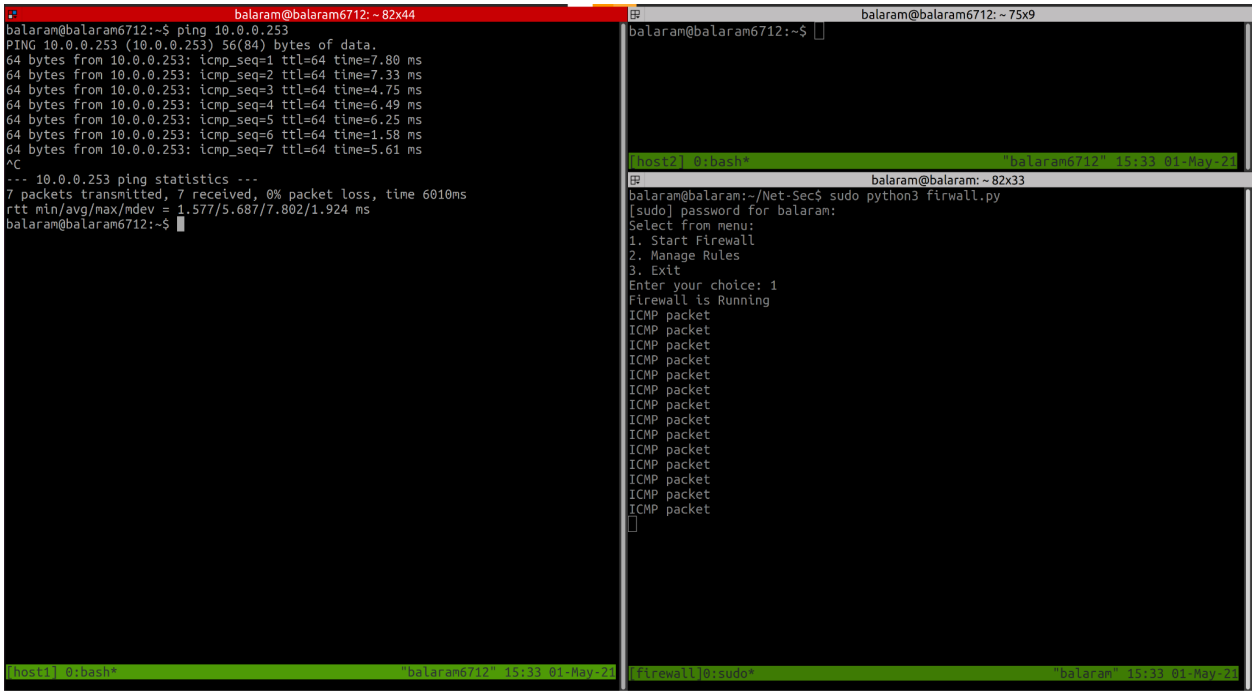
Firewall:

```
       valid_lft forever preferred_lft forever
2: ens1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 52:54:00:3b:90:ce brd ff:ff:ff:ff:ff:ff
    inet6 fe80::5054:ff:fe3b:90ce/64 scope link
       valid_lft forever preferred_lft forever
3: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP grou
p default qlen 1000
    link/ether 52:54:00:dd:97:0f brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.176/24 brd 192.168.122.255 scope global dynamic enp1s0
       valid_lft 2254sec preferred_lft 2254sec
    inet6 fe80::5054:ff:fedd:970f/64 scope link
       valid_lft forever preferred_lft forever
4: ens2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 52:54:00:c0:e2:da brd ff:ff:ff:ff:ff:ff
    inet6 fe80::5054:ff:fec0:e2da/64 scope link
       valid_lft forever preferred_lft forever
```

We have used a router to act as our firewall and the traffic from Host1 will be listened by the ens1 interface of the firewall and then transfers the traffic to Host2 via ens2 interface as implemented in the **firewall.py** using raw socket. We have used separate threads to listen from each interface, validate and to perform required action.

Task 2:

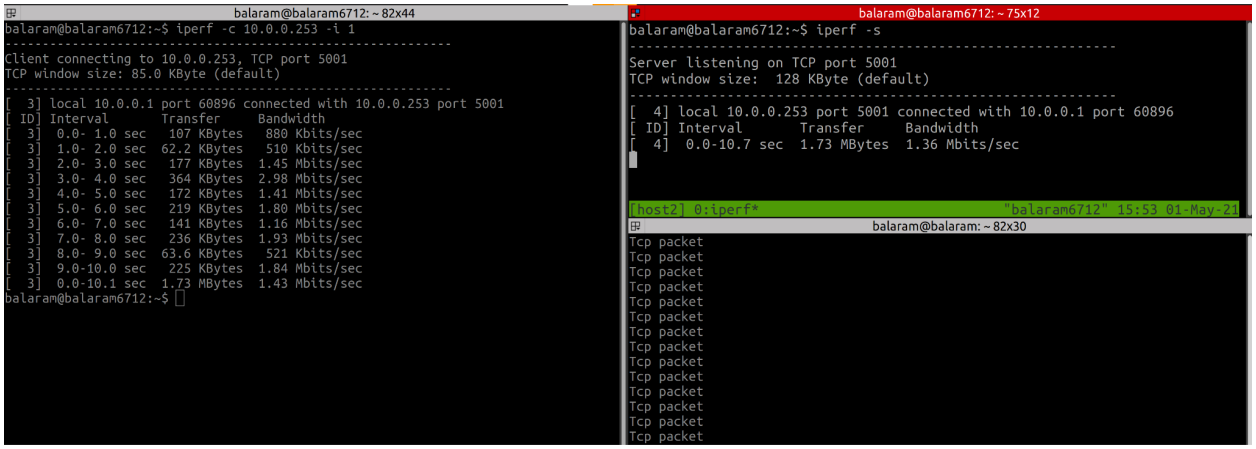
We have analysed each packet until the Layer 4 (MAC, IPv4 IPv6, ICMP for IPv4/v6, TCP/UDP) and maintained in a local dictionary. We have prompted the user for the menu involved to manage the ruleset for the firewall which will be maintained in *rules.json* file. After analysing each packet we check whether this packet is to be transferred forward or to be dropped by loading the elements maintained in the json file and checking its validity in the *validation()* function. In this validation we have checked whether the present packet source IP or destination IP or many other fields are matching with any of their respective fields maintained in the rules.json file and if matched we return 0 instructing the thread to drop the packet. We have used the try and finally block to ensure that the keyErrors doesn't happen when accessing the dictionary. Here is a screenshot of working of the firewall:



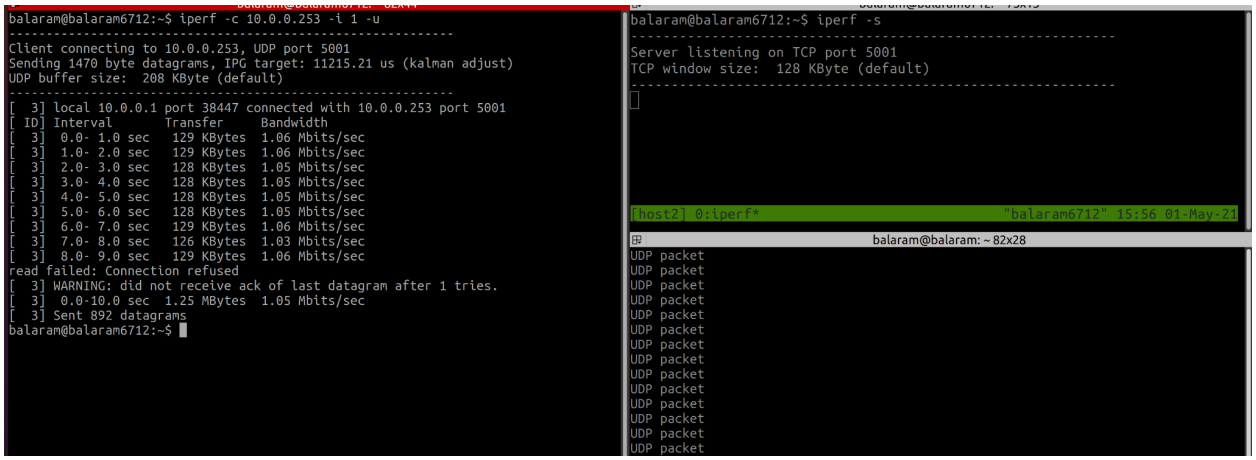
Here when using ping from Host1 to Host2 we can see that it is successful and the firewall can detect that the sent packet is an ICMP packet(Successfully traversed until Layer4).

Similarly we have used the iperf command to check whether its working for TCP/UDP packets and we have successfully implemented its part.

Here is a screenshot of working implementation for TCP Packet



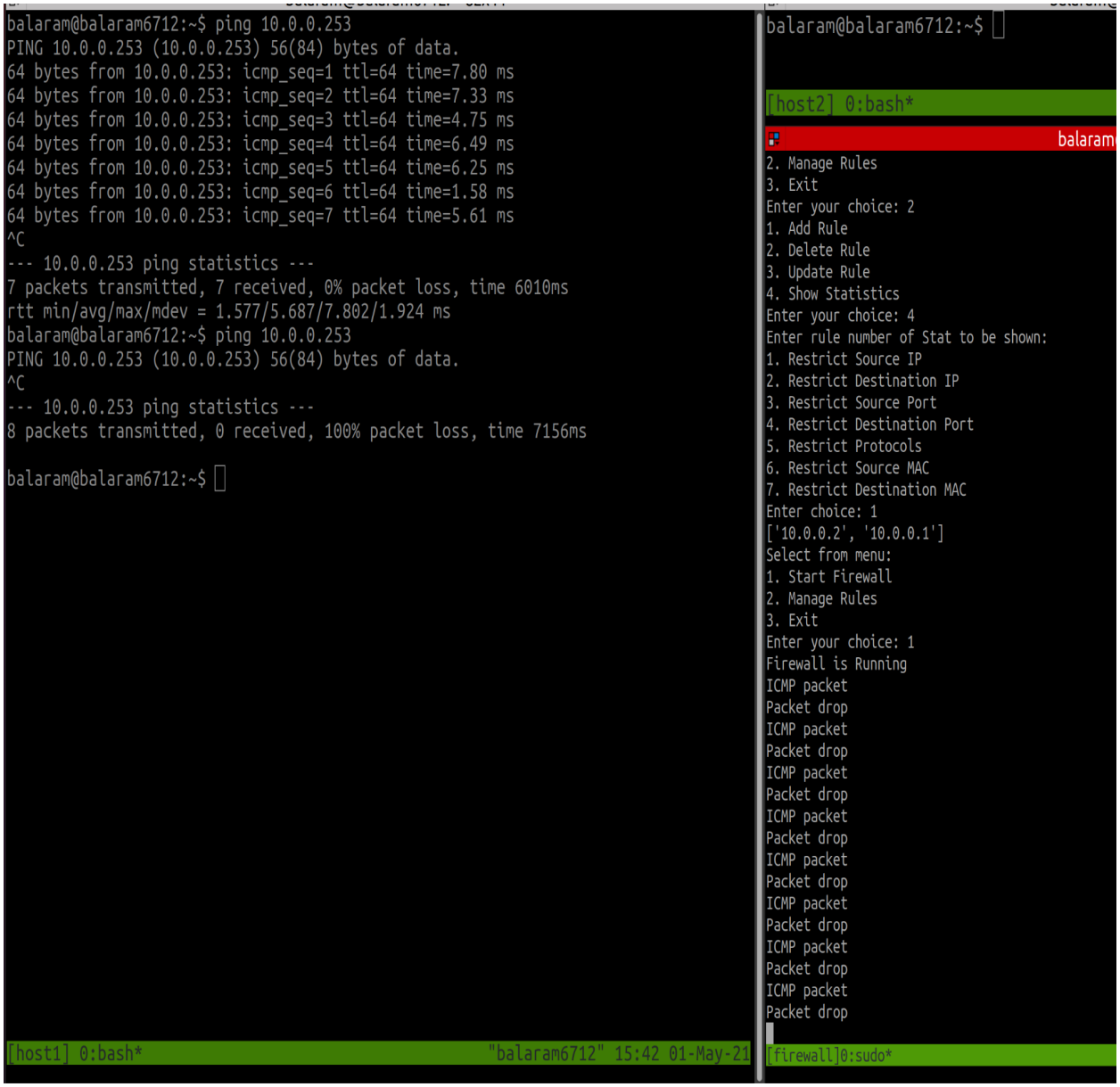
Here is a screenshot of working implementation for UDP Packet



Here is a screenshot depicting how to manage rules to the firewall. Firstly we are adding a rule to block the IP address - ‘10.0.0.1’ and showing statistics of what IP addresses are blocked till now.

```
1. Add Rule
2. Delete Rule
3. Update Rule
4. Show Statistics
Enter your choice: 1
1. Restrict Source IP
2. Restrict Destination IP
3. Restrict Source Port
4. Restrict Destination Port
5. Restrict Protocols
6. Restrict Source MAC
7. Restrict Destination MAC
Enter choice: 1
Enter ip: 10.0.0.1
Select from menu:
1. Start Firewall
2. Manage Rules
3. Exit
Enter your choice: 2
1. Add Rule
2. Delete Rule
3. Update Rule
4. Show Statistics
Enter your choice: 4
Enter rule number of Stat to be shown:
1. Restrict Source IP
2. Restrict Destination IP
3. Restrict Source Port
4. Restrict Destination Port
5. Restrict Protocols
6. Restrict Source MAC
7. Restrict Destination MAC
Enter choice: 1
['10.0.0.2', '10.0.0.1']
Select from menu:
```

Here is a screenshot depicting that the packets transferred from the IP address 10.0.0.1 (Blocked by adding a rule to the firewall) have been dropped by the firewall(Successful in validating the ruleset).

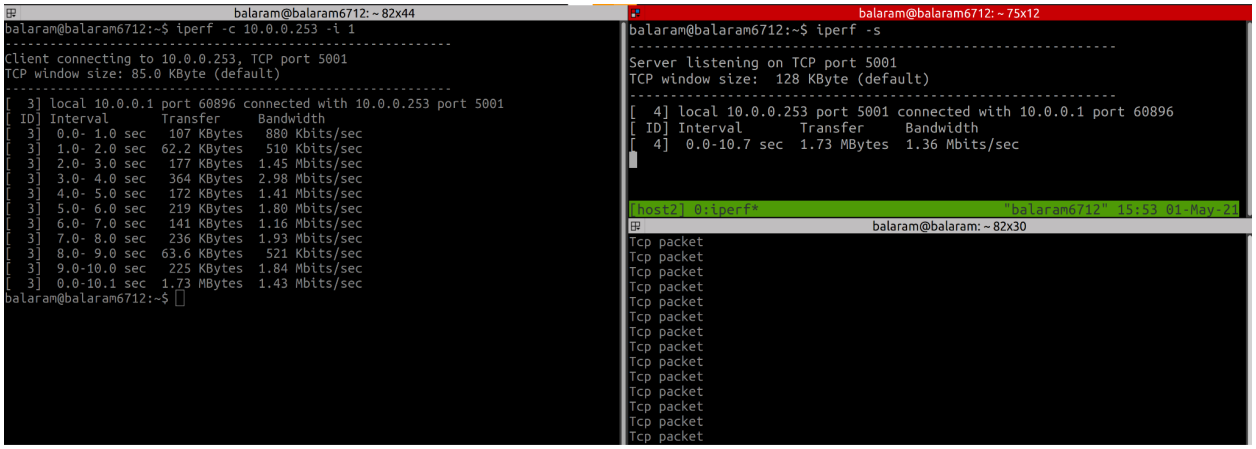


Here is a screenshot depicting how to remove a IP address from the block list

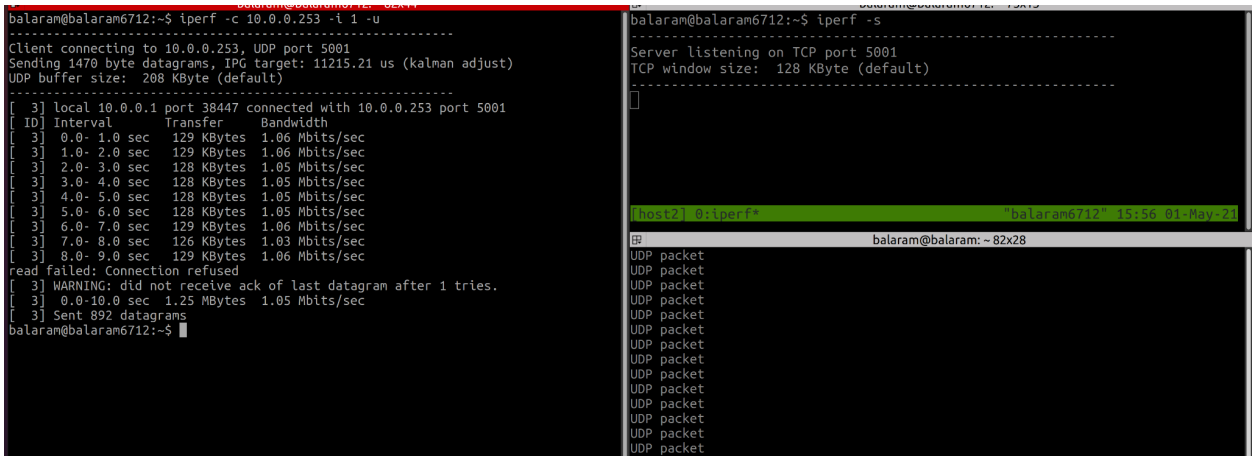
```
1. Add Rule
2. Delete Rule
3. Update Rule
4. Show Statistics
Enter your choice: 2
1. Restrict Source IP
2. Restrict Destination IP
3. Restrict Source Port
4. Restrict Destination Port
5. Restrict Protocols
6. Restrict Source MAC
7. Restrict Destination MAC
Enter choice: 1
Enter ip: 10.0.0.1
Select from menu:
1. Start Firewall
2. Manage Rules
3. Exit
Enter your choice: 2
1. Add Rule
2. Delete Rule
3. Update Rule
4. Show Statistics
Enter your choice: 4
Enter rule number of Stat to be shown:
1. Restrict Source IP
2. Restrict Destination IP
3. Restrict Source Port
4. Restrict Destination Port
5. Restrict Protocols
6. Restrict Source MAC
7. Restrict Destination MAC
Enter choice: 1
['10.0.0.2']
Select from menu:
1. Start Firewall
2. Manage Rules
3. Exit
Enter your choice:
[firewall]0:sudo*
```

Task 3:

Here is a screenshot depicting the performance of the TCP packets when all the packets are sent.



Here is a screenshot depicting the performance of the UDP packets when all the packets are sent.



From these we can say that our implementation can handle upto 832 Packets in 10 sec = 83 PPS

As the number of rules are increasing, the iterations required for each packet during validation with the ruleset will increase and thereby decreasing the overall performance of our firewall.

Task 4a:

To improve the performance of the firewall while validating for each packet we can use multiple threads to run the multiple for loop(each for loop to check each list of a field) which will therefore reduce the time for validating for each packet and therefore decreasing the overall time required for transfer and hence increasing the performance. The overall improvement in performance by using this idea is around **2.3x - 3x** after considering the thread switching too.