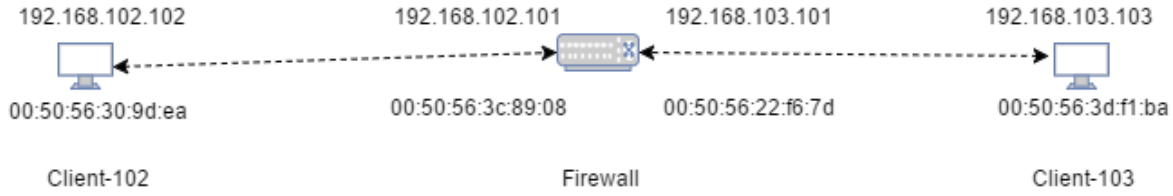
Firewall Assignment

Group Members:
Mrinal Aich - cs16mtech11009
Tulja Vamshi Kiran – cs16mtech11020

Network Architecture:

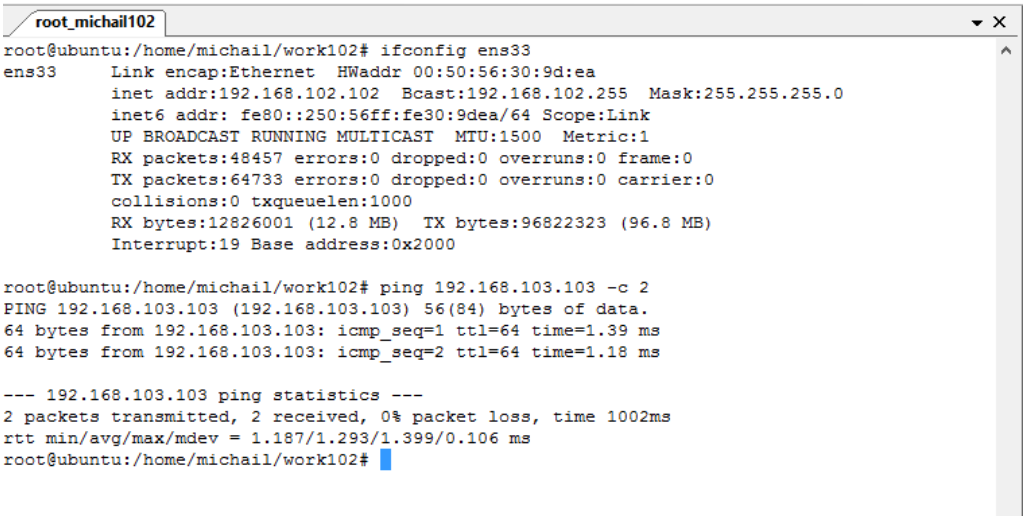
The network consists of two Clients in different subnets 192.168.102.0/24 and 192.168.103.0/24. They are connected to the Firewall with their default gateway.



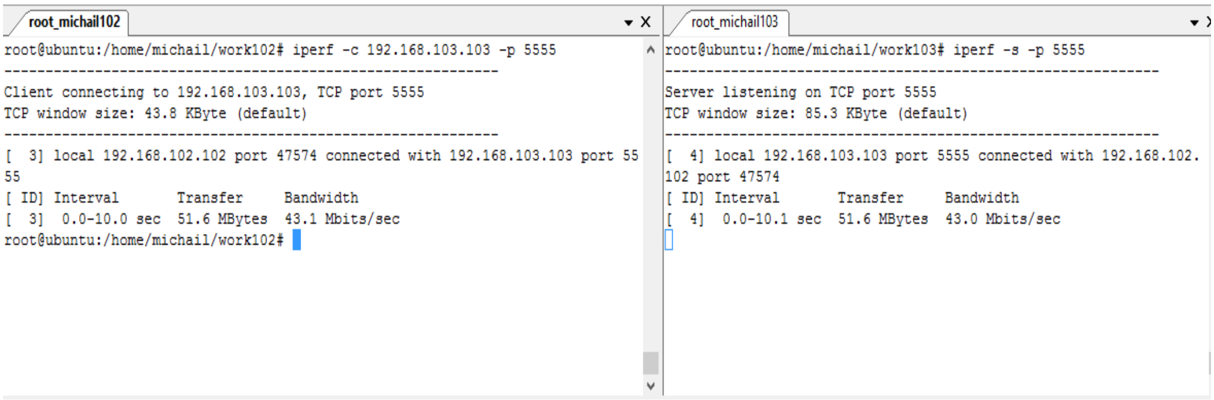
Task 1:

The forwarding logic is handled entirely by the Firewall program. Hence, IP Forwarding is not allowed at the Firewall's kernel.

Case 1: Layer-3 forwarding using ICMP Protocol



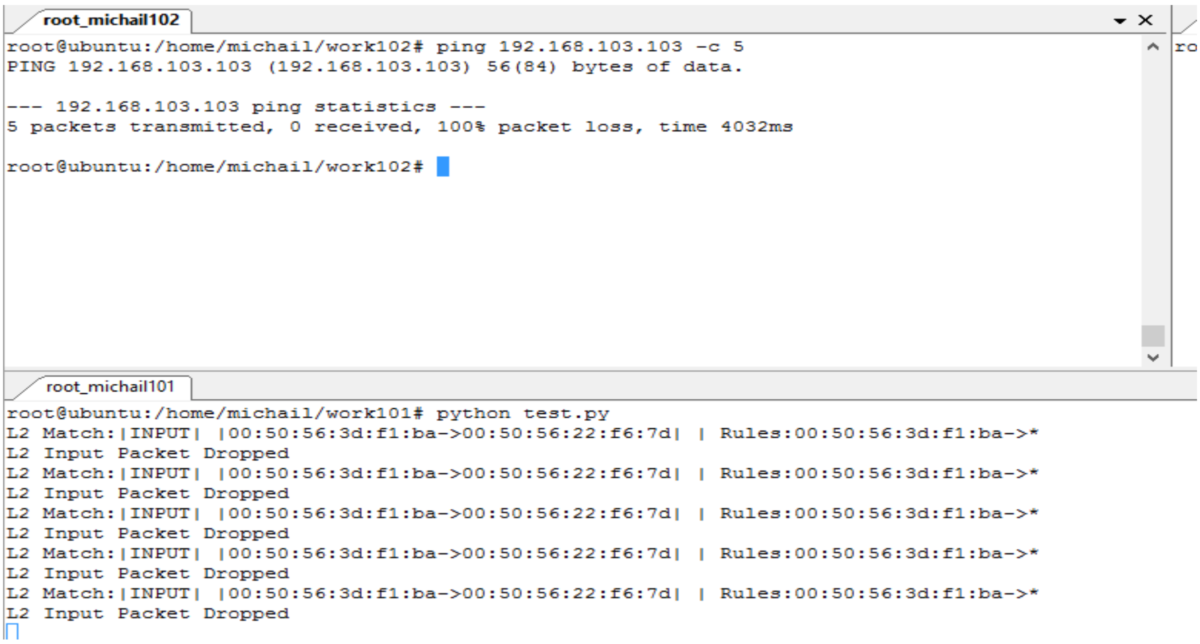
Case 2: Layer-4 forwarding using IPerf Application
‘Host103’ is running an IPerf server at port 5555, and ‘Host102’ acts as the IPerf client.
The measured bandwidth is ***45.1 Mbps***.



Firewall Block Scenarios

Case 3: Layer-2 packets filtered by the Firewall.

The rule-set used:
On INPUT chain, L2 Packets with MAC Address: “00:50:56:3d:f1:ba” DROP
i.e. All L2 packets coming from ‘Host103’ gets dropped.



All the ARP Reply packets sent by ‘Host102’ are dropped by the INPUT chain.

Case 4: Layer-3 packets filtered by the Firewall.

The rule-set used:
On OUTPUT chain, L3 Packets with IP Address: “192.168.102.102” DROP

i.e. All L3 packets going towards 'Host102' gets dropped.

```
root_michail102
root@ubuntu:/home/michail/work102# ping 192.168.103.103 -c 5
PING 192.168.103.103 (192.168.103.103) 56(84) bytes of data.

--- 192.168.103.103 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4032ms

root@ubuntu:/home/michail/work102#
```

```
root_michail101
root@ubuntu:/home/michail/work101# python test.py
L3 Match:|OUTPUT ICMP| |192.168.102.101->192.168.102.102| | Rules:*->192.168.102.102
L3 Output Packet Dropped
L3 Match:|OUTPUT ICMP| |192.168.102.101->192.168.102.102| | Rules:*->192.168.102.102
L3 Output Packet Dropped
L3 Match:|OUTPUT ICMP| |192.168.102.101->192.168.102.102| | Rules:*->192.168.102.102
L3 Output Packet Dropped
L3 Match:|OUTPUT ICMP| |192.168.102.101->192.168.102.102| | Rules:*->192.168.102.102
L3 Output Packet Dropped
L3 Match:|OUTPUT ICMP| |192.168.102.101->192.168.102.102| | Rules:*->192.168.102.102
L3 Output Packet Dropped
```

All ICMP Reply packets towards 'Host102' gets dropped by the OUTPUT chain.

Case 5: Layer-4 packets filtered by the Firewall.

The rule-set used:

On INPUT chain, L4 TCP Packets with Source Port: "5555" DROP

i.e. All L4 TCP packets sent by Source Port 5555 gets dropped.

```
root_michail102
root@ubuntu:/home/michail/work102# iperf -c 192.168.103.103 -p 5555
```

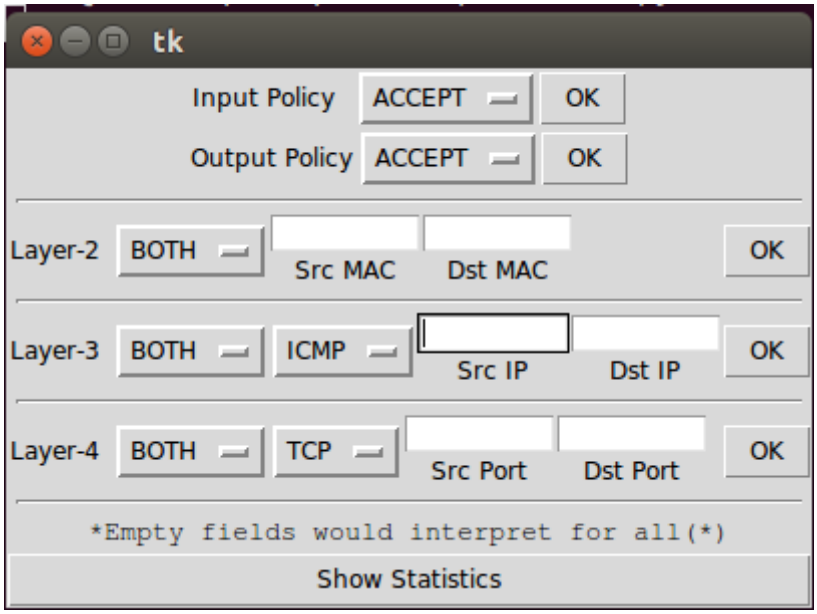
```
root_michail103
root@ubuntu:/home/michail/work103# iperf -s -p 5555
------------------------------------------------------------
Server listening on TCP port 5555
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

```
root_michail101
root@ubuntu:/home/michail/work101# python test.py
L4 Input Packet Dropped
L4 Input Packet Dropped
L4 Input Packet Dropped
L4 Input Packet Dropped
L4 Input Packet Dropped
L4 Input Packet Dropped
L4 Input Packet Dropped
```

All TCP Syn packets sent by 'Host102' with Source Port =5555 gets dropped by the INPUT chain.

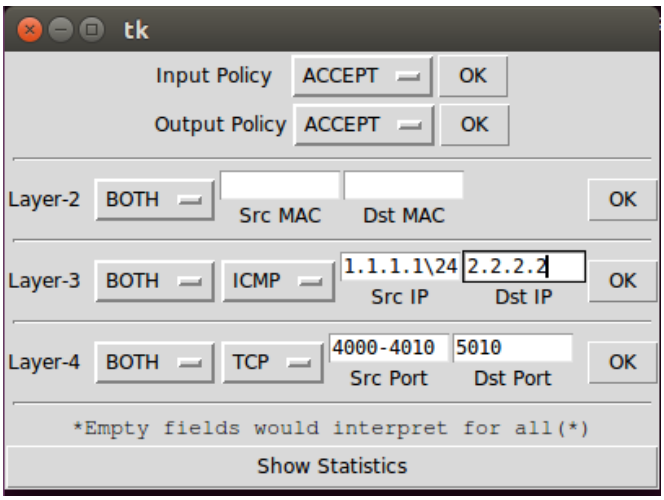
Task 2: Graphical User Interface of the Firewall

The GUI was developed using Tkinter and it looks like



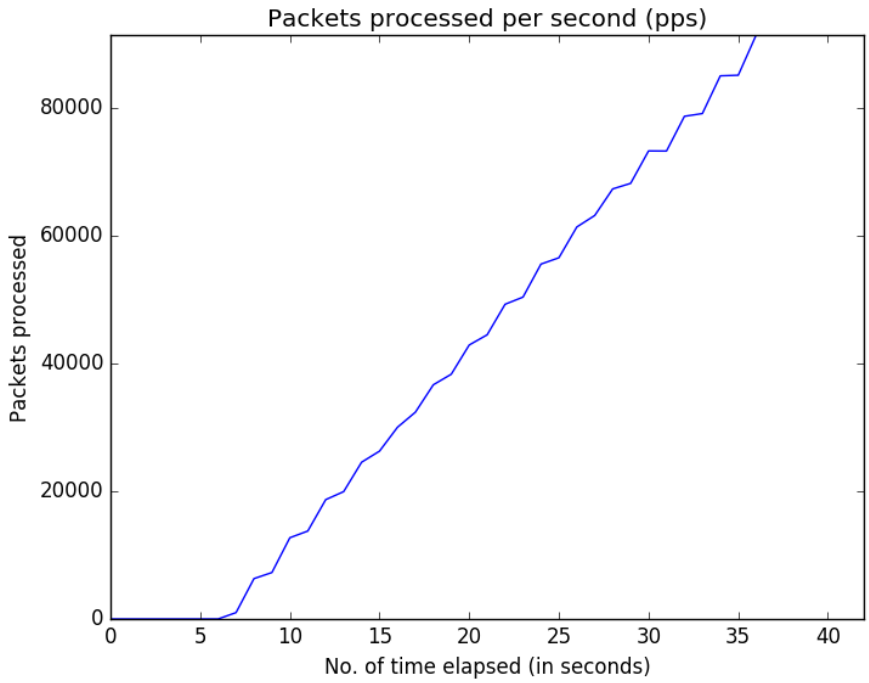
1. INPUT/OUTPUT Chain
   The user can select a default Input/Output Policy as 'Accept' or 'Drop'.
   On selecting a default policy, all the rules on that chain would be reset.

2. 'BOTH' in the Drop down menu signify the rule would be for both Input and Output Chain.
3. If a respective field is empty, the it is interpreted as '*' (for all).
4. Layer-3 supported protocols are 'IPv4' and 'IPv6'.
5. Layer-4 supported protocols are 'TCP' and 'UDP'.
6. Aggregation of IP prefix and Port-range are also handled.



Eg. Layer-3: All source IP packets from 1.1.1.1\24 to 2.2.2.2 to be dropped.
Layer-4: All packets with source ports from 4000 to 4010 to destination port 5010 to be dropped.

7. ‘Show Statistics’ displays the total number of packets processed by the firewall .It also displays the Packet Counts in terms of Layer2,Layer3 andLayer4 along with the no. of packets that are being dropped by the firewall.
   A graph is plotted with the time (in secs) in X-axis and Packets processed in the Y-axis.



Plot for an IPerf application starts at time (t=6) and finishes at time (t=36). The number of packets processed per second increases gradually with time. The bandwidth achieved was 46.8 Mbps.

Task 3: Performance examination and improvement

(i).Measuring performance

Performance is measured by the *throughput* achieved via the firewall.

A single controlled traffic will be generated by 'Host102' to 'Host103'. The throughput achieved by the hosts will be considered as the throughput of the firewall.

(On an average the throughput for an IPerf application is close to 50 Mbps using our implementation.)

(ii). PPS handled by the firewall -

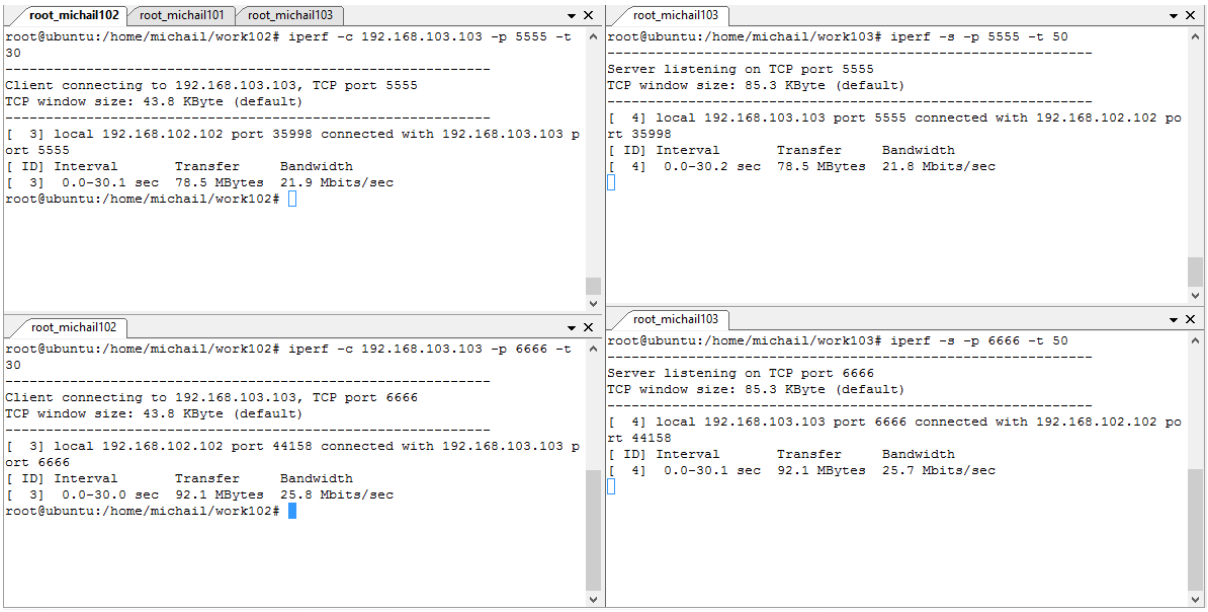
On average the packets processed per second by the firewall is close to 100000 IPerf packets.

(iii). Change of performance if drop or pass the packet -

Considerable change is observed if particular packets are dropped by the firewall.

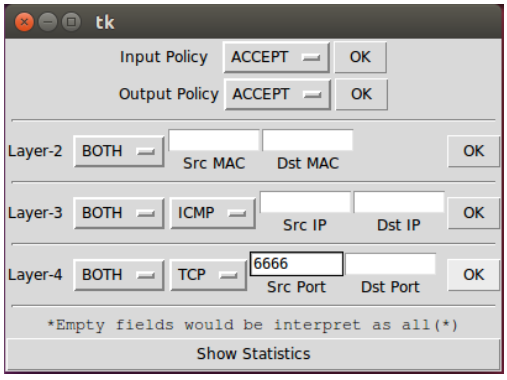
Scenario: Two IPerf servers are running on 'Host103' at ports 5555 and 6666.

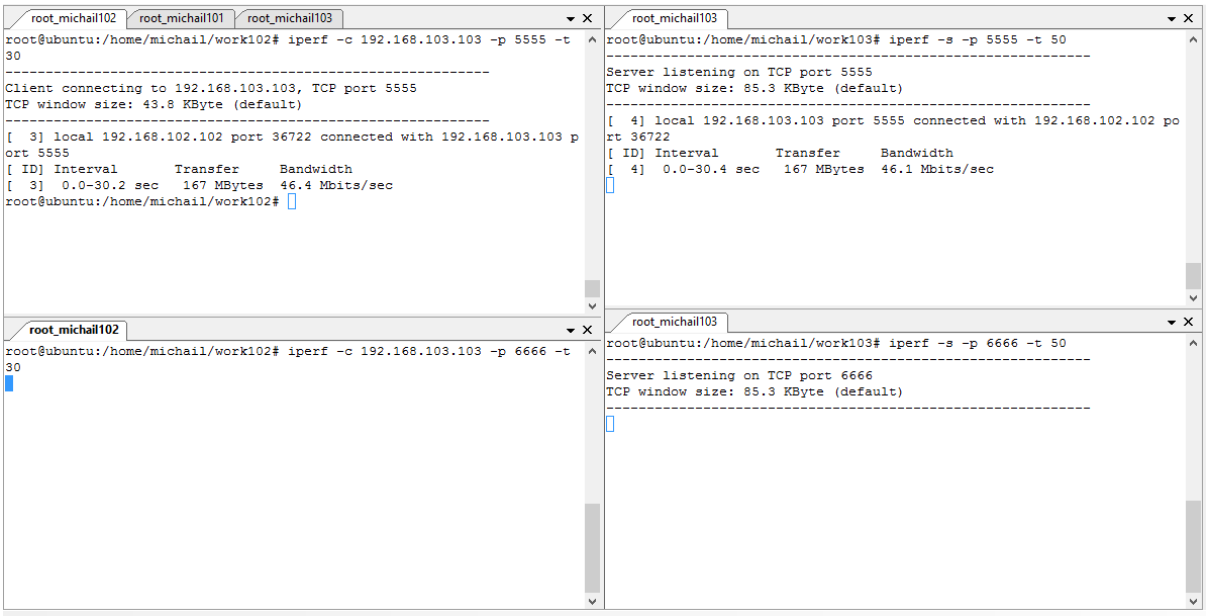
'Host103' runs two IPerf clients and connects to 'Host103'.



Analysis: The two IPerf Clients are getting shared throughput of 21.9 Mbps and 25.0 Mbps.

Scenario: One IPerf flow is blocked by the firewall, say all TCP traffic to port 6666 gets dropped.

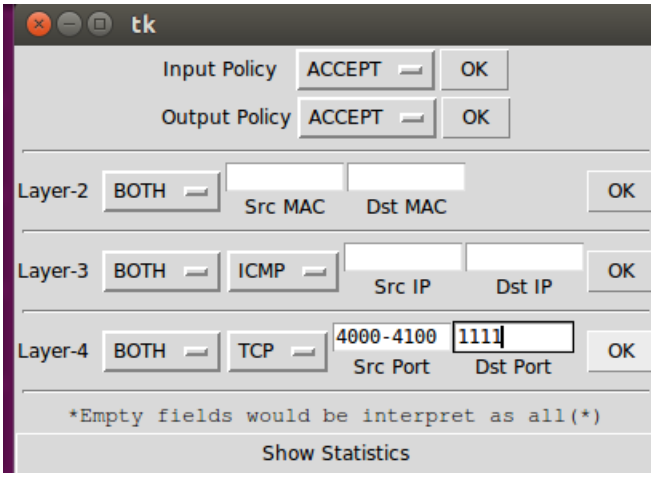


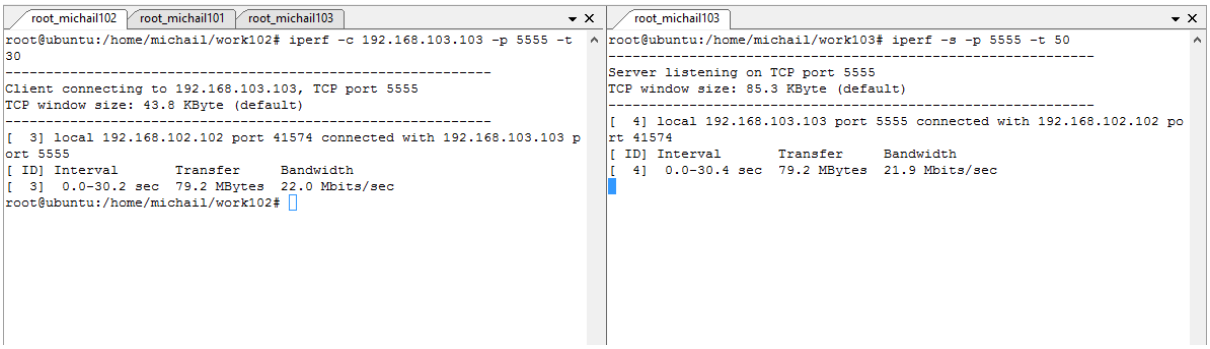


Analysis: On blocking one IPerf flow by the firewall, the throughput of the other flow increases. IPerf packets for port 5555 are successful and for port 6666 are dropped by the firewall. Due to this, the throughput of the valid flow increases to 46.4 Mbps.

(v). Increase the number of matching fields in a filtering rule (MAC address only or Prefix/Port/Protocol combination).

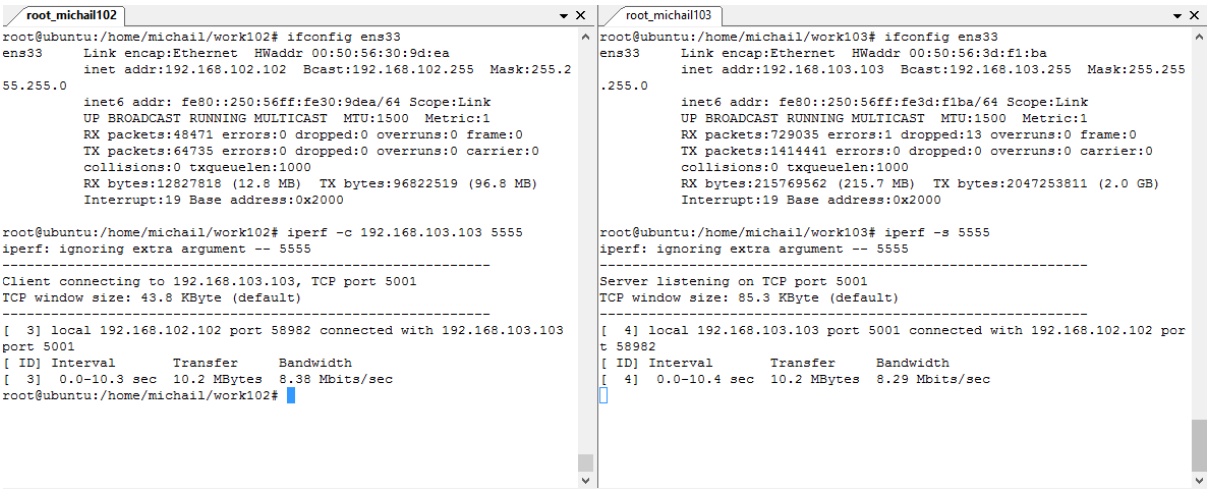
Scenario: Added 200 Rules to drop TCP packets in range Source Ports 4000-4100 and Destination Port:1111. On running the IPerf client-server as before over port 5555.



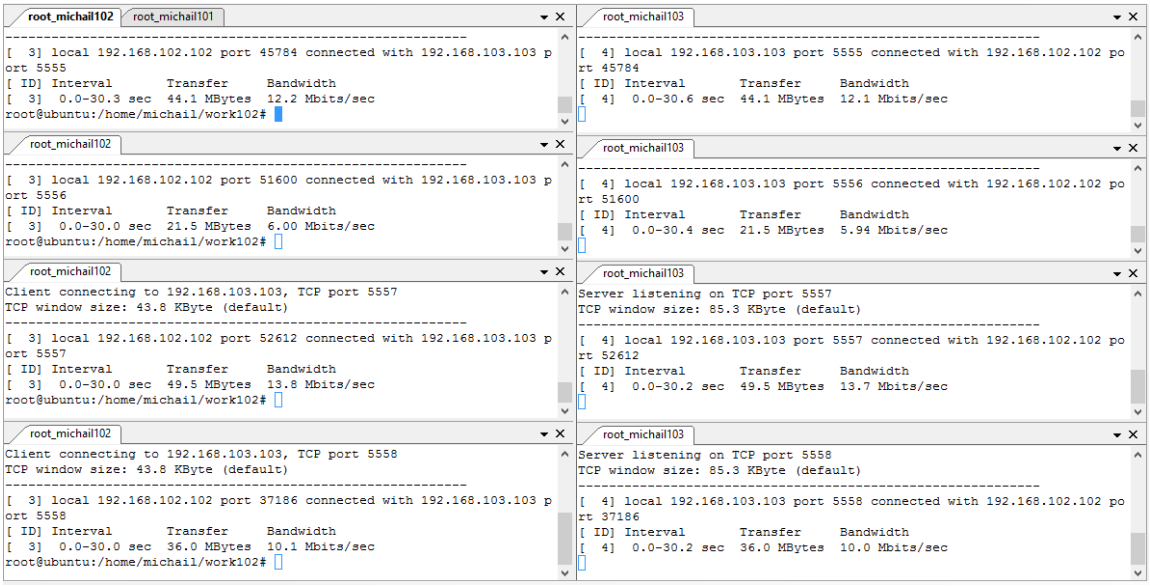


Analysis: The rule matching would be done by the firewall, thus increasing the delay in forwarding the traffic which decreases the throughput from 50 Mbps(avg.) to 22 Mbps.

(vi). Controlled or random traffic, to benchmark the system.
Controlled traffic is generated by a single IPerf application to get achieve the maximum throughput of the firewall. On avg. the value is 50 Mbps.



Random Traffic is generated by running 4 IPerf servers at Host103 and being accessed by Host102. The throughput of all the applications is reduced considerably.



Task 4: Detecting attacks in the network using Firewall

Attack – PORT SCAN

A **port scan attack**, occurs when an attacker sends packets to a victim machine, varying the destination **port**. The attacker can use this to find out what services the victim is running and to get a pretty good idea of its operating system.

Detection:

The detection would be based on a particular ***Pattern*** and its ***frequency***.
Here the pattern is chosen as port numbers and frequency is the unsuccessful connections. Firewall on receiving a TCP SYN packet, RECORD the tuple (SrcIP, Time and Destination Port). Also it maintains, destination IP (DstIP) to match with the returning 'SYN/ACK' or 'RST'.

On receiving 'RST' packet from the destination, the TCP connection is unsuccessful so the firewall would 'Mark' the Source with his IP Address and Time. If successive attempts are made by the same source with different destination port numbers over a pre-defined interval, then the Source would be blocked. Also, the RECORD for this connection is deleted.

On receiving 'SYN/ACK' packet from the destination, the TCP connection is successful so the RECORD would be deleted.

Simulation:

'Host102' sends TCP SYN packets over 10 ports to destination 'Host103'. On detecting 5 Port RST for the same Source, Firewall blocks the 'Host102'.
e.g. IPerf SYN packets are sent from 'Host102' to 'Host103' with ports 5555-5564.The response are all 'RST' packets as no IPerf Server is running.
Note: Emulated using shell script 'portScanAttack.sh'.

Evaluation: