Capstone Project Report

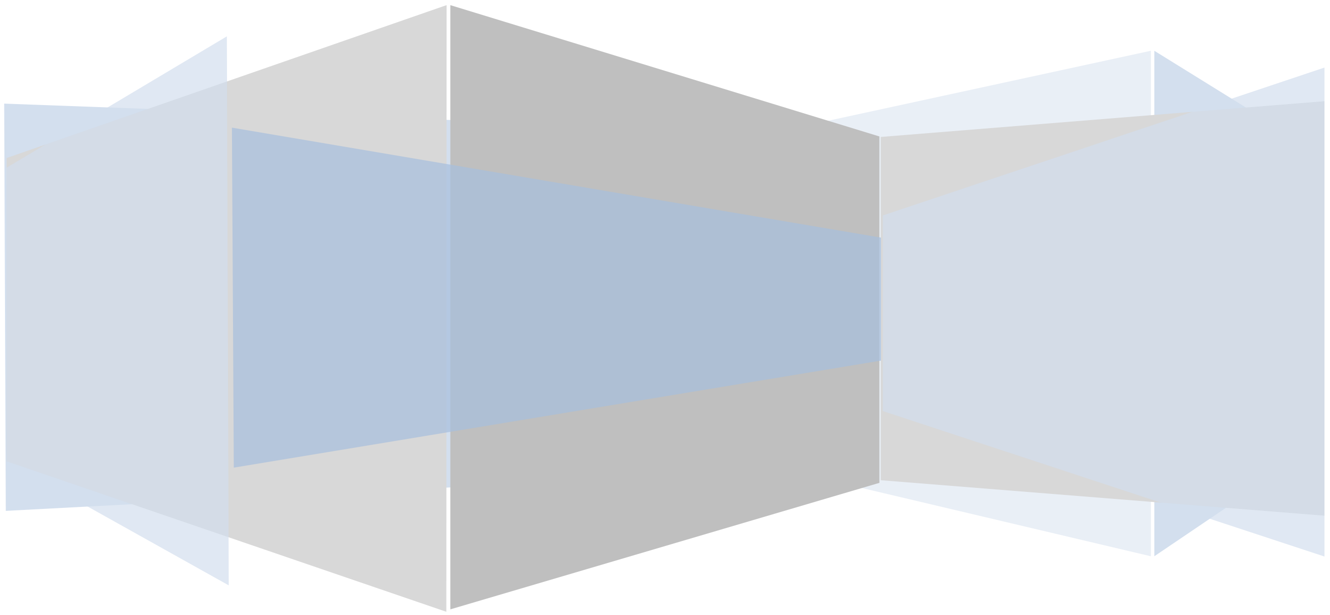
# Linux Traffic Control in Virtual Environment

Master of Science in Internetworking

University of Alberta,
Edmonton, Alberta

Submitted by:
Kanwar Pal Singh

Project Advisor:
Dr. Mike MacGregor



## ACKNOWLEDGEMENT

I take immense pleasure in thanking Dr. Mike MacGregor, Professor and Chair  in Department of Computing Science, University of Alberta, for giving me an opportunity to carry out this project work. I am thankful to him for his guidance and constant supervision, as well as for providing necessary information regarding the project .

A special thanks to Mr. Shahnawaz Mir for all the help extended by him in the  MINT lab.

## ABSTRACT

Virtual environment is an environment in which the old “one server, one application” model is removed and multiple machines are run on each physical machine.

Traffic control is the name given to the sets of queueing systems and mechanisms by which packets are received and transmitted on a router. This includes deciding which packets to accept at what rate on the input of an interface and determining which packets to transmit in what order at what rate on the output of an interface.

The main purpose of this setup was to create a virtual infrastructure and then using Linux, implement traffic control on different virtual machines and physical machines (Traffic Control utility like HTB). It involved the study of throughput and latency as a function of round-trip time and queue discipline.

The study involved implementation of a virtual network by first installing the ESXi server and then installed various OS on that server. These Virtual machines were connected to other Ubuntu machines that were present in a virtual environment on a desktop. After that physical network was introduced to this virtual network using router/switches and then traffic control was done using Ubuntu PC with two NIC's as a router.

The main tasks for setting up virtual environment were:

- Installation of ESX/ESXi server.
- Installing various Virtual Machines on the server (Including Linux).
- Creating a network of connecting ESXi server to other Virtual machines on laptop using routers and switches.
- Running various applications between the virtual machines and physical machines and implementing Traffic Control.

Traffic control may involve various scenarios like:

- Limit bandwidth to a known rate.
- Limit bandwidth of a particular user.
- Reserve bandwidth for a particular application.
- Prefer latency sensitive traffic.

Iperf tool was used to check bandwidth and throughput of the network for all the experiments done.

## Table of Contents

## 1.Introduction to Virtualisation:

Virtualization is mostly concerned about Server virtualization. There are many other types of virtualization but the most widely used is the Server virtualization. They are:

- Server Virtualization
- Storage Virtualization
- Network Virtualization
- Application Virtualization
- Desktop Virtualization

Server virtualization is a process of consolidating multiple physical servers onto a single physical server. Say we have around 15 physicals servers in a data centre; we can consolidate all of these servers onto a single physical server. Saving lots of resources and will require very less resources to manage one physical server than managing around 15 physical servers.

We will need less Network switches, less power utilization, small ups, less cooling required, less management required etc.

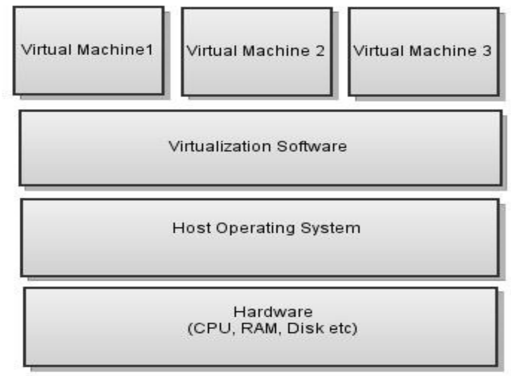
The easiest form of virtualization is Desktop virtualization as we can implement it on our laptop or desktop. We don't need a separate piece of hardware for this; we will be running another OS on our current OS using a software like VMware Workstation. This will allocate specific hardware to the Virtual OS from the existing OS.

Approaches used in virtualization:

- Virtualization management layer approach, also called hosted approach.
- Dedicated virtualization approach.

Virtualization Management Layer approach:

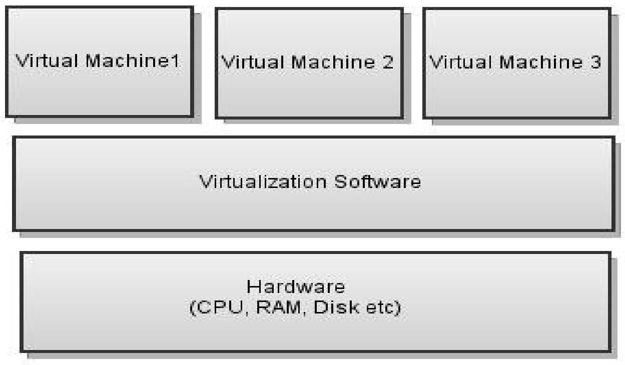
In this we have our desktop or laptop, we have our hardware like CPU, RAM etc. We can install any Operating system on that, and further on that OS we can install an Virtualization software which will help us to install multiple OS's on that single piece of hardware. Our Virtual OS will not be using the hardware directly, but will be interacting with the hardware through the Virtualization host VMware Workstation, the OS installed on the host will called guests. The guest OS's have to go through the Host Operating System to get to the hardware, causing little overhead.



The overhead from Virtualization software can be removed if this software is installed directly over the hardware, which is termed as Dedicated Virtualization approach.

Dedicated virtualization approach:

In this approach Virtualization software like VMware ESX/ESXi are installed directly on the hardware, i.e. it has direct control over the physical hardware as shown in the image below.



Here VMware ESX/ESXi act as the operating system.
The above two types of Virtualization approaches will be used in our scenario, in which we will have multiple guest machine installed on ESXi 4.1 server and multiple guest machines installed in VMware Workstation 7 on a Ubuntu laptop.

We are having one physical server on which ESXi 4.1 is installed and will have Ubuntu Virtual Machines installed on it. Then on a laptop with Ubuntu OS, VMware workstation will be installed and it will have at least 3 Ubuntu machines (one will act as Ubuntu router with 2 NIC's, other two will be acting as clients for the Ubuntu Machine on ESXi) and one Win XP Virtual machine to access the Ubuntu Machine on ESXi through VMware VSphere Client.

The Scenario will be more clear in the experiments performed.

## 2.Introduction to Linux Traffic Control:

Traffic control means controlling the transmission of packets at an interface of an operating system. There is Linux networking code present in each Linux operating system that helps to manage the traffic on the various Ethernet ports on that particular system.

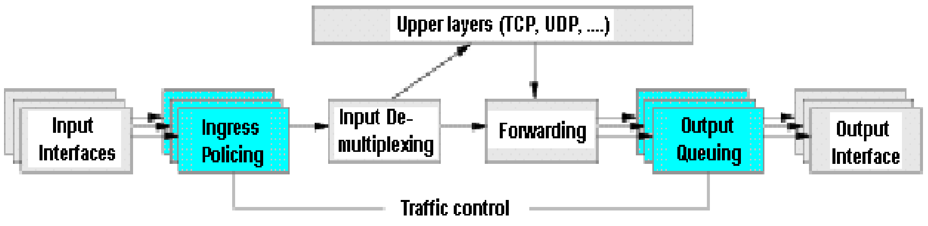
The Linux networking code receives the packets from one interface networking driver and sends it to the required interface networking driver to be forwarded to another node.
When a packet is received at the input interface it is policed over there. This may include discarding the packets if the rate of receiving is too fast for the interface to handle all the incoming packets, then these packets may be routed directly to other interface of the system, if it is acting like a router or they may be transferred to upper layers to do some processing on the packets. These layers may add some data of their own on the packets to transfer them on the network.

If the packets were not sent to the upper layers, then the packets will follow the path from left to right as:

Input interface...Ingress Policing...Input De-multiplexing... Forwarding..Output Queueing...Output Int.

Traffic control can be applied on the packets at Ingress Policing and Output queueing. At Ingress policing the packets can only be discarded if not required or we can put an upper limit to restrict the rate of receiving those packets, but at output queueing packets can be delayed, dropped, queued or prioritized as required.

So in all the next experiments, traffic control will be applied at output queueing.



(Ref: http://opalsoft.net/qos/DS-21.htm)

The main components of traffic control are as follows:

- queue Disciplines
- classes
- filters
- policers

Queue discipline: These are the algorithms that control the way the packets will be sent out on the network through the exit interface. These can be classful or classless.

Classless: these contain no classes and no filters can be attached to them to filter the traffic on basis on IP addresses, port numbers etc. They only define the way traffic is sent out on the network (pfifo, stochastic fairness queueing).

Classful: these contain classes, which provide handles to which filters can be attached to get desired traffic control on basis of port numbers, ip addresses etc.

Class: classes exist inside classful queue discipline; they may have children classes to further classify the traffic. They can have multiple filters to filter traffic.
Classes also help to define what rate should particular type of traffic should follow, further classes may contain children classes which can be used to implement classless queue disciplines like pfifo on the

Filters: Filters are used to decide that which packet should follow which class. Filters can use port numbers, source address or destination address to classify traffic.
Classifiers are used in filters to select packets based on their attributes.

**More explanation about queue disciplines, classes and filters is given along with their implementation.**

## 3. Introduction to Iperf:

Iperf is a free tool available to measure the bandwidth of the network. To measure the bandwidth between two systems we need to install iperf on both the systems. One of them will be acting as an Iperf Server and the other will be Iperf client. By default iperf uses TCP connections.

For making a system as Iperf server we need to give this command: *iperf -s*
This command will make that system a iperf server and will open port number 5001 looking for iperf client, whenever a pc gives the command: *iperf -c <IP add of server>;* it will start looking for iperf server on the network with given ip address. By default it will communicate with that server for 10 seconds and will give us the bandwidth available between the two PC's.

I have used Iperf to calculate bandwidth of the network with and without routers.

Iperf can be used with different port numbers using *-p* argument with both server and client commands:

*On server: iperf -s -p 5002*
*On client: iperf -c <IP add of server> -p 5002*

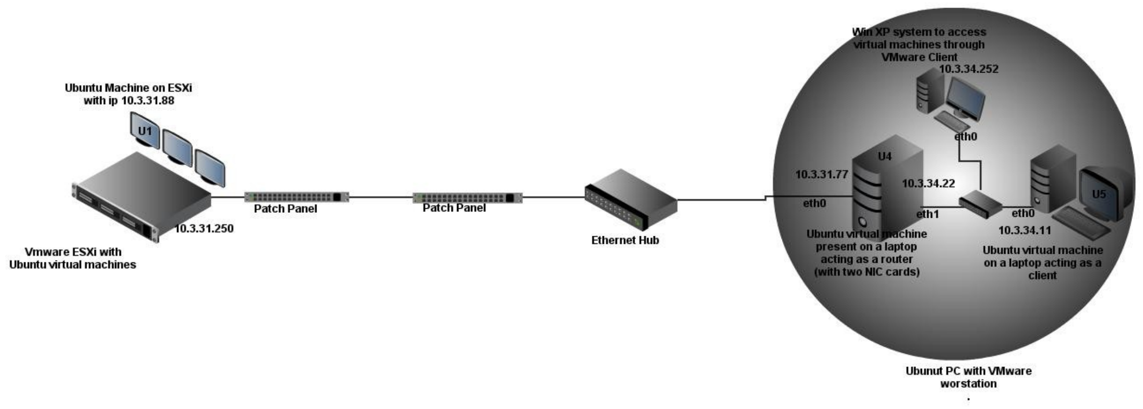
I have used different ports with iperf to check priority implemented in the experiments.

# ***<u>Linux Traffic Control in Virtual Environment</u>***

## 4. Bandwidth and round trip latency of network without routers:

First step is to check the bandwidth of the environment. The environments used were with routers and without routers:

Scenario without routers:


Ubuntu Machine on ESXi with ip 10.3.31.88
U1
10.3.31.250
Vmware ESXi with Ubuntu virtual machines
Patch Panel
Patch Panel
Ethernet Hub
Win XP system to access virtual machines through VMware Client
10.3.34.252
eth0
10.3.31.77
eth0
U4
10.3.34.22
eth1
Ubuntu virtual machine present on a laptop acting as a router (with two NIC cards)
eth0
10.3.34.11
U5
Ubuntu virtual machine on a laptop acting as a client
Ubunut PC with VMware worstation

Here we have an Ubuntu virtual machine (U1) on ESXi server, which is connected to a laptop though a series of patch panels and a hub as shown in above diagram.

The laptop used has three Virtual machines installed on it using VMware Workstation. The machines are as follows:
The first machine is an Ubuntu machine (U4) that has two NIC's and will act as a router to connect the ESXi server to the client machines.
This Ubuntu router (U4) will be connected to a Ubuntu machine (U5) which will act as a client for most of the time to connect with Ubuntu machine (U1) on ESXi server.
The router is connected to another XP machine that is used to access the VM's through VMware Client.
VMware workstation allows us to create a private network for XP and Ubuntu machine.

The IP addressing is as follows:

| | |
|---|---|
| Ubuntu Machine on ESXi | : 10.3.31.88/24 (eth0) |
| Ubuntu Router (U4) | : 10.3.31.77/24 (eth0) |
| Ubuntu Router (U4) | : 10.3.34.22/24 (eth1) |
| Ubuntu Machine (U5) | : 10.3.34.11/24 (eth0) |
| Windows XP | : 10.3.34.252/24 (eth0) |

Network is established with an Ubuntu PC U1 on ESXi server, U4 is a Linux router present on a VMware workstation, and this router connects the ESXi server and another U5 system present on workstation through different networks.

Ubuntu 4 has Ip_forwarding enabled on it with command:

```
sysctl -w net.ipv4.ip_forward=1
```

```
kanwar@ubuntu:~$ sudo sysctl -w net.ipv4.ip_forward=1
[sudo] password for kanwar:
net.ipv4.ip_forward = 1
```

It will enable forwarding the packets between the two interfaces and will make the Ubuntu PC act as a router.

Now the network connectivity is complete. Ping from any machine to any other machine is successful.

Default configuration on the eth0 and eth1 of Ubuntu router(U4):

```
kanwar@ubuntu:~$ sudo tc -s qdisc ls dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 174218911 bytes 115328 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
kanwar@ubuntu:~$ sudo tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 693899 bytes 10310 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Above outputs shows that pfifo (First in, First out) is the default queueing discipline on the interfaces eth0 and eth1 of the Ubuntu router.

Install iperf on both Ubuntu on ESXi-1(U1) and Ubuntu 5 PC using:

```
sudo apt-get install iperf
```

Make U1 as an iperf server with a command: *iperf -s*
So now it will open port number 5001 and will be looking for incoming connections from iperf clients.

On U5 used this command:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 42966 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  86.1 MBytes  72.2 Mbits/sec
```

Above results tell us the bandwidth with which two systems can communicate is around 72Mbits/sec.

On U5:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 20 -i 2
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 27.0 KByte (default)
```

```
------------------------------------------------------------
[  3] local 10.3.34.11 port 51402 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  14.4 MBytes  60.2 Mbits/sec
[  3]  2.0- 4.0 sec  15.9 MBytes  66.8 Mbits/sec
[  3]  4.0- 6.0 sec  17.0 MBytes  71.2 Mbits/sec
[  3]  6.0- 8.0 sec  16.1 MBytes  67.6 Mbits/sec
[  3]  8.0-10.0 sec  15.8 MBytes  66.3 Mbits/sec
[  3] 10.0-12.0 sec  16.4 MBytes  68.6 Mbits/sec
[  3] 12.0-14.0 sec  16.5 MBytes  69.2 Mbits/sec
[  3] 14.0-16.0 sec  15.7 MBytes  65.8 Mbits/sec
[  3] 16.0-18.0 sec  15.0 MBytes  63.1 Mbits/sec
[  3] 18.0-20.0 sec  15.4 MBytes  64.6 Mbits/sec
[  3]  0.0-20.0 sec    158 MBytes  66.3 Mbits/sec
```

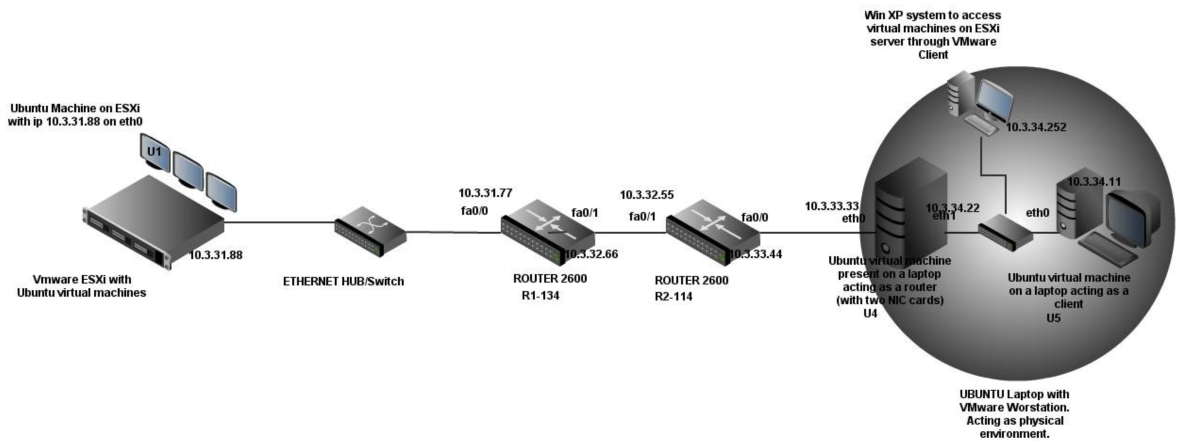
PING test for round-trip latency with default queue discipline from U5 to U1:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.64 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.41 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.26 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.38 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.31 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.22 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.31 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.30 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=63 time=1.45 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=63 time=1.34 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=63 time=1.23 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=63 time=1.21 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=63 time=1.35 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=63 time=1.41 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=63 time=1.48 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=63 time=1.26 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=63 time=1.24 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19041ms
rtt min/avg/max/mdev = 1.212/1.346/1.645/0.102 ms
```

Above results shows us the overall bandwidth available with the systems through the network to communicate and also show us the average round-trip time latency for the network.

## 5. Bandwidth & round trip latency for network with routers:

Now adding two CISCO-2600 series routers in our scenario and configuring the IP addresses as shown in the figure.



Win XP system to access virtual machines on ESXi server through VMware Client
Ubuntu Machine on ESXi with ip 10.3.31.88 on eth0
U1
10.3.34.252
10.3.34.11
10.3.31.77
fa0/0
fa0/1
10.3.32.55
fa0/1
fa0/0
10.3.33.33
eth0
10.3.34.22
eth1
eth0
10.3.31.88
10.3.32.66
10.3.33.44
Vmware ESXi with Ubuntu virtual machines
ETHERNET HUB/Switch
ROUTER 2600
R1-134
ROUTER 2600
R2-114
Ubuntu virtual machine present on a laptop acting as a router (with two NIC cards)
U4
Ubuntu virtual machine on a laptop acting as a client
U5
UBUNTU Laptop with VMware Worstation. Acting as physical environment.

Router Configurations:
Configuration for static routing on both routers.

```
R1-134#sh run
Building configuration...

Current configuration : 774 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R1-134
!
boot-start-marker
boot-end-marker
!
memory-size iomem 10
no aaa new-model
ip subnet-zero
!
ip cef
!
interface FastEthernet0/0
```

```
 ip address 10.3.31.77 255.255.255.0
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 clock rate 64000
!
interface FastEthernet0/1
 ip address 10.3.32.66 255.255.255.0
 duplex auto
 speed auto
!
interface Serial0/1
 no ip address
 shutdown
!
ip http server
ip classless
ip route 10.3.33.0 255.255.255.0 10.3.32.55
ip route 10.3.34.0 255.255.255.0 10.3.32.55
!
voice-port 1/0/0
!
voice-port 1/0/1
!
!
!
!
line con 0
line aux 0
line vty 0 4
!
!
end

R2-114# sh run
Building configuration...

Current configuration : 766 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R2-114
!
boot-start-marker
```

```
boot-end-marker
!
!
memory-size iomem 10
no aaa new-model
ip subnet-zero
!
ip cef
!
interface FastEthernet0/0
 ip address 10.3.33.44 255.255.255.0
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
!
interface FastEthernet0/1
 ip address 10.3.32.55 255.255.255.0
 duplex auto
 speed auto
!
interface Serial0/1
 no ip address
 shutdown
!
ip http server
ip classless
ip route 10.3.31.0 255.255.255.0 10.3.32.66
ip route 10.3.34.0 255.255.255.0 10.3.33.33
!
voice-port 1/0/0
!
voice-port 1/0/1
!
line con 0
line aux 0
line vty 0 4
!
end
```

Now the network connectivity is complete and U1 and Win XP PC can ping U5.
No traffic control is done, and the readings below are the default values achieved by iperf and rtt-ping tests:

On U5:

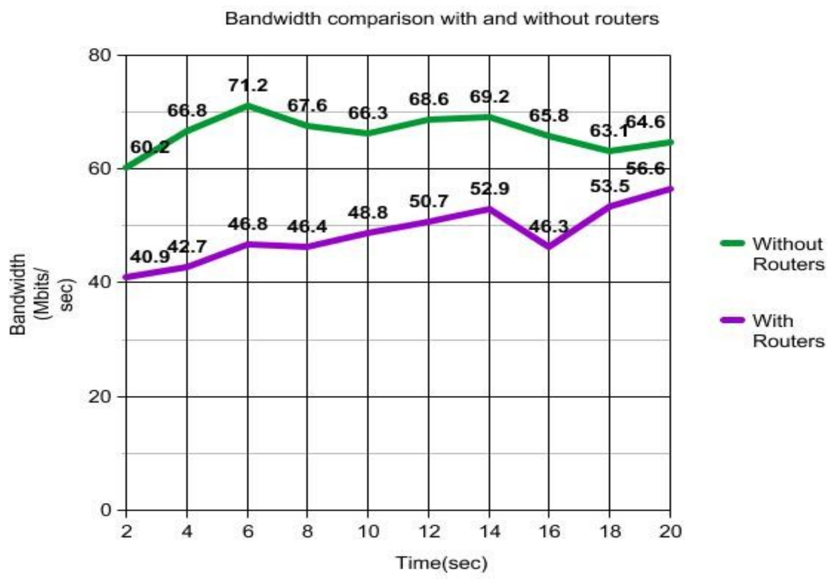
```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
```

```
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 46011 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  56.4 MBytes  47.2 Mbits/sec

ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 20 -i 2
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 57861 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec  4.88 MBytes  40.9 Mbits/sec
[  3]  2.0- 4.0 sec  5.09 MBytes  42.7 Mbits/sec
[  3]  4.0- 6.0 sec  5.58 MBytes  46.8 Mbits/sec
[  3]  6.0- 8.0 sec  5.53 MBytes  46.4 Mbits/sec
[  3]  8.0- 10.0 sec  5.81 MBytes  48.8 Mbits/sec
[  3]  10.0- 12.0 sec  6.04 MBytes  50.7 Mbits/sec
[  3]  12.0- 14.0 sec  6.30 MBytes  52.9 Mbits/sec
[  3]  14.0- 16.0 sec  5.52 MBytes  46.3 Mbits/sec
[  3]  16.0- 18.0 sec  6.38 MBytes  53.5 Mbits/sec
[  3]  18.0-20.0 sec  6.74 MBytes  56.6 Mbits/sec
[  3]  0.0-20.0 sec  57.9 MBytes  48.3 Mbits/sec
```

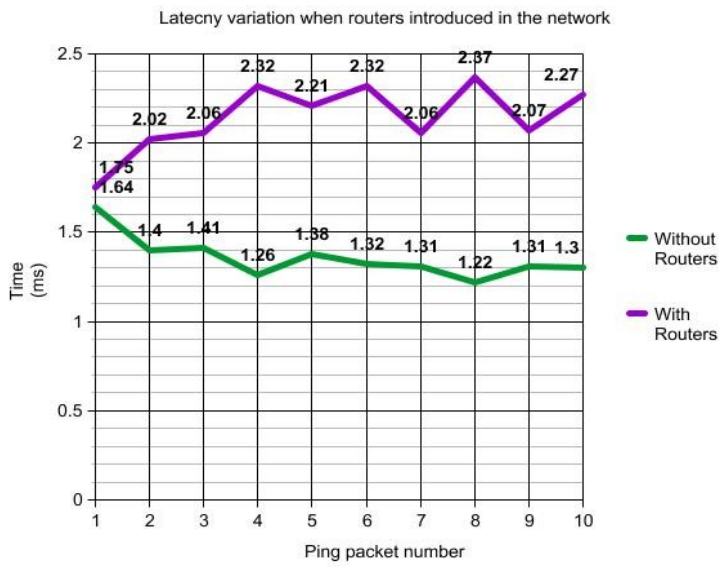
Ping test for rtt-latency:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=1.75 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=2.02 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=2.06 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=2.32 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=2.21 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=2.32 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=2.06 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=2.37 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=2.07 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=2.27 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=61 time=1.67 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=61 time=2.24 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=61 time=2.33 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=61 time=2.09 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=61 time=2.09 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=61 time=2.03 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=61 time=1.93 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=61 time=1.98 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=61 time=1.71 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=61 time=2.12 ms
```

```
--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19035ms
rtt min/avg/max/mdev = 1.673/2.086/2.378/0.201 ms
```



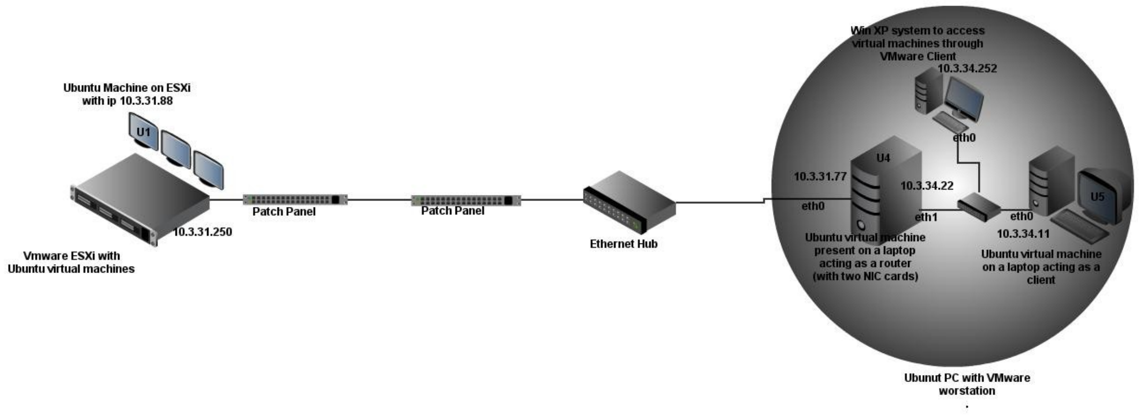
*The two experiments were done at different instances of time; this graph shows only the bandwidth reached after every 2 seconds in a period of 20 seconds.*



*Routers adding latency to the network.*

## 6. Queue discipline on network without routers:

Scenario for this experiment is as follows:


Ubuntu Machine on ESXi with ip 10.3.31.88
U1
10.3.31.250
Vmware ESXi with Ubuntu virtual machines
Patch Panel
Patch Panel
Ethernet Hub
Win XP system to access virtual machines through VMware Client
10.3.34.252
eth0
U4
10.3.31.77
eth0
10.3.34.22
eth1
Ubuntu virtual machine present on a laptop acting as a router (with two NIC cards)
U5
eth0
10.3.34.11
Ubuntu virtual machine on a laptop acting as a client
Ubunut PC with VMware worstation

## 6.1 htb to limit bandwidth to a known rate:

Here in this experiment the bandwidth of the network will be limited to 6Mbits/sec
Our scenario shows that all the client machines in the network will access the ESXi server Virtual Machines through Ubuntu router U4, the configuration will be done on this machine (U4) to limit the bandwidth to a known rate.

Default configuration on Ubuntu router(U4)interfaces before htb:

```
kanwar@ubuntu:~$ sudo tc -s qdisc ls dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 174218911 bytes 115328 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
kanwar@ubuntu:~$ sudo tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 693899 bytes 10310 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Commands to conf htb with rate of 10 Mbits/sec exit interface eth0 of Ubuntu router(U4):

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit burst 15k
```

First commands will create a root htb queue discipline on interface eth0, and will define that the default class that is to be followed is class 10, which is defined in the next command and has a limit to the

bandwidth of the interface of Ubuntu router.
The above commands will make the rate of eth0 of the router to be no more than 10Mbits/sec.
*ceil* value can be set to more than the rate and the bandwidth will increase for the network if more bandwidth is available with the network; c*eil* is used to define the max value that it can reach.
This can be verified by iperf tests as shown below:

Configuration on Ubuntu router(U4) interfaces after tc:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0

root@ubuntu:/# tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

<u>Check bandwidth and round trip latency from our client machine (U5) to the Ubuntu server (U1):</u>

We have iperf server running on U1.
Iperf test on Ubuntu client (U5):

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 57516 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  12.1 MBytes  9.95 Mbits/sec

ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 58779 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.19 MBytes  9.96 Mbits/sec
[  3]  1.0- 2.0 sec  1.16 MBytes  9.76 Mbits/sec
[  3]  2.0- 3.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  3.0- 4.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  4.0- 5.0 sec  1.13 MBytes  9.50 Mbits/sec
[  3]  5.0- 6.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  6.0- 7.0 sec  1.11 MBytes  9.31 Mbits/sec
[  3]  7.0- 8.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  8.0- 9.0 sec  1.14 MBytes  9.57 Mbits/sec
[  3]  9.0-10.0 sec  1.23 MBytes  10.3 Mbits/sec
[  3]  0.0-10.1 sec  11.5 MBytes  9.60 Mbits/sec
```

rtt-latency test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.21 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.37 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.41 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.23 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.30 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.49 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=63 time=1.33 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=63 time=1.31 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=63 time=1.48 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=63 time=1.25 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=63 time=1.44 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=63 time=1.35 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=63 time=1.33 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=63 time=1.33 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=63 time=1.40 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19037ms

rtt min/avg/max/mdev = 1.212/1.358/1.497/0.080 ms
```

Above test shows how we can limit the bandwidth of a network using Linux Traffic Control to a certain value.

Now the configuration for various queueing disciplines used with htb implemented for limited bandwidth:

## 6.2 htb with pfifo:

One that enters first in a queue will leave first from the queue.

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit burst 15k
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 5
```

As compared to last configuration we have added just one more command that will make the class to use FIFO approach to de-queue the packets from the queue. Limit specifies the maximum queue size one can use with the discipline.

Configuration on eth0 of our Ubuntu Router U4:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 10 direct_packets_stat 0
 Sent 394944424 bytes 261168 pkt (dropped 0, overlimits 500304 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 20: parent 1:10 limit 5p
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Above configuration also shows that pfifo is implemented with a limit of 5 with htb.

Iperf test:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 41907 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.3 MBytes  9.44 Mbits/sec

ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 41908 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.12 MBytes  9.37 Mbits/sec
[  3]  1.0- 2.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  2.0- 3.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  3.0- 4.0 sec  1.13 MBytes  9.50 Mbits/sec
[  3]  4.0- 5.0 sec  1.10 MBytes  9.24 Mbits/sec
[  3]  5.0- 6.0 sec  1.11 MBytes  9.31 Mbits/sec
[  3]  6.0- 7.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  7.0- 8.0 sec  1.14 MBytes  9.57 Mbits/sec
[  3]  8.0- 9.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  9.0-10.0 sec  1.13 MBytes  9.50 Mbits/sec
[  3]  0.0-10.0 sec  11.2 MBytes  9.43 Mbits/sec
```

Ping test

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.50 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.14 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.60 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.30 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.20 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.27 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.42 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.30 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.58 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.29 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=63 time=1.02 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=63 time=1.18 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=63 time=1.16 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=63 time=1.29 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=63 time=1.13 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=63 time=1.42 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=63 time=1.39 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=63 time=1.24 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=63 time=1.40 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19038ms
rtt min/avg/max/mdev = 1.026/1.311/1.604/0.150 ms
```

## <u>**6.3 htb with sfq (Stochastic Fairness Queueing)**</u>**:**

In sfq the traffic is sent in round robin fashion, which gives a fair chance to each and every session to send their data. It will just do the scheduling for the traffic on the interface it is implemented on. The scheduling is done in a manner that each type of traffic should get a fair amount of equal time after regular intervals to send the packets. This switching from one traffic to another is a very fast process.

Delete all the qdisc configuration by htb on the eth0 interface of Ubuntu Router:

```
root@ubuntu:/# tc qdisc del dev eth0 root
```

Now the interface configurations of router interfaces will be default pfifo:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
root@ubuntu:/# tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 9935556 bytes 149532 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Implementing SFQ on the exit interface eth0 of Ubuntu router (U4):

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit burst 15k
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 20: sfq
```

Configuration on eth0:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 20: parent 1:10 limit 127p quantum 1514b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Iperf test from Ubuntu client (U5):

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 27.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 54224 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  11.7 MBytes  9.73 Mbits/sec
```

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 27.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 54230 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.19 MBytes  9.96 Mbits/sec
[  3]  1.0- 2.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  2.0- 3.0 sec  1.12 MBytes  9.37 Mbits/sec
[  3]  3.0- 4.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  4.0- 5.0 sec  1.18 MBytes  9.90 Mbits/sec
[  3]  5.0- 6.0 sec  1.18 MBytes  9.90 Mbits/sec
[  3]  6.0- 7.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  7.0- 8.0 sec  1.20 MBytes  10.0 Mbits/sec
[  3]  8.0- 9.0 sec  1.14 MBytes  9.57 Mbits/sec
[  3]  9.0-10.0 sec  1.20 MBytes  10.0 Mbits/sec
[  3]  0.0-10.1 sec  11.7 MBytes  9.73 Mbits/sec
```

Ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.51 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.36 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.17 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.16 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.17 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.23 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.10 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.12 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.29 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=63 time=1.35 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=63 time=1.33 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=63 time=1.14 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=63 time=1.22 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=63 time=1.28 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=63 time=1.11 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=63 time=1.29 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=63 time=1.21 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=63 time=1.25 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=63 time=1.10 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19038ms
rtt min/avg/max/mdev = 1.104/1.239/1.517/0.116 ms
```

## 6.4 htb with sfq (perturb value):

Perturb value defines that after how much time the hashing will be reconfigured. If not set hashing will never be reconfigured.

Delete previous htb configuration:

```
root@ubuntu:/# tc qdisc del dev eth0 root
```

Check Configuration:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Commands on eth0:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit burst 15k
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 20: sfq perturb 10
```

Configuration of eth0 after above commands(sfq with a limit):

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 20: parent 1:10 limit 127p quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Iperf test:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 57422 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  11.5 MBytes  9.55 Mbits/sec

ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 57428 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.35 MBytes  11.3 Mbits/sec
[  3]  1.0- 2.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  2.0- 3.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  3.0- 4.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  4.0- 5.0 sec  1.05 MBytes  8.85 Mbits/sec
[  3]  5.0- 6.0 sec  1.14 MBytes  9.57 Mbits/sec
[  3]  6.0- 7.0 sec  1.14 MBytes  9.57 Mbits/sec
[  3]  7.0- 8.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  8.0- 9.0 sec  1.16 MBytes  9.76 Mbits/sec
[  3]  9.0-10.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  0.0-10.1 sec  11.6 MBytes  9.68 Mbits/sec
```

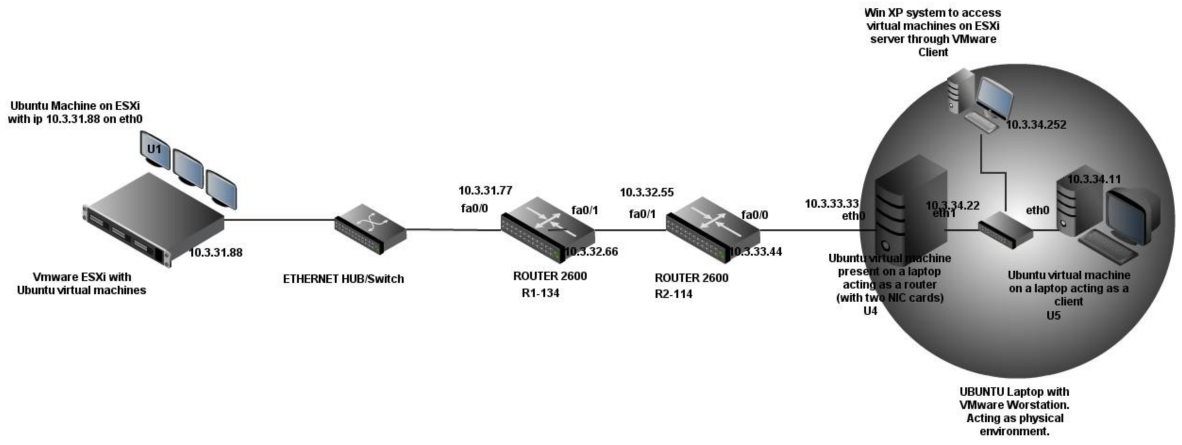
Ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.38 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.16 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.37 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=0.977 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.10 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.15 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.25 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.10 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.30 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.34 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=63 time=1.11 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=63 time=1.07 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=63 time=1.47 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=63 time=1.36 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=63 time=1.14 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=63 time=1.27 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=63 time=1.19 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=63 time=1.06 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=63 time=1.20 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19037ms
rtt min/avg/max/mdev = 0.977/1.220/1.479/0.135 ms
```

*sfq can be helpful in environments where traffic is high and all sort of traffic needs to be transmitted and helps not to let any type of traffic to dominate and suppress other transmissions.*

## 7. Queue disciplines on network with routers:

Routers will increase round trip latency of the network.
Now we have a scenario as shown in the above diagram:


Win XP system to access virtual machines on ESXi server through VMware Client
Ubuntu Machine on ESXi with ip 10.3.31.88 on eth0
U1
10.3.31.88
Vmware ESXi with Ubuntu virtual machines
ETHERNET HUB/Switch
10.3.31.77
fa0/0
fa0/1
10.3.32.66
ROUTER 2600
R1-134
10.3.32.55
fa0/1
fa0/0
10.3.33.44
ROUTER 2600
R2-114
10.3.33.33
eth0
10.3.34.22
eth1
10.3.34.252
10.3.34.11
eth0
Ubuntu virtual machine present on a laptop acting as a router (with two NIC cards)
U4
Ubuntu virtual machine on a laptop acting as a client
U5
UBUNTU Laptop with VMware Worstation. Acting as physical environment.

We have two router in between the server and client,, this introduces more round trip latency and even will reduce the effective bandwidth of the link between server and client.
The entire configuration is same for the experiments done in last section for pfifo and sfq, only difference is the readings for throughput and rtt-latency because of the routers in the network; also two different tests for tbf and prio are done in this section.

Now first test is for: Using htb to limit bandwidth of an interface eth0 on Ubuntu router U5 to 6Mbits/sec:

Check default configuration on Ethernet interfaces of router:

```
root@ubuntu:/# tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 567302247 bytes 412898 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 315826122 bytes 378061 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Commands to configure htb with rate of 10 mbps on exit interface eth0 for client Ubuntu 6:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit burst 15k
```

Configuration on Ubuntu router interfaces:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 10 direct_packets_stat 0
 Sent 3516 bytes 14 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
root@ubuntu:/# tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 567325296 bytes 412924 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Iperf test:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 57381 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  12.0 MBytes  9.96 Mbits/sec

ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 47566 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.19 MBytes  9.96 Mbits/sec
[  3]  1.0- 2.0 sec  1.02 MBytes  8.59 Mbits/sec
[  3]  2.0- 3.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  3.0- 4.0 sec  1.13 MBytes  9.50 Mbits/sec
[  3]  4.0- 5.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  5.0- 6.0 sec  1.12 MBytes  9.37 Mbits/sec
[  3]  6.0- 7.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  7.0- 8.0 sec  1.20 MBytes  10.0 Mbits/sec
[  3]  8.0- 9.0 sec  1.10 MBytes  9.24 Mbits/sec
[  3]  9.0-10.0 sec  1.20 MBytes  10.1 Mbits/sec
[  3]  0.0-10.0 sec  11.4 MBytes  9.50 Mbits/sec
```

*ping test(To know round trip latency):*

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
```

```
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=1.88 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=1.99 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=2.00 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.82 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=1.73 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=1.84 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=1.73 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=1.83 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=1.89 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=61 time=1.95 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=61 time=1.76 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=61 time=1.84 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=61 time=2.02 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=61 time=1.73 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=61 time=2.21 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=61 time=1.79 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=61 time=1.89 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=61 time=1.88 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=61 time=1.87 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19041ms
rtt min/avg/max/mdev = 1.733/1.883/2.216/0.122 ms
```

## 7.1 htb with pfifo:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit burst 15k
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 5
```

Configuration on ethernet interface of Ubuntu router(U4):

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 10 direct_packets_stat 0
 Sent 288637671 bytes 191053 pkt (dropped 0, overlimits 356443 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 20: parent 1:10 limit 5p
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Eth1 will have default configuration:

```
root@ubuntu:/# tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 574023259 bytes 511316 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

*iperf tests:*

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 49117 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.2 MBytes  9.44 Mbits/sec

ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 49118 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.16 MBytes  9.70 Mbits/sec
[  3]  1.0- 2.0 sec  1.12 MBytes  9.37 Mbits/sec
[  3]  2.0- 3.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  3.0- 4.0 sec  1.12 MBytes  9.37 Mbits/sec
[  3]  4.0- 5.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  5.0- 6.0 sec  1.14 MBytes  9.57 Mbits/sec
[  3]  6.0- 7.0 sec  1.11 MBytes  9.31 Mbits/sec
[  3]  7.0- 8.0 sec  1.13 MBytes  9.50 Mbits/sec
[  3]  8.0- 9.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  9.0-10.0 sec  1.12 MBytes  9.37 Mbits/sec
[  3]  0.0-10.0 sec  11.3 MBytes  9.45 Mbits/sec
```

ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=2.23 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=1.92 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=1.81 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=1.86 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.81 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=1.77 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=1.83 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=1.71 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=1.84 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=1.62 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=61 time=1.81 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=61 time=2.02 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=61 time=1.94 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=61 time=1.91 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=61 time=1.83 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=61 time=1.79 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=61 time=1.93 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=61 time=1.80 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19043ms
rtt min/avg/max/mdev = 1.623/1.865/2.235/0.123 ms
```

## 7.2 htb with sfq:

Delete previous htb configuration on eth0:

```
root@ubuntu:/# tc qdisc del dev eth0 root
```

Ubuntu Router interface configurations:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
root@ubuntu:/# tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 574908934 bytes 523188 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Command to be put on Ubuntu Router to configure sfq:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit burst 15k
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 20: sfq perturb 10
```

Now Eth0 Configuration will be:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 20: parent 1:10 limit 127p quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

iperf test:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 47967 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.4 MBytes  9.56 Mbits/sec

ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 47969 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.33 MBytes  11.1 Mbits/sec
[  3]  1.0- 2.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  2.0- 3.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  3.0- 4.0 sec  1.11 MBytes  9.31 Mbits/sec
[  3]  4.0- 5.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  5.0- 6.0 sec  1.06 MBytes  8.91 Mbits/sec
[  3]  6.0- 7.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  7.0- 8.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  8.0- 9.0 sec  1.15 MBytes  9.63 Mbits/sec
[  3]  9.0-10.0 sec  1.06 MBytes  8.91 Mbits/sec
[  3]  0.0-10.0 sec  11.5 MBytes  9.60 Mbits/sec
```

ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=2.10 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=1.77 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=1.94 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=1.93 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.78 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=1.85 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=2.11 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=1.93 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=61 time=2.00 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=61 time=2.12 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=61 time=1.94 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=61 time=1.83 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=61 time=1.93 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=61 time=1.86 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=61 time=1.97 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=61 time=1.79 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19039ms
rtt min/avg/max/mdev = 1.774/1.926/2.123/0.115 ms
```

## <u>7.3 TBF(Token bucket filter):</u>

TBF is a queue discipline used to set rate of an interface to a particular level. It only passes the received packets only at a specific rate, no matter how fast it receives them. It is used to limit the upload from an interface to any unlink, so that it should not suppress other data transfers.
It should be used on the exit interface of the machine from which that machines connects to the uplink.

First in this scenario the overall bandwidth of the network was limited to 800kbits/sec using htb. This was done on the Ubuntu router as the communication is in between Virtual machines on ESXi server and other Ubuntu machine on the other side of the Ubuntu router, present on VMware workstation.

```
root@ubuntu:/# tc qdisc del dev eth0 root
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 10
root@ubuntu:/# sudo tc class add dev eth0 parent 1:1 classid 1:10 htb rate 800kbit ceil 800kbit burst
15k
```

Iperf test will show us the bandwidth of the network:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 37203 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.9 sec  1.24 MBytes    954 Kbits/sec
```

Now we will use htb to control the speed of the ethernet interface of our Ubuntu system U5:

```
root@ubuntu:/#tc qdisc add dev eth0 root tbf rate 220kbit latency 50ms burst 1540
```

The above command can be used on the network interface itself without any htb configuration.
Now iperf test will show the real available bandwidth for the client machine to upload:

```
root@ubuntu:/# iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 37204 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec    272 KBytes    215 Kbits/sec
```

This scenario is useful when we are connected to any DSL modem or a cable modem through an Ethernet interface. Ethernet interface can upload data at a very fast rate and will suppress other interactive data, so tbf is helpful over here.

## 7.4 PRIO (priority):

There is no traffic control implemented on any interface of any machine:
Now we use the following configuration to implement prio:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: prio
root@ubuntu:/# tc qdisc add dev eth0 parent 1:1 handle 10: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:2 handle 20: tbf rate 1000kbit buffer 1600 limit 3000
root@ubuntu:/# tc qdisc add dev eth0 parent 1:3 handle 30: sfq
```

The first command in above configuration creates a root qdisc 1: and it has three classes 1:1, 1:2 and 1:3, these three classes have separate qdisc implemented in them as shown. By default all the three classes have pfifo but in above configuration it has got three different queue disciplines.

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc prio 1: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 4672 bytes 31 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 10: parent 1:1 limit 127p quantum 1514b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc tbf 20: parent 1:2 rate 1000Kbit burst 1600b lat 11.2ms
 Sent 606 bytes 9 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 30: parent 1:3 limit 127p quantum 1514b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Above configuration on the interface eth0 shows that we have three priority levels 10, 20 and 30 in decreasing priority. At this moment very less traffic in there through the interface that is why sent bytes in above output has either zero or very less value for priority levels 10, 20 and 30.
The amount of data sent should be noted here to be more familiar with the way prio works.
Bulk traffic will follow 30(less priority), and interactive traffic (more priority traffic) will follow levels 20 and 10:

For bulk traffic test we will use scp (secure copy) to copy a file from Ubuntu U5 to Ubuntu U1 on ESXi:
We have a file on desktop of machine U5 that we will copy to U1.

```
root@ubuntu:/home/ubuntu/Desktop# ls -l
total 105368
-rwxr-xr-x 1 ubuntu ubuntu      412 2011-06-06 15:06 filezilla.desktop
-rwxr-xr-x 1 ubuntu ubuntu      476 2011-05-24 11:53 gnome-terminal.desktop
-rw-r--r-- 1 ubuntu ubuntu   582321 2011-06-14 17:24 interactive traffic.png
-rwxr-xr-x 1 ubuntu ubuntu      226 2011-06-14 16:04 putty.desktop
-rw-r--r-- 1 ubuntu ubuntu   121344 2011-06-14 19:42 TEST 1 Readings with high bandwidth.doc
-rw-r--r-- 1 ubuntu ubuntu     8565 2011-06-14 14:36 Test File.odt
-rw-r--r-- 1 root   root   53576427 2011-06-14 19:23 ubuntu@10.3.31.88
-rw-r--r-- 1 ubuntu ubuntu 53576427 2011-06-14 16:17 Video08-3.wmv
```

```
-rwxr-xr-x 1 ubuntu ubuntu     3679 2011-06-02 13:19 vlc.desktop
```

*Video08-3.wmv is the file that we will copy to other machine.*

Command to copy files from U5 to U1:

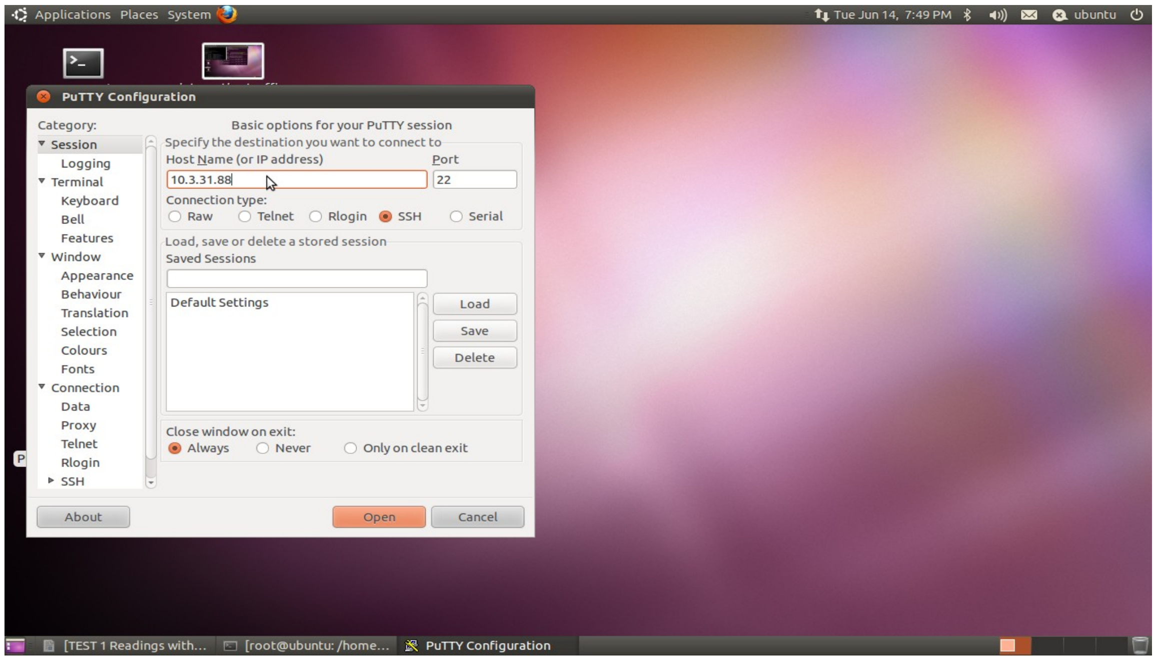
```
root@ubuntu:/home/ubuntu/Desktop# scp Video08-3.wmv kanwar@10.3.31.88:./
kanwar@10.3.31.88's password:
Video08-3.wmv                              100%   51MB   5.1MB/s   00:10
```

File copied now we will check the eth0 of our router U4 to see which priority level the packet followed:

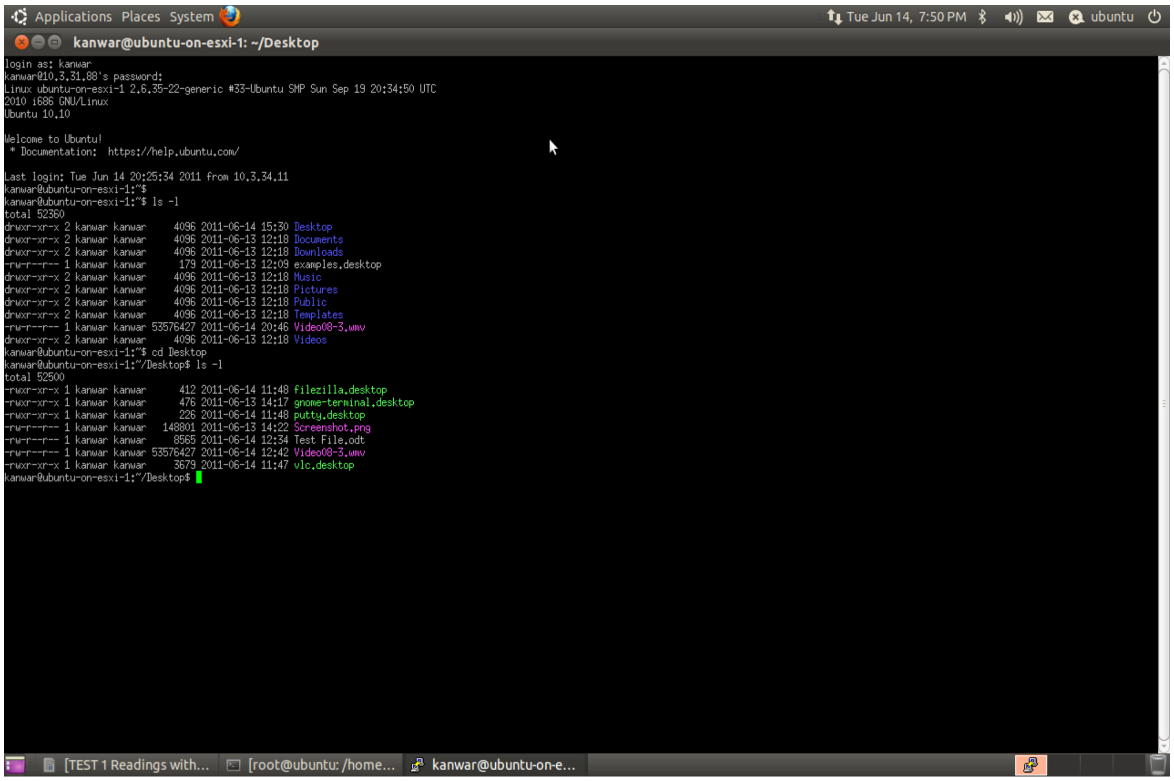
```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc prio 1: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 56198623 bytes 38059 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 10: parent 1:1 limit 127p quantum 1514b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc tbf 20: parent 1:2 rate 1000Kbit burst 1600b lat 11.2ms
 Sent 88799 bytes 946 pkt (dropped 0, overlimits 3 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 30: parent 1:3 limit 127p quantum 1514b
 Sent 56105758 bytes 37091 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

They went through least priority level 30.

Now will check for more priority data:
ssh using putty to U1 from U5:



Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address): 10.3.31.88
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Close window on exit: [✓ Always] Never Only on clean exit



```
login as: kanwar
kanwar@10.3.31.88's password:
Linux ubuntu-on-esxi-1 2.6.35-22-generic #33-Ubuntu SMP Sun Sep 19 20:34:50 UTC
2010 i686 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

Last login: Tue Jun 14 20:25:34 2011 from 10.3.34.11
kanwar@ubuntu-on-esxi-1:~$
kanwar@ubuntu-on-esxi-1:~$ ls -l
total 52360
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-14 15:30 Desktop
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-13 12:18 Documents
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-13 12:18 Downloads
-rw-r--r-- 1 kanwar kanwar      179 2011-06-13 12:09 examples.desktop
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-13 12:18 Music
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-13 12:18 Pictures
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-13 12:18 Public
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-13 12:18 Templates
-rw-r--r-- 1 kanwar kanwar 53576427 2011-06-14 20:46 Video08-3.wmv
drwxr-xr-x 2 kanwar kanwar     4096 2011-06-13 12:18 Videos
kanwar@ubuntu-on-esxi-1:~$ cd Desktop
kanwar@ubuntu-on-esxi-1:~/Desktop$ ls -l
total 52500
-rwxr-xr-x 1 kanwar kanwar      412 2011-06-14 11:48 filezilla.desktop
-rwxr-xr-x 1 kanwar kanwar      476 2011-06-13 14:17 gnome-terminal.desktop
-rwxr-xr-x 1 kanwar kanwar      226 2011-06-14 11:48 putty.desktop
-rw-r--r-- 1 kanwar kanwar   148801 2011-06-13 14:22 Screenshot.png
-rw-r--r-- 1 kanwar kanwar     8565 2011-06-14 12:34 Test File.odt
-rw-r--r-- 1 kanwar kanwar 53576427 2011-06-14 12:42 Video08-3.wmv
-rwxr-xr-x 1 kanwar kanwar     3679 2011-06-14 11:47 vlc.desktop
kanwar@ubuntu-on-esxi-1:~/Desktop$
```

Few commands were run in the ssh session from our client on U5 interface available through ssh as shown in above picture.

Now we will check the eth 0 of our Ubuntu router:

*root@ubuntu:/# tc -s qdisc ls dev eth0*
*qdisc prio 1: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1*
 *Sent 56228167 bytes 38278 pkt (dropped 0, overlimits 0 requeues 0)*
 *backlog 0b 0p requeues 0*
*qdisc sfq 10: parent 1:1 limit 127p quantum 1514b*
 *Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)*
 *backlog 0b 0p requeues 0*
***qdisc tbf 20: parent 1:2 rate 1000Kbit burst 1600b lat 11.2ms***
 ***Sent 118343 bytes 1165 pkt (dropped 0, overlimits 5 requeues 0)***
 ***backlog 0b 0p requeues 0***
*qdisc sfq 30: parent 1:3 limit 127p quantum 1514b*
 *Sent 56105758 bytes 37091 pkt (dropped 0, overlimits 0 requeues 0)*
 *backlog 0b 0p requeues 0*

Traffic went through priority  level 20.

Now for telnet:
*Telnet connection to Cisco router from Ubuntu 5 was established and few commands were run:*
*Telnet 10.3.32.66*

Now after running some commands on Cisco router using the telnet session, the eth0 of Ubuntu router U4 shows:

```
root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc prio 1: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 56248232 bytes 38484 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 10: parent 1:1 limit 127p quantum 1514b
 Sent 6249 bytes 112 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc tbf 20: parent 1:2 rate 1000Kbit burst 1600b lat 11.2ms
 Sent 132159 bytes 1259 pkt (dropped 0, overlimits 6 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 30: parent 1:3 limit 127p quantum 1514b
 Sent 56105758 bytes 37091 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Shows traffic went thorough highest priority for telnet.

Iperf test:

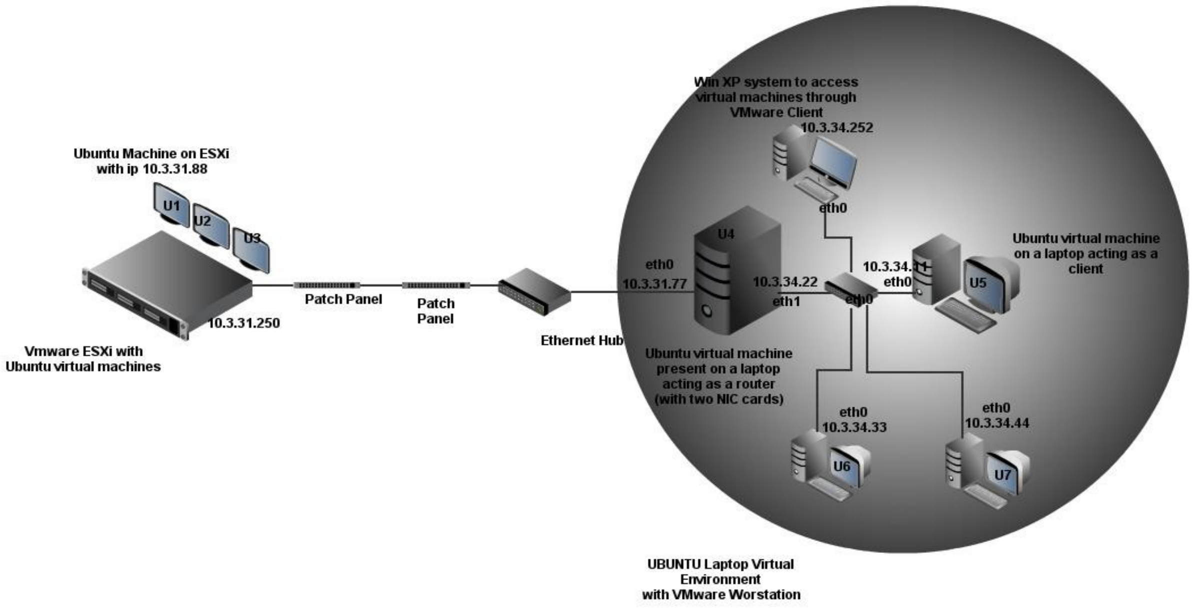
```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 47122 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.13 MBytes    947 Kbits/sec

root@ubuntu:/# tc -s qdisc ls dev eth0
qdisc prio 1: root refcnt 2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 57638550 bytes 41055 pkt (dropped 236, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 10: parent 1:1 limit 127p quantum 1514b
 Sent 6249 bytes 112 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc tbf 20: parent 1:2 rate 1000Kbit burst 1600b lat 11.2ms
 Sent 1522477 bytes 3830 pkt (dropped 236, overlimits 1095 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 30: parent 1:3 limit 127p quantum 1514b
 Sent 56105758 bytes 37091 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Iperf test is also followed the middle priority handle 20.
So above all test shows us that traffic can be limited to a particular value using htb and we can use different queueing disciplines with them too.

## 8. Limit bandwidth of particular user.

### 8.1 Without Routers:

In the scenario there are now two more Ubuntu machines U6 and U7 on VMware Workstation.



The bandwidth available and rtt-latency for the network:

iperf test from a clientU6 to Ubuntu iperf server U1 on ESXi:

```
kanwar2@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 39547 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  85.9 MBytes  72.0 Mbits/sec
```

ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.43 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.23 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.23 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.36 ms
```

```
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.23 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.21 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.41 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.41 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.18 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=63 time=1.39 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=63 time=1.15 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=63 time=1.21 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=63 time=1.66 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=63 time=1.33 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=63 time=1.37 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=63 time=1.36 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=63 time=2.90 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=63 time=1.38 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19040ms
rtt min/avg/max/mdev = 1.158/1.415/2.904/0.361 ms
```

<u>To limit the bandwidth of particular user:</u>
For this IP address of the different machines will be added to filters used in traffic control configuration on Ubuntu Server U4:
There are two Ubuntu machines behind the Ubuntu router U4 with IP addresses as follows:
Ubuntu 5: 10.3.34.11/24
Ubuntu 6: 10.3.34.33/24

Total bandwidth is around 70 Mbits/sec for both clients. Bandwidth is to be limited as follows:
Ubuntu U5: 6 Mbits/sec
Ubuntu U6: 3 Mbits/sec
Default will be around: 1 Mbits/sec(i.e. any other machine from any other network or the same network that will try to access U1 through Ubuntu router U4 will fall in this category and will not be given bandwidth more than 1 Mbits/sec).

So the configuration on our Ubuntu router U4 is be as follows:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 10mbit burst 15k
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 6mbit ceil 6mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 3mbit ceil 3mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1mbit ceil 1mbit
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: sfq perturb 10
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 10.3.34.11 flowid
1:10
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 10.3.34.33 flowid
1:20
```

This traffic control configuration works as follows:
Firstly, packets source address is matched with the filters attached, if it is of any of the first two filters them it will be sent to corresponding class (10 or 20) else it will follow the default class 30 of 1Mbits/sec.
In above configuration instead of sfq we can use pfifo or pfifo with a limit. They will only change the way the packets are sent onto the network by the exit interface eth0 of Ubuntu router U4.

Below is the information received by the show commands on Ubuntu router U4: (It shows the configuration of the interface i.e. all the traffic control implemented on that interface)

```
root@ubuntu:/# tc -s -d qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 30 direct_packets_stat 0 ver 3.17
 Sent 36240254 bytes 23975 pkt (dropped 97, overlimits 48069 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 10: parent 1:10 limit 127p quantum 1514b flows 127/1024 perturb 10sec
 Sent 7699172 bytes 5090 pkt (dropped 2, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 20: parent 1:20 limit 127p quantum 1514b flows 127/1024 perturb 10sec
 Sent 28540872 bytes 18880 pkt (dropped 95, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 30: parent 1:30 limit 127p quantum 1514b flows 127/1024 perturb 10sec
 Sent 210 bytes 5 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
```

Above outputs shows details about the queue disciplines added to eth0.

```
root@ubuntu:/# tc -s -d class show dev eth0
class htb 1:1 root rate 10000Kbit ceil 10000Kbit burst 15Kb/8 mpu 0b overhead 0b cburst 1600b/8 mpu
0b overhead 0b level 7
 Sent 36239230 bytes 23975 pkt (dropped 0, overlimits 0 requeues 0)
 rate 73120bit 6pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 191110 ctokens: 19110

class htb 1:10 parent 1:1 leaf 10: prio 0 quantum 75000 rate 6000Kbit ceil 6000Kbit burst 1599b/8 mpu
0b overhead 0b cburst 1599b/8 mpu 0b overhead 0b level 0
 Sent 7699172 bytes 5090 pkt (dropped 2, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 5090 borrowed: 0 giants: 0
 tokens: 31828 ctokens: 31828


class htb 1:20 parent 1:1 leaf 20: prio 0 quantum 37500 rate 3000Kbit ceil 3000Kbit burst 1599b/8 mpu
0b overhead 0b cburst 1599b/8 mpu 0b overhead 0b level 0
 Sent 28540872 bytes 18880 pkt (dropped 95, overlimits 0 requeues 0)
 rate 72152bit 6pps backlog 0b 0p requeues 0
 lended: 18880 borrowed: 0 giants: 0
 tokens: 63656 ctokens: 63656
```

```
class htb 1:30 parent 1:1 leaf 30: prio 0 quantum 12500 rate 1000Kbit ceil 1000Kbit burst 1600b/8 mpu
0b overhead 0b cburst 1600b/8 mpu 0b overhead 0b level 0
 Sent 210 bytes 5 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 5 borrowed: 0 giants: 0
 tokens: 194000 ctokens: 194000

root@ubuntu:/# tc -s -d filter show dev eth0
filter parent 1: protocol ip pref 1 u32
filter parent 1: protocol ip pref 1 u32 fh 800: ht divisor 1
filter parent 1: protocol ip pref 1 u32 fh 800::800 order 2048 key ht 800 bkt 0 flowid 1:10
  match 0a03220b/ffffffff at 12
filter parent 1: protocol ip pref 1 u32 fh 800::801 order 2049 key ht 800 bkt 0 flowid 1:20
  match 0a032221/ffffffff at 12
```

In the configuration there are three different classes with different bandwidth, so to test the configuration iperf test results from three different machines:

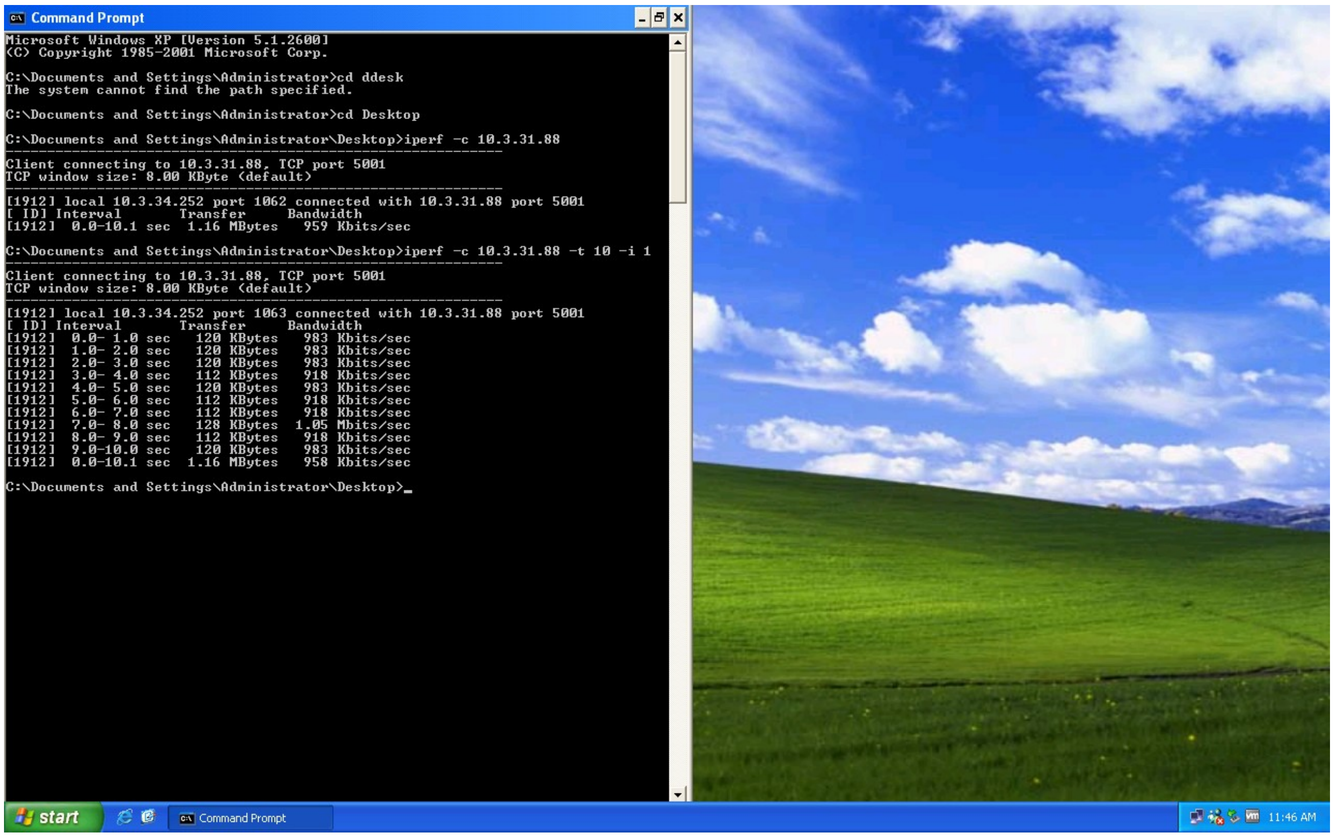
Ubuntu 5 (with 6Mbits limit) with IP address 10.3.34.11

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 50743 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  7.00 MBytes  5.85 Mbits/sec
```

Ubuntu 6 (with 3Mbits limit) with IP address 10.3.34.33

```
kanwar2@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 49585 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.8 sec  3.70 MBytes  2.88 Mbits/sec
```

Windows XP system is connected to the network and below is the diagram showing that as the default rule is 1Mbits/sec so it will get overall bandwidth no more than 1 mbps.

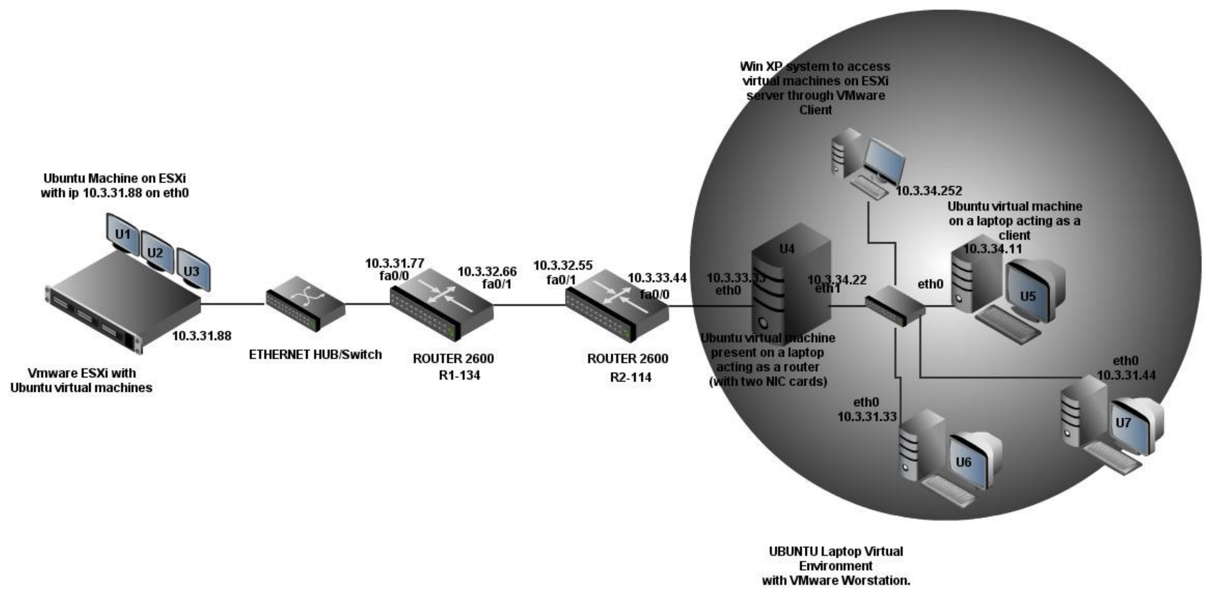


```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd ddesk
The system cannot find the path specified.

C:\Documents and Settings\Administrator>cd Desktop

C:\Documents and Settings\Administrator\Desktop>iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1912] local 10.3.34.252 port 1062 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[1912]  0.0-10.1 sec  1.16 MBytes   959 Kbits/sec

C:\Documents and Settings\Administrator\Desktop>iperf -c 10.3.31.88 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1912] local 10.3.34.252 port 1063 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[1912]  0.0- 1.0 sec   120 KBytes   983 Kbits/sec
[1912]  1.0- 2.0 sec   120 KBytes   983 Kbits/sec
[1912]  2.0- 3.0 sec   120 KBytes   983 Kbits/sec
[1912]  3.0- 4.0 sec   112 KBytes   918 Kbits/sec
[1912]  4.0- 5.0 sec   120 KBytes   983 Kbits/sec
[1912]  5.0- 6.0 sec   112 KBytes   918 Kbits/sec
[1912]  6.0- 7.0 sec   112 KBytes   918 Kbits/sec
[1912]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[1912]  8.0- 9.0 sec   112 KBytes   918 Kbits/sec
[1912]  9.0-10.0 sec   120 KBytes   983 Kbits/sec
[1912]  0.0-10.1 sec  1.16 MBytes   958 Kbits/sec

C:\Documents and Settings\Administrator\Desktop>_
```

Above configuration show us how to implement bandwidth limitations for particular users in a network.
Filters can be applied to for particular applications using port numbers and latency sensitive traffic like video streaming can also be preferred using port numbers.

## 8.2 With Routers:

Now we can test this by introducing two routers in between the server and clients, this will add some latency to the network:



Below are few Iperf test results performed on the network:

TCP test:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 42053 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  53.5 MBytes  44.9 Mbits/sec
```

Iperf parallel tests: two iperf test sessions were run parallel to each other using -P attribute.

```
kanwar2@ubuntu:~$ iperf -c 10.3.31.88 -P 2
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 48322 connected with 10.3.31.88 port 5001
[  4] local 10.3.34.33 port 48323 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  33.4 MBytes  28.0 Mbits/sec
[  3]  0.0-10.0 sec  24.9 MBytes  20.8 Mbits/sec
[SUM]  0.0-10.0 sec  58.2 MBytes  48.7 Mbits/sec
```

rtt- ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=1.87 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=1.98 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=2.43 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=1.97 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.95 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=1.86 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=1.76 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=1.70 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=1.77 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=2.03 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=61 time=1.90 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=61 time=2.26 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=61 time=1.89 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=61 time=1.84 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=61 time=1.91 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=61 time=1.88 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=61 time=1.85 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=61 time=1.98 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=61 time=1.60 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=61 time=1.60 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19039ms
rtt min/avg/max/mdev = 1.600/1.905/2.431/0.194 ms
```

Routers are configured with static routing as shown in first experiment.
Connectivity is complete.
So conf on Ubuntu Router to limit bandwidth of different users:
Ubuntu 5(10.3.34.11):  10 Mbits/sec
Ubuntu 6(10.3.34.33):  5 Mbits/sec
Any other IP address :  3 Mbits/sec

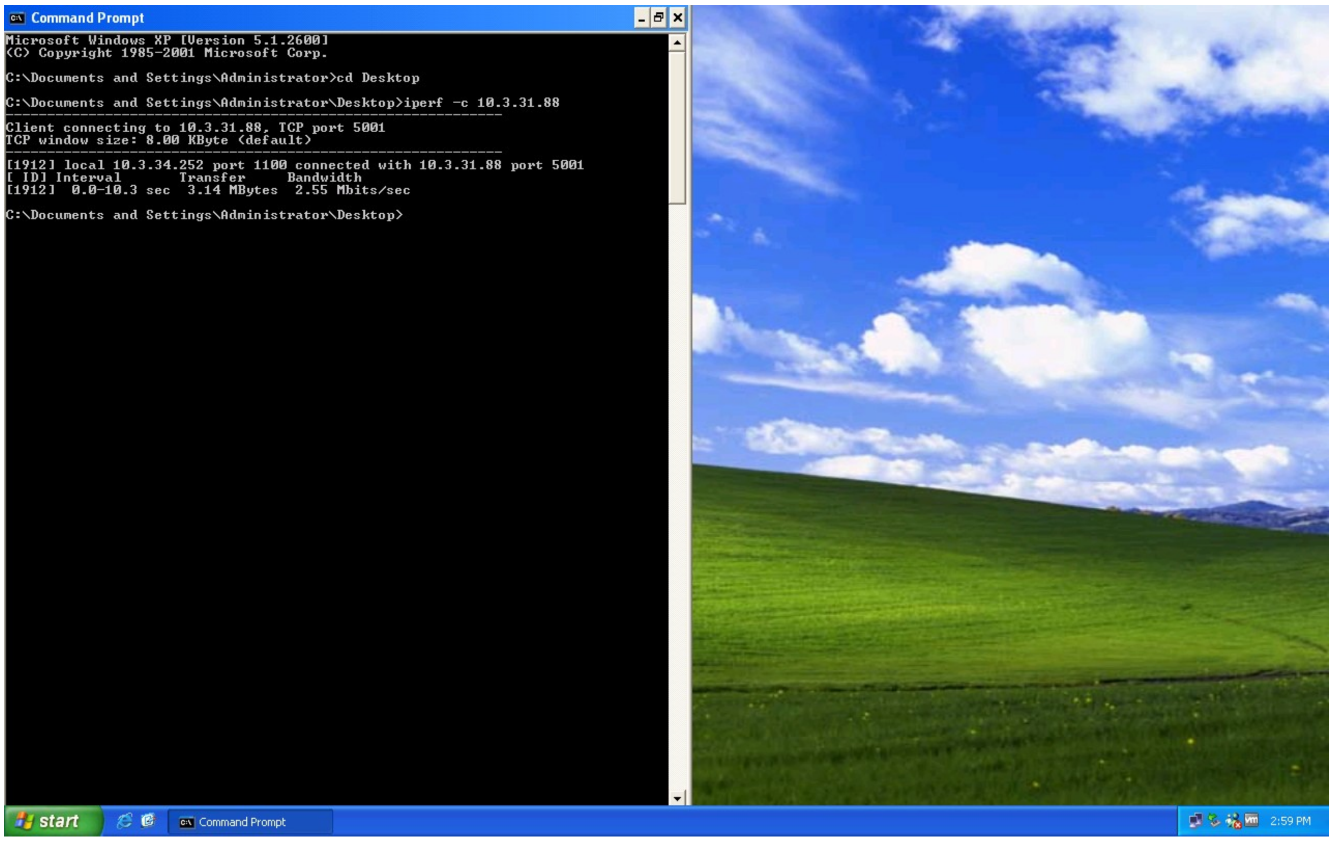
```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 18mbit burst 15k
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 10mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 5mbit ceil 5mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 3mbit ceil 3mbit
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: pfifo limit 20
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 10.3.34.11 flowid
1:10
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 10.3.34.33 flowid
1:20
```

Iperf tests showing limited bandwidths for users and default also:
Ubuntu U5 (10.3.34.11)

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 54798 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  11.4 MBytes  9.46 Mbits/sec
```

Ubuntu U6 (10.3.34.33):

```
kanwar2@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 40208 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  5.86 MBytes  4.87 Mbits/sec
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd Desktop

C:\Documents and Settings\Administrator\Desktop>iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1912] local 10.3.34.252 port 1100 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[1912]  0.0-10.3 sec  3.14 MBytes  2.55 Mbits/sec

C:\Documents and Settings\Administrator\Desktop>
```

The above configuration gives a ceiling value to each user; user can't exceed that ceiling value.
But if more bandwidth is available with the network then the traffic can get more bandwidth up to the ceiling value.
i.e. we can limit the minimum value for a user but if more bandwidth is available on the network then the user can get that value too.

In our environment we are having around 45Mbits/sec bandwidth for our network. Now we will give min value for two users and if more is available they can also utilize that bandwidth:

Configuration for the scenario is:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 40mbit burst 15k
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 10mbit ceil 15mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 5mbit ceil 10mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 3mbit ceil 5mbit
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: pfifo limit 20
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 10.3.34.11 flowid
1:10
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip src 10.3.34.33 flowid
1:20
```

Iperf test on Ubuntu with ip add 10.3.34.33: limit for this is 5mbit but if available it can utilize up to 10mbits.sec according to ceiling value.

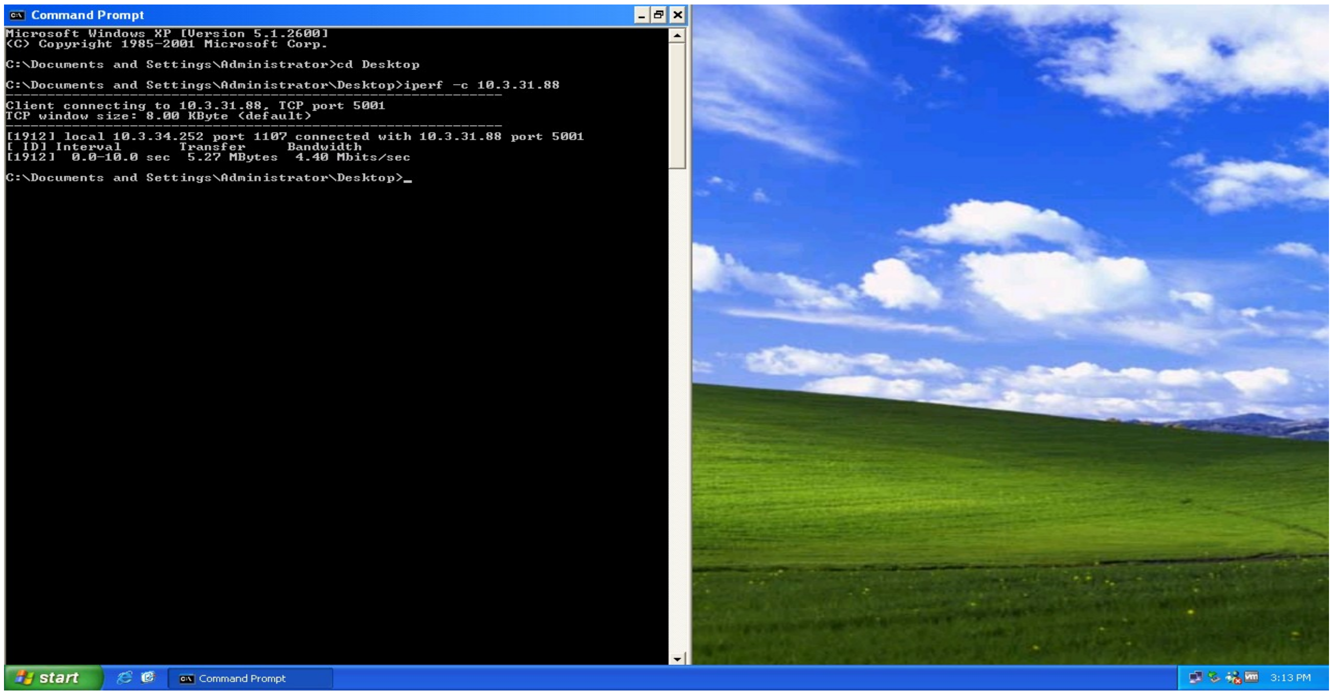
```
kanwar2@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 51567 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  11.6 MBytes  9.66 Mbits/sec
```

Iperf test of ubuntu with ip 10.3.31.11:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size:   493 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 39183 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  16.6 MBytes  13.9 Mbits/sec
```

Above two results prove the ceil value test, as sufficient amount of bandwidth is available on the network, so the router will give them bandwidth upto their ceiling values.

Other clients that will follow the default rule:



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd Desktop

C:\Documents and Settings\Administrator\Desktop>iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1912] local 10.3.34.252 port 1107 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[1912]  0.0-10.0 sec  5.27 MBytes  4.40 Mbits/sec

C:\Documents and Settings\Administrator\Desktop>_
```

Show commands on Ubuntu router U4:

```
root@ubuntu:/# tc -s -d qdisc show dev eth0
qdisc htb 1: root refcnt 2 r2q 10 default 30 direct_packets_stat 0 ver 3.17
 Sent 41903988 bytes 28224 pkt (dropped 17, overlimits 57515 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 10: parent 1:10 limit 127p quantum 1514b flows 127/1024 perturb 10sec
 Sent 17329662 bytes 11451 pkt (dropped 7, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 20: parent 1:20 limit 127p quantum 1514b flows 127/1024
 Sent 12625744 bytes 8344 pkt (dropped 10, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc pfifo 30: parent 1:30 limit 20p
 Sent 11948582 bytes 8429 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0

root@ubuntu:/# tc -s -d class show dev eth0
class htb 1:1 root rate 40000Kbit ceil 40000Kbit burst 15Kb/8 mpu 0b overhead 0b cburst 1600b/8 mpu 0b overhead 0b level 7
 Sent 41903988 bytes 28224 pkt (dropped 0, overlimits 0 requeues 0)
 rate 40bit 0pps backlog 0b 0p requeues 0
 lended: 10974 borrowed: 0 giants: 0
 tokens: 47829 ctokens: 4829

class htb 1:10 parent 1:1 leaf 10: prio 0 quantum 125000 rate 10000Kbit ceil 15000Kbit burst 1600b/8 mpu 0b overhead 0b cburst 1597b/8 mpu 0b overhead 0b level 0
 Sent 17329662 bytes 11451 pkt (dropped 7, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 7930 borrowed: 3521 giants: 0
 tokens: 19110 ctokens: 12735
```

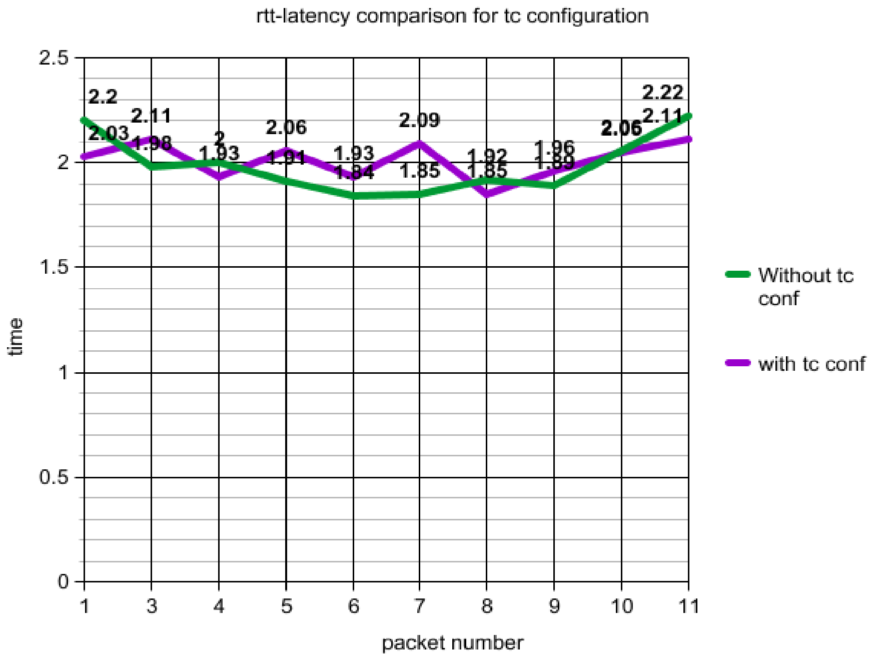
```
class htb 1:20 parent 1:1 leaf 20: prio 0 quantum 62500 rate 5000Kbit ceil 10000Kbit burst 1600b/8
mpu 0b overhead 0b cburst 1600b/8 mpu 0b overhead 0b level 0
 Sent 12625744 bytes 8344 pkt (dropped 10, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 4210 borrowed: 4134 giants: 0
 tokens: 38204 ctokens: 19110

class htb 1:30 parent 1:1 leaf 30: prio 0 quantum 37500 rate 3000Kbit ceil 5000Kbit burst 1599b/8 mpu
0b overhead 0b cburst 1600b/8 mpu 0b overhead 0b level 0
 Sent 11948582 bytes 8429 pkt (dropped 0, overlimits 0 requeues 0)
 rate 40bit 0pps backlog 0b 0p requeues 0
 lended: 5110 borrowed: 3319 giants: 0
 tokens: 64328 ctokens: 38610

root@ubuntu:/# tc -s -d filter show dev eth0
filter parent 1: protocol ip pref 1 u32
filter parent 1: protocol ip pref 1 u32 fh 800: ht divisor 1
filter parent 1: protocol ip pref 1 u32 fh 800::800 order 2048 key ht 800 bkt 0 flowid 1:10
  match 0a03220b/ffffffff at 12
filter parent 1: protocol ip pref 1 u32 fh 800::801 order 2049 key ht 800 bkt 0 flowid 1:20
  match 0a032221/ffffffff at 12
```

Test to check latency introduced by Traffic control configuration:

rtt-ping test for network with routers but no tc configuration is done:

```
kanwar2@ubuntu:~$ ping 10.3.31.88 -c 10
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=2.20 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=1.98 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=2.00 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=1.91 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.84 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=1.85 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=1.92 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=1.89 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=2.06 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=2.22 ms

--- 10.3.31.88 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9022ms
rtt min/avg/max/mdev = 1.844/1.992/2.222/0.129 ms

kanwar2@ubuntu:~$ ping 10.3.31.88 -c 10
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=2.03 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=2.11 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=1.93 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=2.06 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.93 ms
```

```
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=2.09 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=1.85 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=1.96 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=2.05 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=2.11 ms

--- 10.3.31.88 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9019ms
rtt min/avg/max/mdev = 1.854/2.016/2.116/0.098 ms
```



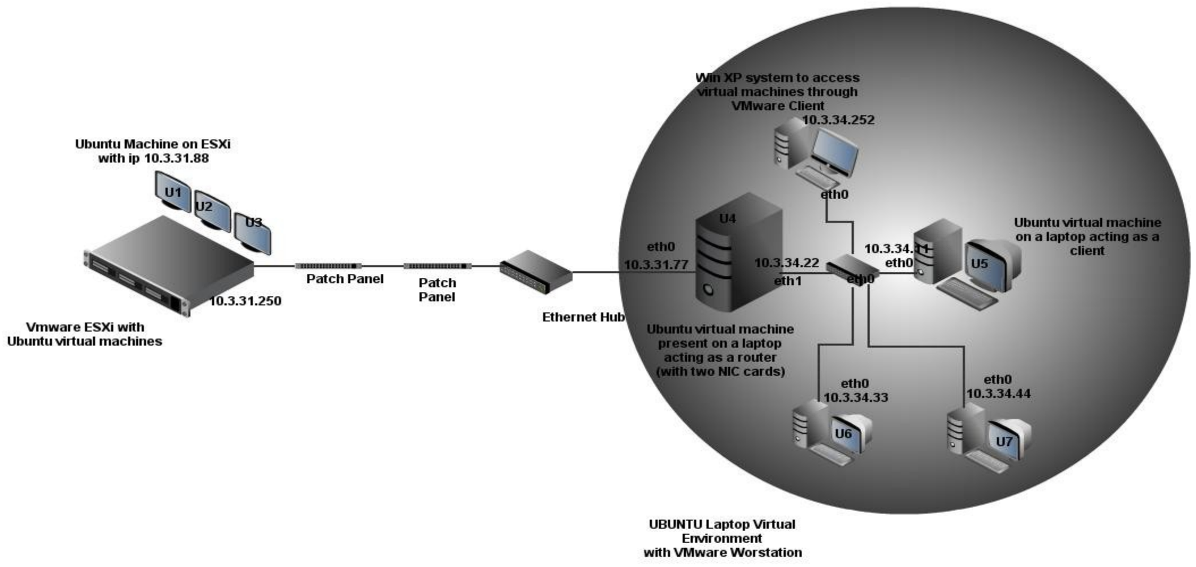
The mean value shows us a slight increase in rtt-latency for tc configuration scenario.

## 9. Limit bandwidth for particular applications:

To limit bandwidth for various applications port number of those applications must be used, then those port numbers can be used in iperf to check bandwidth for the applications and to check the traffic control configuration.

We can use the default port number 5001 in configuration that is used by iperf and will use another port for another application, let it be 5002.

### 9.1 Scenario without routers:


Ubuntu Machine on ESXi with ip 10.3.31.88
U1
U2
U3
10.3.31.250
Vmware ESXi with Ubuntu virtual machines
Patch Panel
Patch Panel
Ethernet Hub
Win XP system to access virtual machines through VMware Client
10.3.34.252
eth0
U4
eth0
10.3.31.77
10.3.34.22
eth1
eth0
10.3.34.11
eth0
U5
Ubuntu virtual machine on a laptop acting as a client
Ubuntu virtual machine present on a laptop acting as a router (with two NIC cards)
eth0
10.3.34.33
U6
eth0
10.3.34.44
U7
UBUNTU Laptop Virtual Environment with VMware Worstation

As for a application on the network there is always a port number associated; and the packets on the network have the port number information for that application. So to limit the bandwidth for a known application, we can put traffic control filters with respect to these port numbers.

Default configuration i.e. No traffic control configured:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 60374 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  85.0 MBytes  71.2 Mbits/sec
```

rtt-latency test without configuration done:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 10
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.62 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.11 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.23 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.24 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.26 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.15 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.21 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.19 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.37 ms

--- 10.3.31.88 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9019ms
rtt min/avg/max/mdev = 1.112/1.281/1.621/0.142 ms
```

Configuration on Ubuntu router U4 for limiting bandwidth for applications:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 10mbit burst 15k
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 6mbit ceil 6mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 3mbit ceil 3mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1mbit ceil 1mbit
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: pfifo limit 20
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 5001 0xffff
flowid 1:10
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 5002 0xffff
flowid 1:20
```

In the above configuration we have added filters in different classes to filter traffic based on the port number. All applications are based on port number in networking. So applying filter on port numbers can help us to control traffic for port numbers. The filter at the exit interface of the Ubuntu router will look for destination port number and will take appropriate action according to the filters defined on it.
Whenever there is a packet on eth0 interface to be sent out, it will first check for the filters, if there is a specific filter for that traffic then it will be sent to that particular class, else default filter will work on it and send it to the default class. These classes will define the bandwidth for that traffic.
Further one can implement any type of queue discipline on any type of traffic as shown in the configuration above.

We used two filters in above configuration to send any traffic for port number 5001 and 5002 to classes 10 and 20 respectively with bandwidth specified.
So whenever any traffic for port number 5001 or 5002 is found on the interface it will not take bandwidth more than the specified values.
The readings below for iperf test on three different PC's to connect to the server using iperf with port 5001, 5002 and 5003:

Iperf test results:
For port# 5001:
On iperf server: *iperf -s*

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 52723 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  7.11 MBytes  5.89 Mbits/sec
```

For port# 5002: We first need to run this command on server to communicate through port 5002:
*iperf -s -p 5002*

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5002
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5002
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 45224 connected with 10.3.31.88 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  3.46 MBytes  2.89 Mbits/sec
```

For any other application that has a different port associated except defined ones(let it be 5003)
We first need to run this command on server: iperf -s -p 5003

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5003
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5003
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 46538 connected with 10.3.31.88 port 5003
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  1.20 MBytes    983 Kbits/sec
```

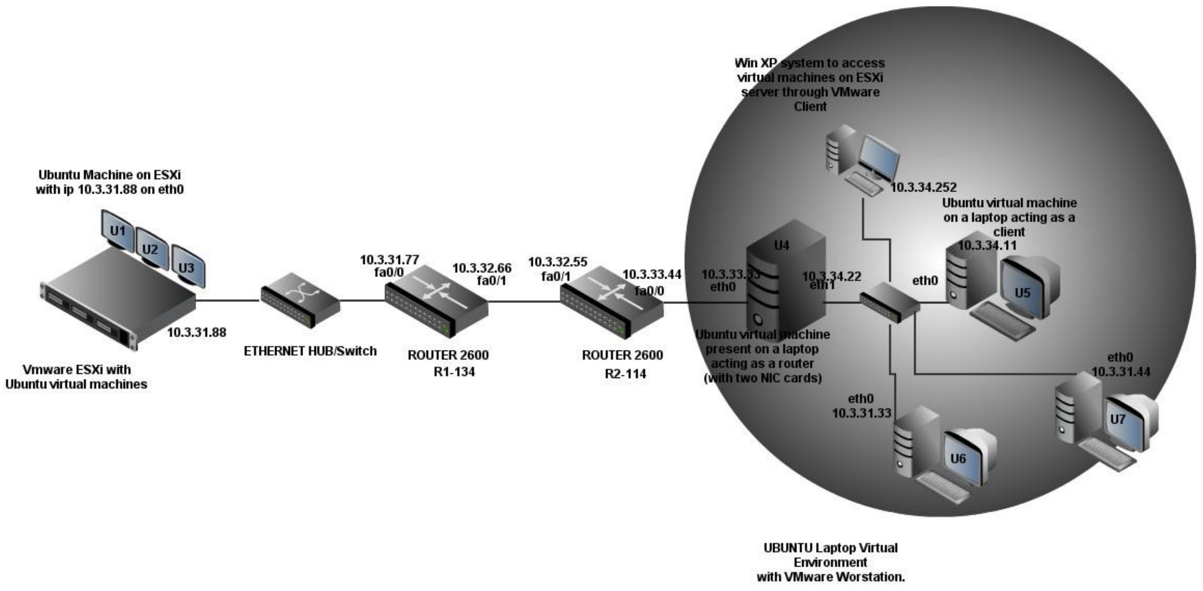
So the above configuration is working for port numbers. Hence we can restrict or control bandwidth for various applications by applying filters on port numbers.

rtt-latency test with configuration:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 10
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.51 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.33 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.18 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.43 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.24 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.29 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.27 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.25 ms
```

```
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.35 ms

--- 10.3.31.88 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9018ms
rtt min/avg/max/mdev = 1.189/1.329/1.512/0.096 ms
```

## 9.2 Scenario with Routers:


Ubuntu Machine on ESXi with ip 10.3.31.88 on eth0
U1
U2
U3
10.3.31.88
Vmware ESXi with Ubuntu virtual machines
ETHERNET HUB/Switch
10.3.31.77 fa0/0
10.3.32.66 fa0/1
ROUTER 2600 R1-134
10.3.32.55 fa0/1
10.3.33.44 fa0/0
ROUTER 2600 R2-114
Win XP system to access virtual machines on ESXi server through VMware Client
10.3.34.252
U4
10.3.33.33 eth0
10.3.34.22 eth1
Ubuntu virtual machine present on a laptop acting as a router (with two NIC cards)
Ubuntu virtual machine on a laptop acting as a client
10.3.34.11
eth0
U5
eth0 10.3.31.44
U7
eth0 10.3.31.33
U6
UBUNTU Laptop Virtual Environment with VMware Worstation.

Configuration on Ubuntu router U4:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 10mbit burst 15k
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 6mbit ceil 6mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 3mbit ceil 3mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1mbit ceil 1mbit
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: pfifo limit 20
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 5001 0xffff
flowid 1:10
root@ubuntu:/# tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 5002 0xffff
flowid 1:20
```

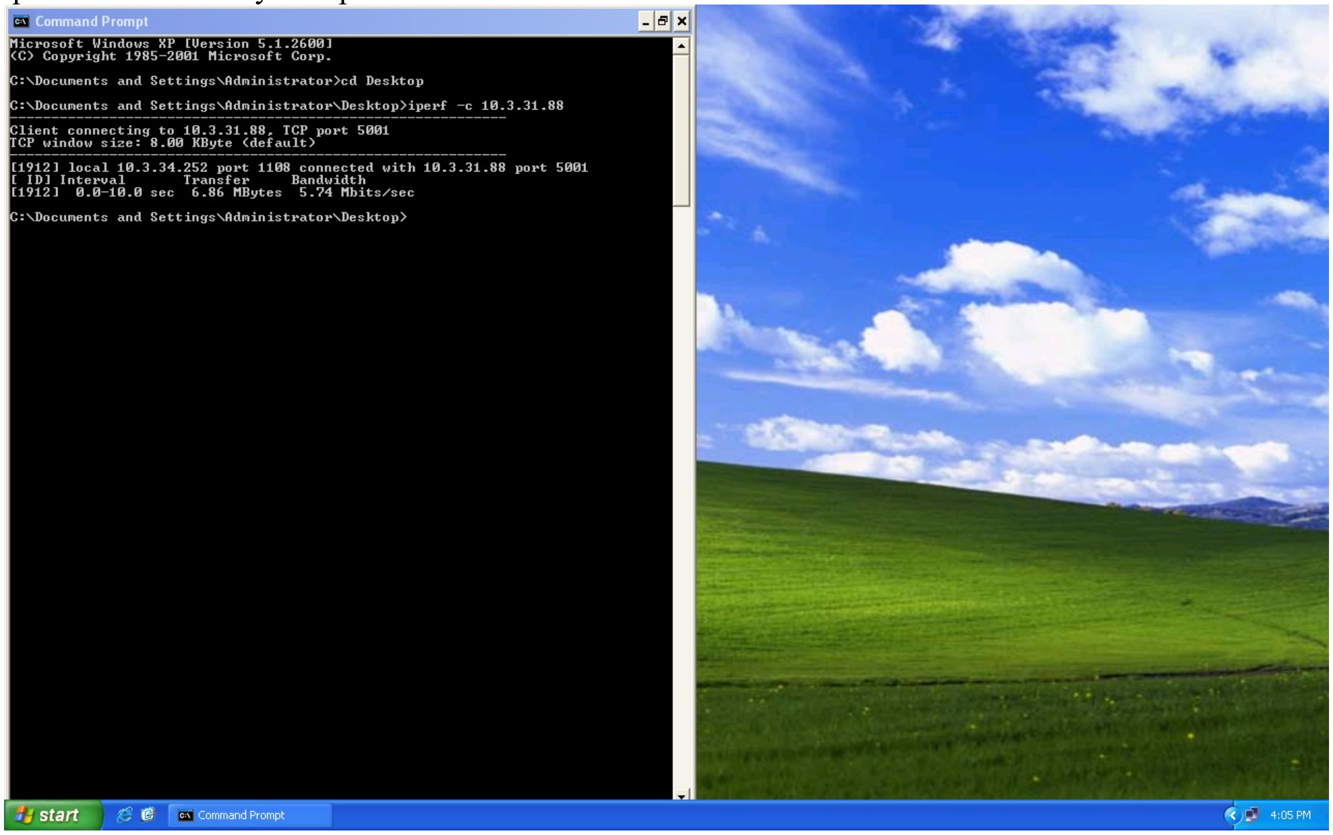
Ubuntu 6:

```
kanwar2@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 35292 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  7.16 MBytes  5.92 Mbits/sec
```

Ubuntu 5:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 35375 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  7.09 MBytes  5.90 Mbits/sec
```

Iperf test from XP system present in network:



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd Desktop

C:\Documents and Settings\Administrator\Desktop>iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1912] local 10.3.34.252 port 1108 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[1912]  0.0-10.0 sec  6.86 MBytes  5.74 Mbits/sec

C:\Documents and Settings\Administrator\Desktop>
```

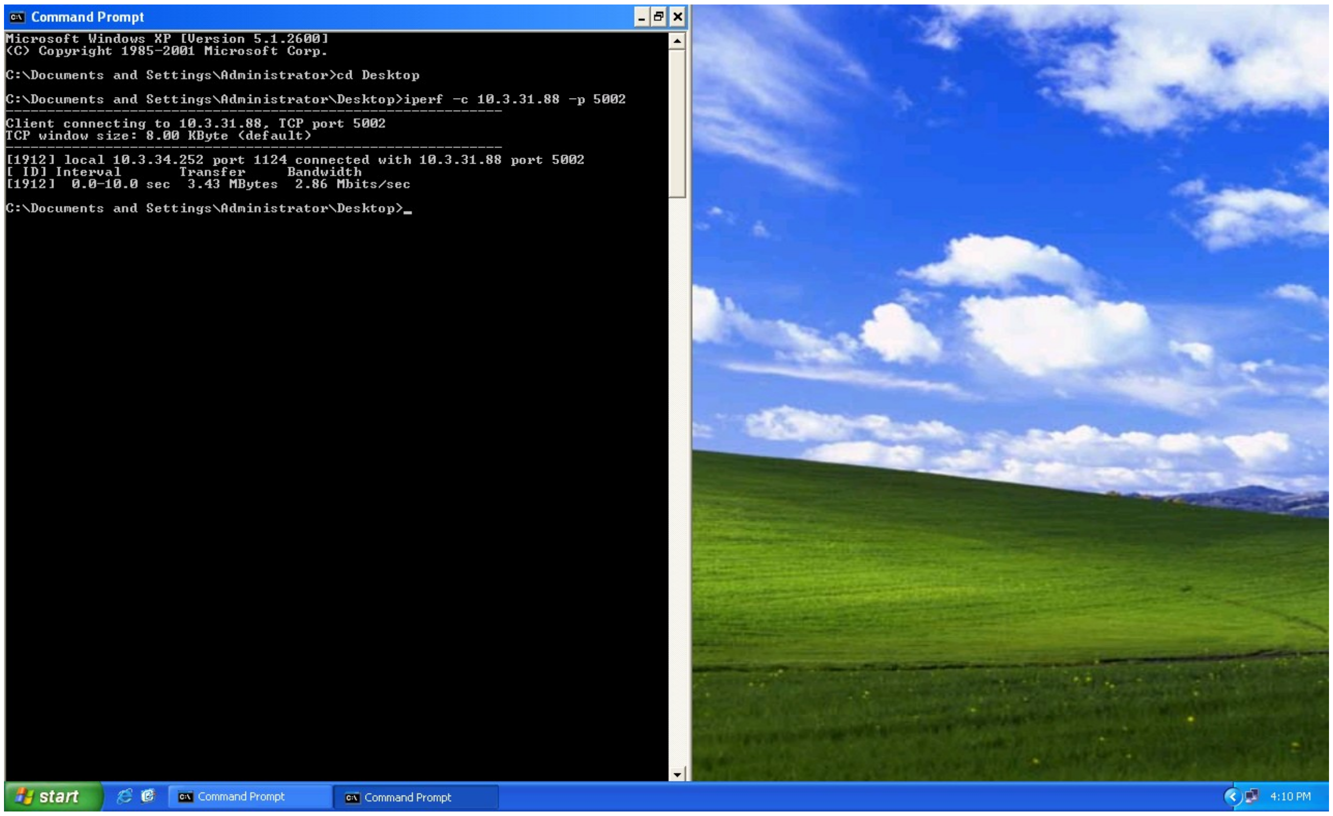
Three tests for port number 5001 from three different locations through Ubuntu router U4 proves that the bandwidth for any application to port number 5001 can be given desired bandwidth.

Test for port #5002 for which we have set the bandwidth to 3Mbits/sec:

On Ubuntu 1 on ESXi server we will start iperf server with port 5002 for communication:

```
Command given on Ubuntu 1: iperf -s -p 5002
Ubuntu 5:
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5002
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5002
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 52726 connected with 10.3.31.88 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  3.51 MBytes  2.92 Mbits/sec
Ubuntu 6:
kanwar2@ubuntu:~$ iperf -c 10.3.31.88 -p 5002
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5002
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 35580 connected with 10.3.31.88 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.3 sec  3.66 MBytes  2.98 Mbits/sec
```

Win XP:



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd Desktop

C:\Documents and Settings\Administrator\Desktop>iperf -c 10.3.31.88 -p 5002
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5002
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1912] local 10.3.34.252 port 1124 connected with 10.3.31.88 port 5002
[ ID] Interval       Transfer     Bandwidth
[1912]  0.0-10.0 sec  3.43 MBytes  2.86 Mbits/sec

C:\Documents and Settings\Administrator\Desktop>_
```

So all the above three test shows us how to limit bandwidth for a particular application.
This setup can also be used to prefer any latency sensitive data using port numbers.
Just like for vlc video streaming we can define a port number and that any traffic for port number can be given any desired bandwidth but setting priority for that traffic will make sure that particular data should be preferred all the times. (That experiment is done in last part: TEST 4)

To check for the default traffic we will open port number 5005 on server and try to check the bandwidth from clients:
We have set the rate to be 1Mbits/sec:

Ubuntu 1:
```
iperf -s -p 5005
```
Ubuntu 5:
```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5005
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5005
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 53935 connected with 10.3.31.88 port 5005
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  1.20 MBytes    988 Kbits/sec
```

Ubuntu 6:
```
kanwar2@ubuntu:~$ iperf -c 10.3.31.88 -p 5005
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5005
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.33 port 37217 connected with 10.3.31.88 port 5005
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.9 sec  1.16 MBytes    892 Kbits/sec
```

So above iperf test prove the configuration is correct and we can limit bandwidth for various applications using their port numbers.

Round trip latency for the network with tc configuration:
```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=2.13 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=2.06 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=2.06 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=2.32 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=2.13 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=2.13 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=2.16 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=2.15 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=1.83 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=2.15 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=61 time=1.76 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=61 time=2.12 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=61 time=2.23 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=61 time=1.71 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=61 time=1.97 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=61 time=2.01 ms
```

```
64 bytes from 10.3.31.88: icmp_req=17 ttl=61 time=2.09 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=61 time=2.19 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=61 time=2.13 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=61 time=2.03 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19033ms
rtt min/avg/max/mdev = 1.717/2.072/2.323/0.158 ms
```

rtt- latency for network without tc configuration:

```
kanwar2@ubuntu:~$ ping 10.3.31.88 -c 20
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=1.99 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=1.93 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=2.00 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=1.95 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.71 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=1.74 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=2.04 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=1.98 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=1.84 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=1.79 ms
64 bytes from 10.3.31.88: icmp_req=11 ttl=61 time=1.99 ms
64 bytes from 10.3.31.88: icmp_req=12 ttl=61 time=1.70 ms
64 bytes from 10.3.31.88: icmp_req=13 ttl=61 time=2.09 ms
64 bytes from 10.3.31.88: icmp_req=14 ttl=61 time=2.04 ms
64 bytes from 10.3.31.88: icmp_req=15 ttl=61 time=2.28 ms
64 bytes from 10.3.31.88: icmp_req=16 ttl=61 time=2.03 ms
64 bytes from 10.3.31.88: icmp_req=17 ttl=61 time=2.04 ms
64 bytes from 10.3.31.88: icmp_req=18 ttl=61 time=2.07 ms
64 bytes from 10.3.31.88: icmp_req=19 ttl=61 time=2.15 ms
64 bytes from 10.3.31.88: icmp_req=20 ttl=61 time=1.62 ms

--- 10.3.31.88 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19040ms
rtt min/avg/max/mdev = 1.624/1.953/2.286/0.170 ms
```

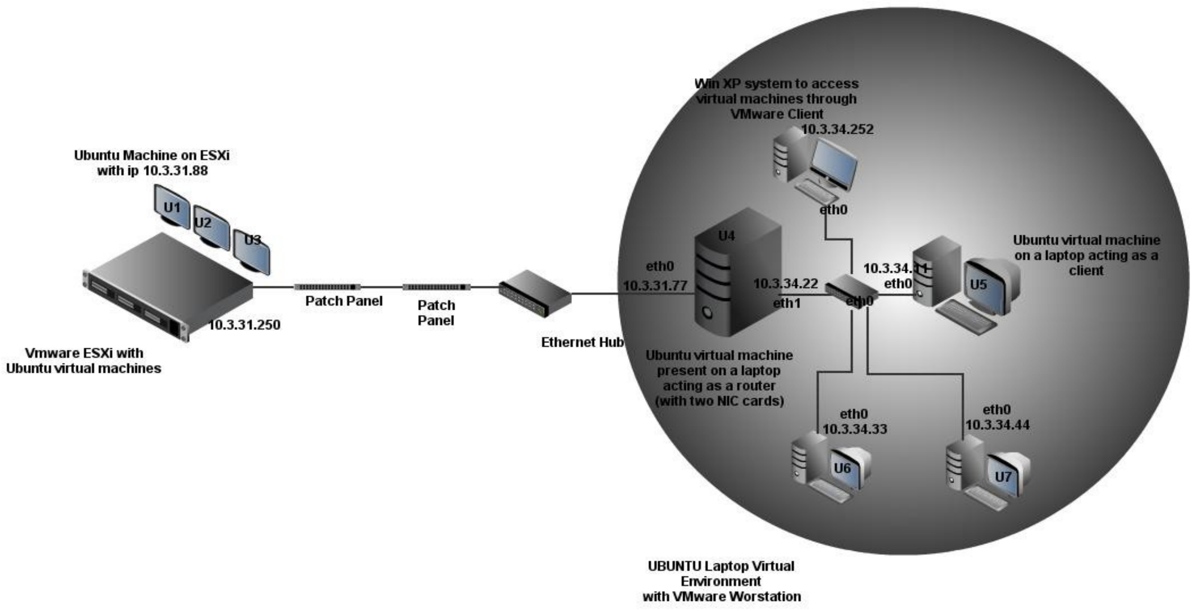
The configuration adds very less latency to the network, it does not utilize too much CPU.

## 10. Prefer latency sensitive data

There is a requirement for preferring latency sensitive data in various network environments.
In Linux traffic control it can be done in two ways:

One way is to match ip ports in the packets and apply traffic rules using htb. But it wont give any traffic priority over the other, it will work only with defined rates for particular traffic.
Other way is to mark the packets for traffic using iptables on the incoming port of the ubuntu router so that they can be filtered on the exit port and given desired bandwidth through the rules in htb (using prio) on the exit interface for those requests and using prio in traffic control configuration.

### 10.1 Scenario without routers:



We always get a bandwidth of around 71 Mbits/sec in direct connections without routers.
Ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 10
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=1.53 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.68 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.39 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.50 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.29 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=0.919 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.28 ms
```

```
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.43 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.47 ms

--- 10.3.31.88 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9018ms
rtt min/avg/max/mdev = 0.919/1.386/1.683/0.196 ms
```

Configuration on the Ubuntu router:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 15mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 6mbit ceil 6mbit prio 0
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 4mbit ceil 8mbit prio 1
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1mbit ceil 1mbit prio 2
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: pfifo limit 20
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 10 fw flowid 1:10
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 20 fw flowid 1:20
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 30 fw flowid 1:30
root@ubuntu:/# iptables -t mangle -N INSHAPER
root@ubuntu:/# iptables -t mangle -I PREROUTING -i eth1 -j INSHAPER
root@ubuntu:/# iptables -t mangle -A INSHAPER -p tcp --dport 5001 -j MARK --set-mark 10
root@ubuntu:/# iptables -t mangle -A INSHAPER -p tcp --dport 5002 -j MARK --set-mark 20
root@ubuntu:/# iptables -t mangle -A INSHAPER -m mark --mark 0 -j MARK --set-mark 30
```

Iptables were used at the incoming interface of the Ubuntu routers to add or set mark on packets of a particular traffic, so that when it reaches the exit port of the router they should match this mark with filters to get desired bandwidth and with desired priority. (0 is highest priority and 2 is least priority.)

In the above configuration whenever any traffic(iperf in our case) for any port comes to eth1 it will mark it with 10, 20 or 30 according to the destination ports numbers: 5001, 5002 and app other ports ports respectively.
When these market packets are to be sent out of eth0 traffic control configuration comes into play. Firstly, filters are matched for and the traffic is directed to the corresponding class according to the fwmark set on the packets on interface eth1. So now the class decides about the bandwidth available for that application as defined by the user.
Here we have a default priority class to be 1:30 so any unmarked packets will be sent through this class and for fwmark as 10 and 20 to class 1:10 & 1:20 respectively.

We used ceiling for 5002 port to be 8 Mbits/sec, which it can achieve if free bandwidth is available on the network.

ping test:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 10
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=63 time=3.55 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=63 time=1.66 ms
```

```
64 bytes from 10.3.31.88: icmp_req=3 ttl=63 time=1.29 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=63 time=1.54 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=63 time=1.32 ms
64 bytes from 10.3.31.88: icmp_req=6 ttl=63 time=1.33 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=63 time=1.40 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=63 time=1.30 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=63 time=1.36 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=63 time=1.42 ms

--- 10.3.31.88 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9020ms
rtt min/avg/max/mdev = 1.296/1.621/3.553/0.653 ms
```

We can implement various queue disciplines on various classes as shown in the above configuration.
Iperf test results:

For port# 5001:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 55431 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  7.00 MBytes  5.83 Mbits/sec
```

For port# 5002:

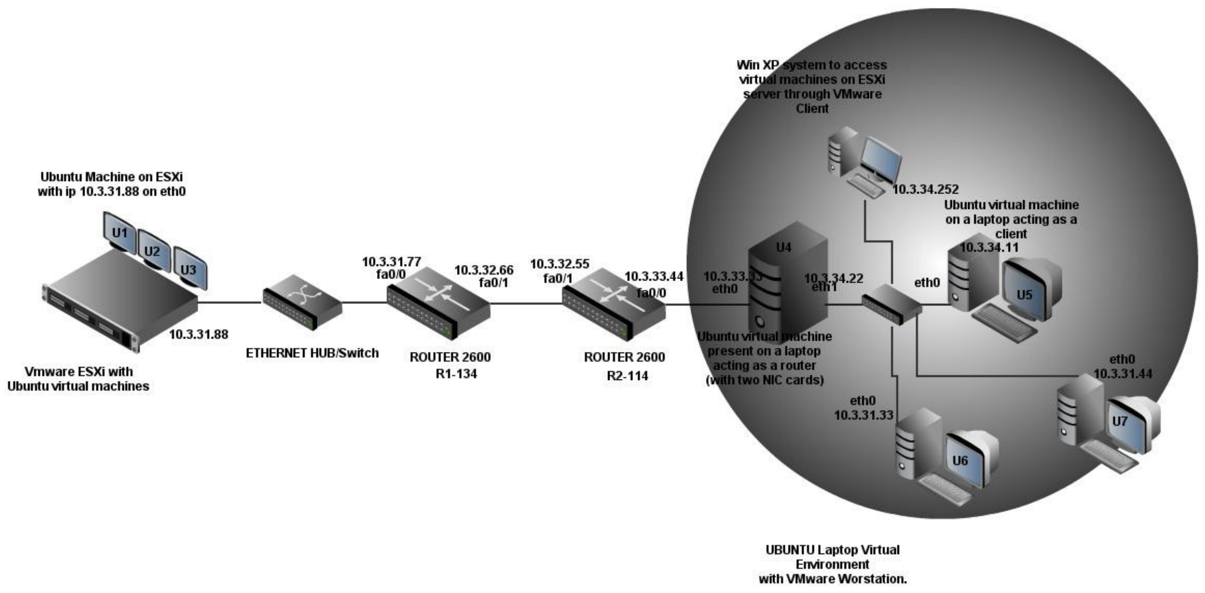
```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5002
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5002
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 34352 connected with 10.3.31.88 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  9.13 MBytes  7.66 Mbits/sec
```

For port# 5003:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5003
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5003
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 57051 connected with 10.3.31.88 port 5003
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  1.18 MBytes    971 Kbits/sec
```

The above configuration is correct and is giving desired bandwidth to various applications but we still need to check the priority function, that whether it is preferring the data for port number 5001 over other ports or not. That test is done in the next part with routers.

## 10.2 Scenario with routers:



Adding routers will add some latency to the network:
iperf and ping test result using routers in the network and no configuration is done on the Ubuntu router.
Iperf test for the environment:
Ubuntu 4:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 37972 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  55.9 MBytes  46.8 Mbits/sec
```

Ping test for round trip latency:

```
ubuntu@ubuntu:~$ ping 10.3.31.88 -c 10
PING 10.3.31.88 (10.3.31.88) 56(84) bytes of data.
64 bytes from 10.3.31.88: icmp_req=1 ttl=61 time=1.55 ms
64 bytes from 10.3.31.88: icmp_req=2 ttl=61 time=1.84 ms
64 bytes from 10.3.31.88: icmp_req=3 ttl=61 time=1.73 ms
64 bytes from 10.3.31.88: icmp_req=4 ttl=61 time=2.03 ms
64 bytes from 10.3.31.88: icmp_req=5 ttl=61 time=1.83 ms
```

```
64 bytes from 10.3.31.88: icmp_req=6 ttl=61 time=1.96 ms
64 bytes from 10.3.31.88: icmp_req=7 ttl=61 time=1.88 ms
64 bytes from 10.3.31.88: icmp_req=8 ttl=61 time=1.94 ms
64 bytes from 10.3.31.88: icmp_req=9 ttl=61 time=2.14 ms
64 bytes from 10.3.31.88: icmp_req=10 ttl=61 time=1.71 ms

--- 10.3.31.88 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 1.553/1.865/2.148/0.169 ms
```

Now we will be configuring traffic control using htb with prio. We will mark packets on the incoming interface of the router i.e interface that is connected to the clients network 10.3.34.0/24.
These marked packets will be filtered at the exit interface of the router eth0 and will give priority to specific data according to the configuration.

Configuration on Ubuntu router will be as follows:

```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 15mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 6mbit ceil 6mbit prio 0
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 4mbit ceil 8mbit prio 1
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1mbit ceil 1mbit prio 2
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: pfifo limit 20
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 10 fw flowid 1:10
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 20 fw flowid 1:20
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 30 fw flowid 1:30
root@ubuntu:/# iptables -t mangle -N INSHAPER
root@ubuntu:/# iptables -t mangle -I PREROUTING -i eth1 -j INSHAPER
root@ubuntu:/# iptables -t mangle -A INSHAPER -p tcp --dport 5001 -j MARK --set-mark 10
root@ubuntu:/# iptables -t mangle -A INSHAPER -p tcp --dport 5002 -j MARK --set-mark 20
root@ubuntu:/# iptables -t mangle -A INSHAPER -m mark --mark 0 -j MARK --set-mark 30
```

For port 5001:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 34215 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  6.95 MBytes  5.79 Mbits/sec
```

For port 5002 :

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5002
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5002
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
```

```
[  3] local 10.3.34.11 port 57417 connected with 10.3.31.88 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  9.12 MBytes  7.64 Mbits/sec
```

For any other port or data that is not marked will be given least priority:

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5003
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5003
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 42913 connected with 10.3.31.88 port 5003
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  1.21 MBytes    999 Kbits/sec
```

Above Iperf test shows us the bandwidth selected for various application:

Run multiple iperf sessions on server to check which one is given more priority:
Change the configuration on the router as follows:

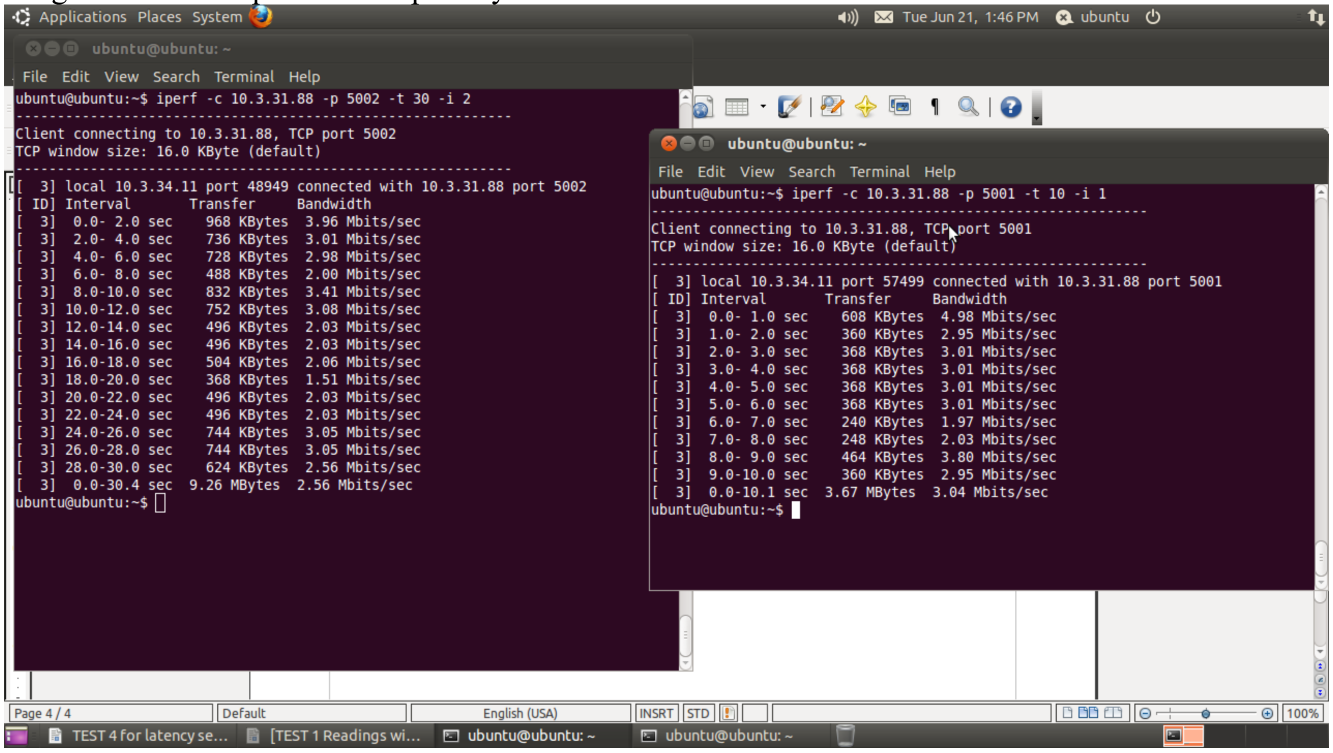
```
root@ubuntu:/# tc qdisc add dev eth0 root handle 1: htb default 30
root@ubuntu:/# tc class add dev eth0 parent 1: classid 1:1 htb rate 5mbit
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:10 htb rate 2mbit ceil 3mbit prio 0
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:20 htb rate 2mbit ceil 3mbit prio 1
root@ubuntu:/# tc class add dev eth0 parent 1:1 classid 1:30 htb rate 1mbit ceil 3mbit prio 2
root@ubuntu:/# tc qdisc add dev eth0 parent 1:10 handle 10: sfq
root@ubuntu:/# tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
root@ubuntu:/# tc qdisc add dev eth0 parent 1:30 handle 30: pfifo limit 20
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 10 fw flowid 1:10
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 20 fw flowid 1:20
root@ubuntu:/# tc filter add dev eth0 parent 1:0 prio 0 protocol ip handle 30 fw flowid 1:30
root@ubuntu:/# iptables -t mangle -N INSHAPER
root@ubuntu:/# iptables -t mangle -I PREROUTING -i eth1 -j INSHAPER
root@ubuntu:/# iptables -t mangle -A INSHAPER -p tcp --dport 5001 -j MARK --set-mark 10
root@ubuntu:/# iptables -t mangle -A INSHAPER -p tcp --dport 5002 -j MARK --set-mark 20
root@ubuntu:/# iptables -t mangle -A INSHAPER -m mark --mark 0 -j MARK --set-mark 30
```

The maximum bandwidth was kept to only 5Mbits/sec to confirm the priority function. Here the desired output should be like this:

Suppose if only class 1:20 i.e for port number 5002 traffic is on the network then our configuration should give 3Mbits/sec to that network, but as soon as class 1:10 traffic i.e for port number 5001 is there on the network the router should give priority to this traffic than for the traffic for port number 5002. Test below gives the proof for the priority implemented.
The conf states that class 10 and 20 both should have same bandwidth of 2Mbits/sec but if more bandwidth is available then the max value for both can reach up to 3Mbits/sec.

Now running iperf test for port 5001 and 5002 simultaneously from a client to the server will make clear which has more priority and which will be preferred:

Image below is the proof of the priority used in the test:



```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5002 -t 30 -i 2
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5002
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 48949 connected with 10.3.31.88 port 5002
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec    968 KBytes  3.96 Mbits/sec
[  3]  2.0- 4.0 sec    736 KBytes  3.01 Mbits/sec
[  3]  4.0- 6.0 sec    728 KBytes  2.98 Mbits/sec
[  3]  6.0- 8.0 sec    488 KBytes  2.00 Mbits/sec
[  3]  8.0-10.0 sec    832 KBytes  3.41 Mbits/sec
[  3] 10.0-12.0 sec    752 KBytes  3.08 Mbits/sec
[  3] 12.0-14.0 sec    496 KBytes  2.03 Mbits/sec
[  3] 14.0-16.0 sec    496 KBytes  2.03 Mbits/sec
[  3] 16.0-18.0 sec    504 KBytes  2.06 Mbits/sec
[  3] 18.0-20.0 sec    368 KBytes  1.51 Mbits/sec
[  3] 20.0-22.0 sec    496 KBytes  2.03 Mbits/sec
[  3] 22.0-24.0 sec    496 KBytes  2.03 Mbits/sec
[  3] 24.0-26.0 sec    744 KBytes  3.05 Mbits/sec
[  3] 26.0-28.0 sec    744 KBytes  3.05 Mbits/sec
[  3] 28.0-30.0 sec    624 KBytes  2.56 Mbits/sec
[  3]  0.0-30.4 sec  9.26 MBytes  2.56 Mbits/sec
ubuntu@ubuntu:~$
```

```
ubuntu@ubuntu:~$ iperf -c 10.3.31.88 -p 5001 -t 10 -i 1
------------------------------------------------------------
Client connecting to 10.3.31.88, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.3.34.11 port 57499 connected with 10.3.31.88 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec    608 KBytes  4.98 Mbits/sec
[  3]  1.0- 2.0 sec    360 KBytes  2.95 Mbits/sec
[  3]  2.0- 3.0 sec    368 KBytes  3.01 Mbits/sec
[  3]  3.0- 4.0 sec    368 KBytes  3.01 Mbits/sec
[  3]  4.0- 5.0 sec    368 KBytes  3.01 Mbits/sec
[  3]  5.0- 6.0 sec    368 KBytes  3.01 Mbits/sec
[  3]  6.0- 7.0 sec    240 KBytes  1.97 Mbits/sec
[  3]  7.0- 8.0 sec    248 KBytes  2.03 Mbits/sec
[  3]  8.0- 9.0 sec    464 KBytes  3.80 Mbits/sec
[  3]  9.0-10.0 sec    360 KBytes  2.95 Mbits/sec
[  3]  0.0-10.1 sec  3.67 MBytes  3.04 Mbits/sec
ubuntu@ubuntu:~$
```

The two iperf test run from a client to iperf server on different ports(5001 & 5002).
We used 5Mbits/sec in this scenario to test our priority setup, as when iperf test for both ports 5001 and 5002 will be running only one can have bandwidth of 3Mbits/sec and the other will have around 2Mbits/sec, the test is as follows:
Iperf test for port 5002 was run for 30 seconds, for the first 10 seconds we can see that port 5002 was given bandwidth around 3Mbits/sec i.e ceiling value, but after 10 sec when test for port 5001 was started bandwidth was taken away from 5002 port and given to test for 5001 port, and then again after 10 seconds i.e. around 20 seconds the test results shows that as soon as the iperf test for port 5001 was complete bandwidth to test for 5002 again increased.

This test shows how we can assign priority to some data using htb with prio and can use different queue disciplines in with each class as desired.

## 11.Conclusion:

All the experiments prove that if traffic control implemented in a network we can get desired applications to run successfully in the network. We can decide which traffic can be sent at what rate on the network. It helps us to prefer our sensitive traffic over other traffic.

As the IP networks are stateless networks i.e there is no connection state between two machines for transport of data. Stateless is in the fundamental strength of the IP networks but the disadvantage of this is there is lack of differentiation of the flow of traffic of various types. This property can be given to the network with the help of Traffic Control configuration. We can implement traffic control to limit bandwidth for the network, for particular users and even various services. The traffic for various services can be differentiated using traffic control.

Most of the times buying more bandwidth is considered a solution rather than implementing Traffic Control, but i think if implemented properly Traffic control can save lots of bandwidth and cost for an organization.

The main advantages of implementing Traffic Control:

- Prevents monopolization of bandwidth by a user or application.
- Proper use of network resources.
- Help to prefer latency sensitive data.
- Can be used to drop or prefer traffic from particular user.
- Can be used to prefer traffic from a particular user, helpful for network administrators who have to perform network setup and other function on the network. So their traffic should not be suppressed in the high traffic times.
- If traffic is utilized properly, there is no need to buy more bandwidth; will be more economical.

## REFERENCES

- A Practical Guide to Linux Traffic Control, Jason Boxman
- Differentiated Service on Linux HOWTO, Leonardo Balliache
- Linux Advanced Routing & Traffic Control HOWTO,  Bert Hubert
- Linux Ethernet-Howto, Paul Gortmaker
- Linux Networking HOWTO, Joshua Drake
- The Linux Networking Overview HOWTO By: Daniel Lopez Ridrujo
- The Linux Router By Kaleem Anwar, Muhammad Amir, Ahmad Saeed and Muhammad Imran
- Traffic Control HOWTO, Martin A. Brown
- Traffic Control using tcng and HTB HOWTO,  Martin A. Brown
- QoS Connection Tuning HOWTO by:Emmanuel Roger
- http://openmaniak.com/iperf.php
- http://www.vmware.com/pdf/GuestOS_guide.pdf
- https://help.ubuntu.com/community/Internet/ConnectionSharing
- https://help.ubuntu.com/community/Router
- http://www.rhyshaden.com/ipdgram.htm
- http://www.lesbell.com.au/Home.nsf/22a1497b7c7df99bca256d1700811540/c9d5888e0b0f9f3bca256f080010ff4e?OpenDocument
- http://www.linuxquestions.org