International Journal of Conceptions on Computing and Information Technology
Vol. 9, Issue 2, August' 2024; ISSN: 2345 - 9808

Prioritization scheme as Panacea for good network performance

Munir Kolapo Yahya-Imam and Prof. Sellapan Palaniappan
School of Science and Engineering,
Malaysia University of Science and Technology,
Selangor, Malaysia.
mk.yahya2007@yahoo.com and sell@must.edu.my

Dr. Por Lip Yee
Dept. of Computer System & Technology,
University of Malaya,
Kuala Lumpur, Malaysia.
porlip@um.edu.my

Abstract— **Bandwidth management in communication is an important concern in academic institution with bandwidth costs as high as in some developing countries. ISPs and businesses cannot afford to shove a router on their network, connect it to the Internet and hope for high quality of service.**

This paper presents the design and implementation of a prioritization scheme on a LAN network by combining several bandwidth management tools. Specifically, it manages the limited bandwidth resources that are available in the most efficient way using CBQ (Class Based Queuing), SQUID delay pools with iptable as network optimizer.

Keywords- Bandwidth management; Prioritization scheme; Class Based Queuing; Squid Delay pools; University Network performance; iptable.

I. INTRODUCTION

Bandwidth can be described as the amount of data that can be transmitted in a fixed amount of time, expressed in kilobits per second "Ref. [1]". Today the network traffic in institutions of higher learning has greatly increased and this is mainly due to increase in the number of users which results in bandwidth congestion and poor quality of service to end users. "Ref. [2]" "In computer networks, bandwidth is often used as a synonym for data transfer rate" i.e. the amount of data that can be carried from one node to another in a given time period, usually a second. It has been recognized that bandwidth is a valuable resource/asset, and therefore needs to be managed, conserved and shared effectively via innovative approaches.

For instance, bandwidth management in universities facilitate a robust campus network with a good connectivity over the Internet. Students and their parents consider good access to networked resources as a factor for their choice of institutions. At the same time, research demands are also growing. "Ref. [3]" Advanced research such as UCSC Genome Bioinformatics and Mars exploration makes use of a large network device for massive calculation which requires a huge amount of bandwidth. Nevertheless, it is often not practical to meet the increased demand for bandwidth by simply buying more. Peer-to-peer computing environment and applications such as Napster, Kazaa, Audio-galaxy and Gnutella result in greater demand for network bandwidth that most colleges and universities cannot afford. Each campus must decide when the cost of investing in bandwidth management strategy will cost less rather than buying more bandwidth. Investing in neither bandwidth management strategy nor more bandwidth is tantamount to leaving the campus network at risk of been hopelessly bogged down to the point where users are not well served.

One effective solution for these problems is to manage the existing university network bandwidth almost equally, using suitable queuing disciplines and filters that exist in Linux. It is a full-featured technology that reduces cost and improves network quality of service. "Ref. [4]" To ensure that your network users have access to critical applications and a good network quality of service will require knowledge of where and how your network bandwidth is been consume and the ability to set policies to prioritize it. "Ref. [5]" Bandwidth Prioritization lets you control network bandwidth to specific content categories and gives you understandable report on your network bandwidth usage so you can troubleshoot problems and limit or prioritize certain users or applications to maintain the highest level of a good network quality of service and performance.

II. LITERATURE SURVEY

Tertiary institutions are faced with major obstacles in their use of networked information resources "Ref. [6]". The price of bandwidth is disproportionately high, and it is costly and difficult to improve international network connectivity. Bandwidth management is a general term given as a collection of tools and techniques that an institution can use to reduce demand on critical segment of their network "Ref. [7]". In order to effectively manage a network connection of any size, you will need to take a multifaceted approach that includes effective network monitoring, a sensible policy that defines acceptable behavior, and a solid implementation that enforces these rules "Ref. [8]". Effective management and optimization of bandwidth are critical to research and education and there is urgent demand for training skills and knowledge developed within this area. Research has shown that majority of Tertiary Institutions undertake little or no monitoring or management of their bandwidth "Ref. [9]". The same research also recommended that improving bandwidth management is probably the easiest way for universities to improve the quality and quantity of their bandwidth for educational purposes. Moreover, internet connectivity and access to networked information resources are increasingly essential requirement for any research or educational institution and to achieve this, capacity development within the area of bandwidth management is an essential element. According to "unpublished" [10] bandwidth management and optimization activities are often not undertaken or when they are, they often face significant problems. There are number of contributing factors to this:

(i) Lack of information, skills, knowledge and actions at the technical level.

(ii) Lack of leadership and direction to help guide actions and policy development.

(iii) A non-supportive wider strategic and policy frame work within which the appropriate technical solution can be implemented.

III. System Design

Designing our new prioritization scheme will involve the combination of some of the existing bandwidth management tools as aforementioned. The first tool that will be use is CBQ. It is a queuing method that performs classification by port number or IP address, and controls the bandwidth usage of traffic in accordance with the pre-defined classes. The second tool is Squid which is a proxy server and web cache daemon "Ref. [11]". It has a wide variety of uses, in this paper squid will be used to fine tune the speed of traffics that passes through our network. Finally, iptable will be used to set priorities for traffics that will be passing through our network. This will be achieved with the help of a tool know as "iptables" which can be found inside our Linux kernel. Implementing our new bandwidth management configuration inside a Linux sever will highly improve the network quality of service and also the issue of insecurity will not be a threat on the university network because the entire network bandwidth management design and configuration will be done in a command line mode which will make it difficult to hacked into by intruders. The entire network bandwidth will be shared at two levels inside our main network server. Any network traffic that falls into any of these levels will be limit to a specified transmitting speed that is assigned to that level.

A. The Input Design

The input of the bandwidth optimizer is shown in Fig 1.1 it consists of the PREROUTING chain and the FORWARD chain. The packets are inspected and a routing decision is made based on the MARK type set by the CBQ classifier. When the packets pass through the INPUT handle which is onus for all traffic that enters the system, the bandwidth is split among the various LOCAL system processes (e.g. Apache, MySQL etc).

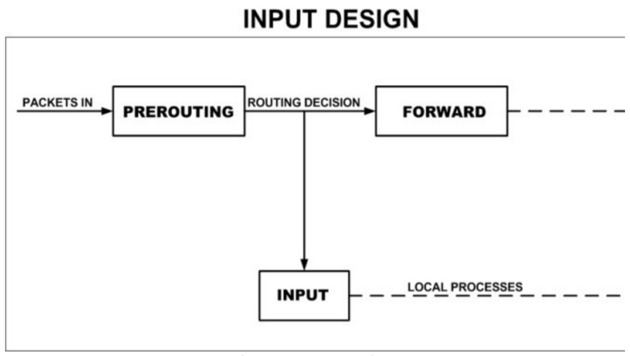


Fig 1.1 Input Design

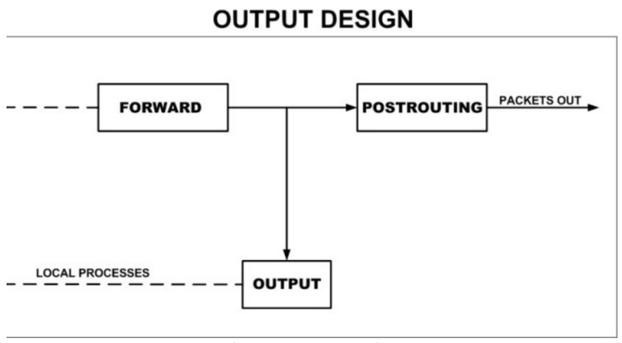


Fig 1.2 Output Design

B. The Output Design

The output of the bandwidth optimizer is shown in Fig 1.2 it consists of the FORWARD chain and the POSTROUTING chain. The packets after being inspected from the input of the optimizer and routed based on the MARK type set by the CBQ classifier, the packets either goes to the POSTROUTING chain or the OUTPUT chain which is onus for all traffic leaving the system. The output traffic as seen comes from the LOCAL system processes forming the egress of the optimizer.

IV. Implementation

We will start our configuration by setting up an ultimate conditioner script which creates a stable platform for the prioritization of our scarce bandwidth. This will guarantee a stable download even when huge upload is happening on the network.

```
#!/bin/bash

# The Ultimate Setup For Your Internet Connection

# Set the following values to somewhat less than your actual
#download

# and uplink speed. In kilobits

DOWNLINK=2000

UPLINK=512

DEV=eth0

# clean existing down- and uplink qdiscs, hide errors
```

```
tc qdisc del dev $DEV root    2> /dev/null > /dev/null

tc qdisc del dev $DEV ingress 2> /dev/null > /dev/null


###### uplink######

# install root CBQ

tc qdisc add dev $DEV root handle 1: cbq avpkt 1000
bandwidth 10mbit

# shape everything at $UPLINK speed - this prevents huge
queues in your

# DSL modem which destroy latency:

# main class

tc class add dev $DEV parent 1: classid 1:1 cbq rate
${UPLINK}kbit \

allot 1500 prio 5 bounded isolated

# high prio class 1:10:

tc class add dev $DEV parent 1:1 classid 1:10 cbq rate
${UPLINK}kbit \

  allot 1600 prio 1 avpkt 1000

# bulk and default class 1:20 - gets slightly less traffic,

#  and a lower priority:

tc class add dev $DEV parent 1:1 classid 1:20 cbq rate
$[9*$UPLINK/10]kbit \

  allot 1600 prio 2 avpkt 1000

# both get Stochastic Fairness:

tc qdisc add dev $DEV parent 1:10 handle 10: sfq perturb 10

tc qdisc add dev $DEV parent 1:20 handle 20: sfq perturb 10

# start filters

# TOS Minimum Delay (ssh, NOT scp) in 1:10:

tc filter add dev $DEV parent 1:0 protocol ip prio 10 u32 \

    match ip tos 0x10 0xff  flowid 1:10


########## downlink #############

# slow downloads down to somewhat less than the real speed
to prevent

# queuing at our ISP.

# attach ingress policer:

tc qdisc add dev $DEV handle ffff: ingress

# filter *everything* to it (0.0.0.0/0), drop everything that's

# coming in too fast:

tc filter add dev $DEV parent ffff: protocol ip prio 50 u32
match ip src \

0.0.0.0/0 police rate ${DOWNLINK}kbit burst 10k drop
flowid :1
```

Within the CBQ qdisc we place two Stochastic Fairness Queues that make sure that multiple bulk streams don't drown each other out. Downstream traffic is policed using a "**tc**" filter containing a Token Bucket Filter. After which we write a script for our CBQ bandwidth optimizer as shown in the following script code:

```
#!/bin/bash

#Adjust CEIL to 75% of your upstream bandwidth limit. In our
case a 512Kbps Uplink

CEIL=384

IFACE="eth0"

MTC="tc"

IPT="/sbin/iptables"


# Createqdics for root

$MTC qdisc add dev $IFACE root handle 1: cbq default 15

$MTC class add dev $IFACE parent 1: classid 1:1 cbq rate
${CEIL}kbit ceil ${CEIL}kbit

$MTC class add dev $IFACE parent 1:1 classid 1:10 cbqrate
90kbit ceil 90kbit prio 0

$MTC class add dev $IFACE parent 1:1 classid 1:11 cbq rate
120kbit ceil ${CEIL}kbitprio 1

$MTC class add dev $IFACE parent 1:1 classid 1:12 cbq rate
20kbit ceil ${CEIL}kbitprio 2

$MTC class add dev $IFACE parent 1:1 classid 1:13 cbq rate
20kbit ceil ${CEIL}kbitprio 2

$MTC class add dev $IFACE parent 1:1 classid 1:14 cbq rate
20kbit ceil ${CEIL}kbitprio 3

$MTC class add dev $IFACE parent 1:1 classid 1:15 cbq rate
30kbit ceil ${CEIL}kbitprio 3


#Now we set the filters so we can classify the packets with
iptables.

$MTC filter add dev $IFACE parent 1:0 protocol ipprio 1
handle 1 fwclassid 1:10

$MTC filter add dev $IFACE parent 1:0 protocol ipprio 2
handle 2 fwclassid 1:11

$MTC filter add dev $IFACE parent 1:0 protocol ipprio 3
handle 3 fwclassid 1:12
```

```
$MTC filter add dev $IFACE parent 1:0 protocol ipprio 4 handle 4 fwclassid 1:13

$MTC filter add dev $IFACE parent 1:0 protocol ipprio 5 handle 5 fwclassid 1:14

$MTC filter add dev $IFACE parent 1:0 protocol ipprio 6 handle 6 fwclassid 1:1
```

```
# We can start marking packets adding rules to the PREROUTING chain in the mangle table.

# We have done a -j RETURN so packets don't traverse all rules.

$IPT -t mangle -A PREROUTING -p icmp -j MARK --set-mark 0x1

$IPT -t mangle -A PREROUTING -p icmp -j RETURN
```

```
# Now we can start adding more rules, lets do proper TOS handling:

$IPT -t mangle -A PREROUTING -m tos --tos Minimize-Delay -j MARK --set-mark 0x1

$IPT -t mangle -A PREROUTING -m tos --tos Minimize-Delay -j RETURN

$IPT -t mangle -A PREROUTING -m tos --tos Minimize-Cost -j MARK --set-mark 0x5

$IPT -t mangle -A PREROUTING -m tos --tos Minimize-Cost -j RETURN

$IPT -t mangle -A PREROUTING -m tos --tos Maximize-Throughput -j MARK --set-mark 0x6

$IPT -t mangle -A PREROUTING -m tos --tos Maximize-Throughput -j RETURN
```

```
# Now prioritize ssh, dns, telnet e.t.c packets:

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 22 -j MARK --set-mark 0x1

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 22 -j RETURN

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 53 -j MARK --set-mark 0x1

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 53 -j RETURN

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 23 -j MARK --set-mark 0x1

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 23 -j RETURN
```

```
# A good idea is to prioritize packets to begin tcp connections, those with SYN flag set:

$IPT -t mangle -I PREROUTING -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -j MARK --set-mark 0x1

$IPT -t mangle -I PREROUTING -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -j RETURN
```

```
# Now prioritize http and https packets:

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 80 -j MARK --set-mark 0x2

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 80 -j RETURN

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 143 -j MARK --set-mark 0x2

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 143 -j RETURN
```

```
# Now prioritize smtp, pop and imap packets:

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 25 -j MARK --set-mark 0x5

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 25 -j RETURN

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 110 -j MARK --set-mark 0x5

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 110 -j RETURN

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 143 -j MARK --set-mark 0x5

$IPT -t mangle -A PREROUTING -p tcp -m tcp --sport 143 -j RETURN
```

```
# we terminate the PREROUTING table with:

iptables -t mangle -A PREROUTING -j MARK --set-mark 0x6
```

```
# We start marking packets adding rules to the OUTPUT chain in the mangle table.

# We have done a -j RETURN so packets don't traverse all rules.

$IPT -t mangle -A OUTPUT -p icmp -j MARK --set-mark 0x1

$IPT -t mangle -A OUTPUT -p icmp -j RETURN
```

```
# Now we can start adding more rules, lets do proper TOS handling:
```

```
$IPT -t mangle -A OUTPUT -m tos --tos Minimize-Delay -j MARK --set-mark 0x1
$IPT -t mangle -A OUTPUT -m tos --tos Minimize-Delay -j RETURN
$IPT -t mangle -A OUTPUT -m tos --tos Minimize-Cost -j MARK --set-mark 0x5
$IPT -t mangle -A OUTPUT -m tos --tos Minimize-Cost -j RETURN
$IPT -t mangle -A OUTPUT -m tos --tos Maximize-Throughput -j MARK --set-mark 0x6
$IPT -t mangle -A OUTPUT -m tos --tos Maximize-Throughput -j RETURN
```

# Now prioritize ssh, dns, telnet e.t.c packets:

```
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 22 -j MARK --set-mark 0x1
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 22 -j RETURN
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 53 -j MARK --set-mark 0x1
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 53 -j RETURN
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 23 -j MARK --set-mark 0x1
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 23 -j RETURN
```

# A good idea is to prioritize packets to begin tcp connections, those with SYN flag set:

```
$IPT -t mangle -I OUTPUT -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -j MARK --set-mark 0x1
$IPT -t mangle -I OUTPUT -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -j RETURN
```

# Now prioritize http and https packets:

```
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 80 -j MARK --set-mark 0x2
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 80 -j RETURN
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 143 -j MARK --set-mark 0x2
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 143 -j RETURN
```

# Now prioritize smtp, pop and imap packets:

```
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 25 -j MARK --set-mark 0x5
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 25 -j RETURN
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 110 -j MARK --set-mark 0x5
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 110 -j RETURN
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 143 -j MARK --set-mark 0x5
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 143 -j RETURN
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 995 -j MARK --set-mark 0x5
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 995 -j RETURN
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 465 -j MARK --set-mark 0x5
$IPT -t mangle -A OUTPUT -p tcp -m tcp --sport 465 -j RETURN
```

# We terminate the OUTPUT table with:

```
iptables -t mangle -A OUTPUT -j MARK --set-mark 0x3
```

The next is our squid delay pools configuration. Based on our CBQ implementation, we are guaranteed of a fair system (optimizer) that will prioritize traffic and allow them pass through the proper chains as shown on the input and output design diagrams. We will now limit individual clients or user on our network to use the right bandwidth when browsing web pages and downloading files from the network by implementing Squid delay pools to help us fine tune the bandwidth utilization on individual systems / users on the network.

##==========Delay Pools Configuration=========

# Lets Create some couple of Access Control List (ACLs)

```
aclmpot_neturl_regex -i 192.168.2.0/24
aclbad_extensionsurl_regex -i ftp .mp3 .wav .mpeg .avi .mpg .wmv .wma .m4 .iso.zip .rar .tar.gz .tar.bz2 .vbs$
aclmpot_browsingurl_regex -i .html .php .jpg .jpeg .png .gif .ico .swf .asp .aspx .pdf .inc .doc .ppt .cfm .cfm$
aclwork_times time 08:00-22:00
delay_pools 3
```

# Lets us allow unlimited traffic on our local network

delay_class 1 2

delay_parameters 1 -1/-1 -1/-1

delay_access 1 allow mpot_net

# Lets us now limit bandwidth for bad extensions

delay_class 2 2

delay_parameters 2 16000/16000 8000/8000

delay_access 2 allow work_timesbad_extensions

# === EXPERIMENTAL ===

# Lets make sure Pages are served at 80Kbps per client

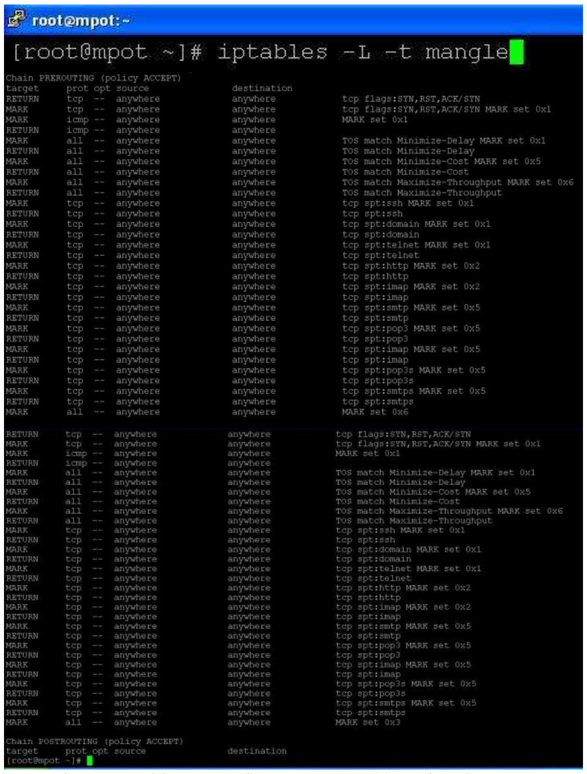
delay_class 3 2

delay_parameters 3 187500/187500 10000/10000

delay_access 3 allow mpot_browsing

V. Results

After putting all the codes and configuration together on our Linux Machine using ubuntu linux server version, some testing were made to ensure that it met with the paper goal. The conditions tested include:-



```
root@mpot:~
[root@mpot ~]# iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
RETURN     tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN
MARK       tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN MARK set 0x1
MARK       icmp --  anywhere             anywhere            MARK set 0x1
RETURN     icmp --  anywhere             anywhere
MARK       all  --  anywhere             anywhere            TOS match Minimize-Delay MARK set 0x1
RETURN     all  --  anywhere             anywhere            TOS match Minimize-Delay
MARK       all  --  anywhere             anywhere            TOS match Minimize-Cost MARK set 0x5
RETURN     all  --  anywhere             anywhere            TOS match Minimize-Cost
MARK       all  --  anywhere             anywhere            TOS match Maximize-Throughput MARK set 0x6
RETURN     all  --  anywhere             anywhere            TOS match Maximize-Throughput
MARK       tcp  --  anywhere             anywhere            tcp spt:ssh MARK set 0x1
RETURN     tcp  --  anywhere             anywhere            tcp spt:ssh
MARK       tcp  --  anywhere             anywhere            tcp spt:domain MARK set 0x1
RETURN     tcp  --  anywhere             anywhere            tcp spt:domain
MARK       tcp  --  anywhere             anywhere            tcp spt:telnet MARK set 0x1
RETURN     tcp  --  anywhere             anywhere            tcp spt:telnet
MARK       tcp  --  anywhere             anywhere            tcp spt:http MARK set 0x2
RETURN     tcp  --  anywhere             anywhere            tcp spt:http
MARK       tcp  --  anywhere             anywhere            tcp spt:imap MARK set 0x2
RETURN     tcp  --  anywhere             anywhere            tcp spt:imap
MARK       tcp  --  anywhere             anywhere            tcp spt:smtp MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:smtp
MARK       tcp  --  anywhere             anywhere            tcp spt:pop3 MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:pop3
MARK       tcp  --  anywhere             anywhere            tcp spt:imap MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:imap
MARK       tcp  --  anywhere             anywhere            tcp spt:pop3s MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:pop3s
MARK       tcp  --  anywhere             anywhere            tcp spt:smtps MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:smtps
MARK       all  --  anywhere             anywhere            MARK set 0x6

RETURN     tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN
MARK       tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN MARK set 0x1
MARK       icmp --  anywhere             anywhere            MARK set 0x1
RETURN     icmp --  anywhere             anywhere
MARK       all  --  anywhere             anywhere            TOS match Minimize-Delay MARK set 0x1
RETURN     all  --  anywhere             anywhere            TOS match Minimize-Delay
MARK       all  --  anywhere             anywhere            TOS match Minimize-Cost MARK set 0x5
RETURN     all  --  anywhere             anywhere            TOS match Minimize-Cost
MARK       all  --  anywhere             anywhere            TOS match Maximize-Throughput MARK set 0x6
RETURN     all  --  anywhere             anywhere            TOS match Maximize-Throughput
MARK       tcp  --  anywhere             anywhere            tcp spt:ssh MARK set 0x1
RETURN     tcp  --  anywhere             anywhere            tcp spt:ssh
MARK       tcp  --  anywhere             anywhere            tcp spt:domain MARK set 0x1
RETURN     tcp  --  anywhere             anywhere            tcp spt:domain
MARK       tcp  --  anywhere             anywhere            tcp spt:telnet MARK set 0x1
RETURN     tcp  --  anywhere             anywhere            tcp spt:telnet
MARK       tcp  --  anywhere             anywhere            tcp spt:http MARK set 0x2
RETURN     tcp  --  anywhere             anywhere            tcp spt:http
MARK       tcp  --  anywhere             anywhere            tcp spt:imap MARK set 0x2
RETURN     tcp  --  anywhere             anywhere            tcp spt:imap
MARK       tcp  --  anywhere             anywhere            tcp spt:smtp MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:smtp
MARK       tcp  --  anywhere             anywhere            tcp spt:pop3 MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:pop3
MARK       tcp  --  anywhere             anywhere            tcp spt:imap MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:imap
MARK       tcp  --  anywhere             anywhere            tcp spt:pop3s MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:pop3s
MARK       tcp  --  anywhere             anywhere            tcp spt:smtps MARK set 0x5
RETURN     tcp  --  anywhere             anywhere            tcp spt:smtps
MARK       all  --  anywhere             anywhere            MARK set 0x3

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
[root@mpot ~]#
```

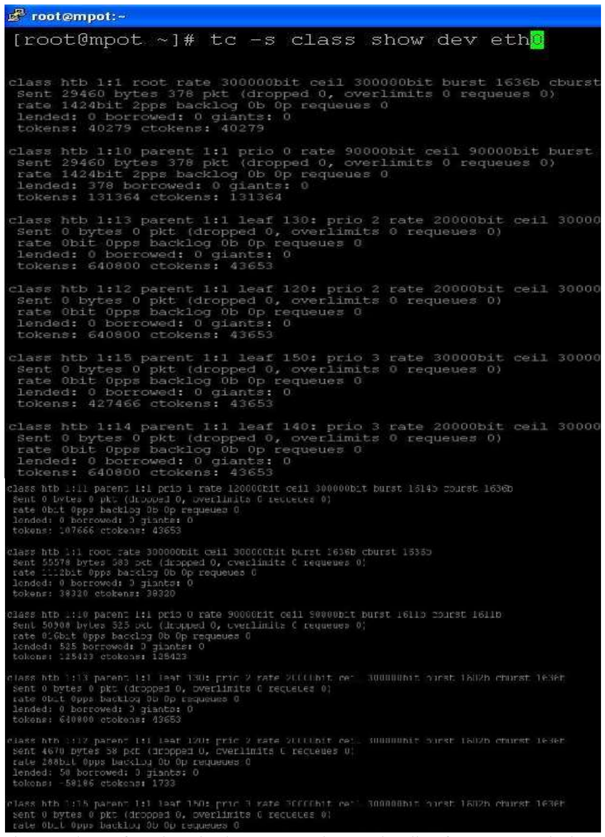
Fig 5.1: Iptables to Confirm Proper Marking of Packets

(i) Flow of packets in the proper class and qdisc from our optimizer (Fig 5.2)

(ii) Iptables to confirm proper MARKing of packets (Fig 5.1)

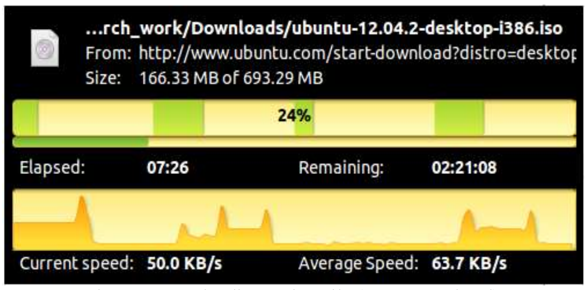
(iii) Squid delay pools bandwidth shaping (Fig 5.3 & Fig 5.4)

For delay pools bandwidth shaping, we connect some workstation systems to our LAN network. After the connection had been established, we try and download an "iso" file from www.ubuntu.com/download and check the speed. The downloading started automatically and the speed was limited to the specified rate we gave during our configuration (64Kbps). Recall that "**.iso**" extension is a member of bad_extensions acl in our squid delay pool configuration. In the 24% and 51% of the downloading, the speed was within the specified limit as shown in Fig 5.3 and Fig 5.4 respectively. The fact that our download speed rate did not exceed the specified limit has proved that users on our network are using our squid configuration and also our network behavior is in line with the configurations we made earlier.



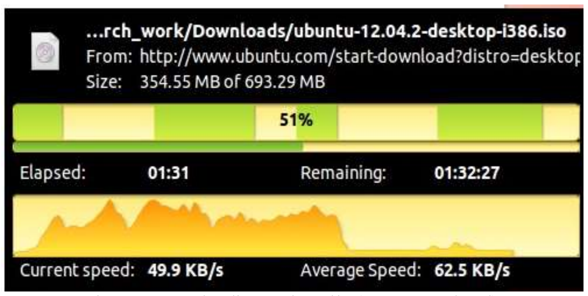
```
root@mpot:~
[root@mpot ~]# tc -s class show dev eth0

class htb 1:1 root rate 300000bit ceil 300000bit burst 1636b cburst
 Sent 29460 bytes 378 pkt (dropped 0, overlimits 0 requeues 0)
 rate 1424bit 2pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 40279 ctokens: 40279

class htb 1:10 parent 1:1 prio 0 rate 90000bit ceil 90000bit burst
 Sent 29460 bytes 378 pkt (dropped 0, overlimits 0 requeues 0)
 rate 1424bit 2pps backlog 0b 0p requeues 0
 lended: 378 borrowed: 0 giants: 0
 tokens: 131364 ctokens: 131364

class htb 1:13 parent 1:1 leaf 130: prio 2 rate 20000bit ceil 30000
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 640800 ctokens: 43653

class htb 1:12 parent 1:1 leaf 120: prio 2 rate 20000bit ceil 30000
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 640800 ctokens: 43653

class htb 1:15 parent 1:1 leaf 150: prio 3 rate 30000bit ceil 30000
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 427466 ctokens: 43653

class htb 1:14 parent 1:1 leaf 140: prio 3 rate 20000bit ceil 30000
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 640800 ctokens: 43653
class htb 1:11 parent 1:1 prio 1 rate 120000bit ceil 300000bit burst 1614b cburst 1636b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 107666 ctokens: 43653

class htb 1:1 root rate 300000bit ceil 300000bit burst 1636b cburst 1636b
 Sent 55578 bytes 583 pkt (dropped 0, overlimits 0 requeues 0)
 rate 1112bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 38320 ctokens: 38320

class htb 1:10 parent 1:1 prio 0 rate 90000bit ceil 90000bit burst 1611b cburst 1611b
 Sent 50908 bytes 525 pkt (dropped 0, overlimits 0 requeues 0)
 rate 816bit 0pps backlog 0b 0p requeues 0
 lended: 525 borrowed: 0 giants: 0
 tokens: 125423 ctokens: 125423

class htb 1:13 parent 1:1 leaf 130: prio 2 rate 20000bit ceil 300000bit burst 1602b cburst 1636b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 640800 ctokens: 43653

class htb 1:12 parent 1:1 leaf 120: prio 2 rate 20000bit ceil 300000bit burst 1602b cburst 1636b
 Sent 4670 bytes 58 pkt (dropped 0, overlimits 0 requeues 0)
 rate 288bit 0pps backlog 0b 0p requeues 0
 lended: 58 borrowed: 0 giants: 0
 tokens: -58186 ctokens: 1733

class htb 1:15 parent 1:1 leaf 150: prio 3 rate 30000bit ceil 300000bit burst 1602b cburst 1636b
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0
```

Fig 5.2 Flow of Packets in the Proper Class and Qdisc from our Optimizer



...rch_work/Downloads/ubuntu-12.04.2-desktop-i386.iso
From: http://www.ubuntu.com/start-download?distro=desktop
Size: 166.33 MB of 693.29 MB
24%
Elapsed: 07:26 Remaining: 02:21:08
Current speed: 50.0 KB/s Average Speed: 63.7 KB/s

Fig 5.3: Downloading an iso File (24% Completed)



...rch_work/Downloads/ubuntu-12.04.2-desktop-i386.iso
From: http://www.ubuntu.com/start-download?distro=desktop
Size: 354.55 MB of 693.29 MB
51%
Elapsed: 01:31 Remaining: 01:32:27
Current speed: 49.9 KB/s Average Speed: 62.5 KB/s

Fig 5.4: Downloading an iso File (51% Completed)

VI. CONCLUSION

Designing a new bandwidth management scheme is an elusive task. It involve consummate technical skills in this area in order to have a better understanding of the problems that network user might encountered. In our new scheme, the output generated has shown that the limited bandwidth available had been fully optimized through the use of our ultimate conditioner script which created a stable platform for the prioritization of our scarce bandwidth and allow the network traffic to pass through the proper chains. Also individual users have now been limited to use the right bandwidth when browsing web pages and downloading files from the network.

REFERENCES

[1] M. Bradley (2012). "Wireless / Networking Guide," Availble:http://compnetworking.about.com/od/speedtests/g/bldef_bandwidth.htm.

[2] M. Rouse (2010). "Improving Business Intelligence with Network Monitoring Tools," pp.7. Available: http://searchenterprisewan.techtarget.com/definition/bandwidth

[3] G. H. Kimberly, "Why bandwidth and the telecoms?" in wireless and the Challenges of the future, K. Brutt, University Press, 2009, Duluth.

[4] C. J Taylor "Bandwidth and the wireless," 4th Ed, Compton Press, 2010, Los Angeles.

[5] Pactum (2012). "Bandwidth Management & Packet Shaper," Pactum Network solutions. Available: http://www.pactumnetworks.com/bandwidth-management-packet-shaper.

[6] L. Chitanana "Bandwidth management in universities in Zimbabwe: Towards a responsible user base through effective policy implementation," appears in International Journal of Education and Development using Information and Communication Technology (IJEDICT), 2012, Vol. 8, Issue 2, pp. 62-76.

[7] K. Mochizuki, S. Shimazaki, D. Hanawa, and K. Oguchi "Proposal of New Traffic Control Method in the Next Generation Home Network," appears in Telecommunications and Signal Processing (TSP), 2012, 35th International Conference on Sci. & Technol., Seikei Univ., Musashino, Japan.

[8] L. M. Patnaik and K. R. Venugopal "Bandwidth Limiter," presented at second international conference on information processing, Jan., 2008. Published by I.K International Pvt limited, India.

[9] C. George and F. I. Kizumba, "Bandwidth Management Issues in African Universities," presented at the AAU Conference in Lome Togo, 2005.

[10] O. Jimoh, "Bandwidth and Applications," M.Sc thesis, Dept of Computer Science, University of Ilorin, Ilorin, Nigeria, 2007.

[11] I. Labbé, F. St-Onge, D. Kidston, and J. Roy "Experience Applying Policy-Based Techniques to Traffic Management in Low-Bandwidth Heterogeneous Networks," presented at Systems and Networks Communications, ICSNC 2007. Second International Conference, Ottawa.