Written By: Jared Watkins – jared (over at) jaredwatkins (do t) com
Last updated: 04102009

Traffic Shaping Under Linux For Voice Over IP (VOIP) With Asterisk PBX

INTRO: This guide will show how to use a combination of iptables (via fwbuilder) and 'tc' to create a traffic shaping policy for a linux based router that is easy to understand and maintain. This specific arrangement works well for me… your mileage may vary.

This approach uses two steps for shaping traffic to provide adequate bandwidth on a limited speed link. Traffic is classified (mostly) by iptables through the graphical firewall manager called Firewall Builder. Examples of individual rules for classifying traffic will also be given. Once classified, traffic is sorted into a 4 layer structure where each layer has rules that specify the minimum and maximum amount of bandwidth it is allowed to use. If setup correctly each level will be able to 'borrow' bandwidth from the other levels to ensure maximum throughput for all possible traffic in all situations.

A reminder that you can only shape what you transmit. This example will demonstrate shaping the outgoing traffic on link with a max theoretical speed of 512k upload.

I will implement the following rules:

1:1 - Root Class Limits Max Upload to 460000 Kbit (90% of max)

1:10 – Voip Class includes IAX2, SIP, RTP, H323
Guaranteed Rate: 428000 Kbit
Max Rate: 428000 Kbit

1:20 Interactive Class includes ACK Packets, DNS, SSH, Telnet, DHCP, VPNs
Guaranteed Rate: 64000 Kbit
Max Rate: 460000 Kbit

1:30 Class Includes Imap, pop, smtp, http, rsync, slingbox
Guaranteed Rate: 128000 Kbit
Max Rate: 460000 Kbit

1:40 Bulk Class Includes Everything Not Specifically Classified
Guaranteed Rate: 38000 Kbit
Max Rate: 460000 Kbit

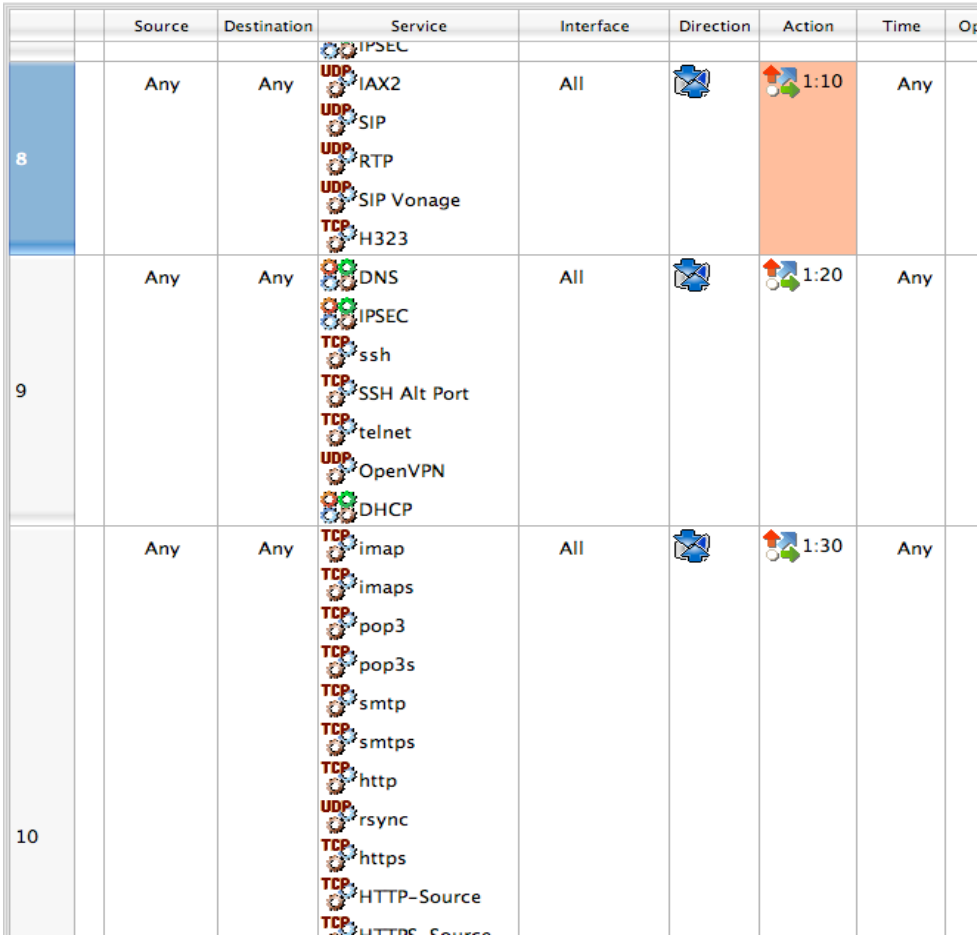
Normally the guaranteed rates should add up to the max upload speed of the root class. In this case I am allowing the voip class to potentially use all but 32 Kbit in the case of multiple calls. The remaining 32 Kbit will be spread across the other classes with priority given to 1:20.

Note: If you want to shape the outgoing traffic of servers you are hosting you must create objects that match on the correct source ports as the standard objects are matching on destination ports.

Note: Values are in bits or Kbits and not Bytes!

Figure 1 is a screen shot from FW Builder showing how the classification rules are built in the GUI.

Figure 2 shows some of these rules as written with iptables.



		Source	Destination	Service	Interface	Direction	Action	Time	Op
				IPSEC					
8		Any	Any	UDP IAX2 UDP SIP UDP RTP UDP SIP Vonage TCP H323	All		1:10	Any	
9		Any	Any	DNS IPSEC TCP ssh TCP SSH Alt Port TCP telnet UDP OpenVPN DHCP	All		1:20	Any	
10		Any	Any	TCP imap TCP imaps TCP pop3 TCP pop3s TCP smtp TCP smtps TCP http UDP rsync TCP https TCP HTTP-Source TCP HTTPS-Source	All		1:30	Any	

Figure 1

```
#
$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  --dport 1720  -j CLASSIFY  --set-class 1:10
$IPTABLES -t mangle -A POSTROUTING -p udp -m udp  --dport 5060:5082  -j CLASSIFY  --set-class 1:10
$IPTABLES -t mangle -A POSTROUTING -p udp -m udp  --dport 8000:10000  -j CLASSIFY  --set-class 1:10
$IPTABLES -t mangle -A POSTROUTING -p udp -m udp  --dport 8000:8020  -j CLASSIFY  --set-class 1:10
$IPTABLES -t mangle -A POSTROUTING -p udp -m udp  -m multiport  --dports 5036,4569  -j CLASSIFY  --set-class 1:10

$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  -m multiport  --dports 53,22,666,23  -j CLASSIFY  --set-class 1:20
$IPTABLES -t mangle -A POSTROUTING -p udp -m udp  -m multiport  --dports 53,500,1194,68,67  -j CLASSIFY  --set-class 1:20
$IPTABLES -t mangle -A POSTROUTING -p 50  -j CLASSIFY  --set-class 1:20
$IPTABLES -t mangle -A POSTROUTING -p 51  -j CLASSIFY  --set-class 1:20

$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  --sport 80  -j CLASSIFY  --set-class 1:30
$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  --sport 443  -j CLASSIFY  --set-class 1:30
$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  --sport 143  -j CLASSIFY  --set-class 1:30
$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  --sport 993  -j CLASSIFY  --set-class 1:30
$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  --sport 25  -j CLASSIFY  --set-class 1:30
$IPTABLES -t mangle -A POSTROUTING -p tcp -m tcp  -m multiport  --dports 143,993,110,995,25,465,80,443,5001  -j CLASSIFY  --set-class 1:30
$IPTABLES -t mangle -A POSTROUTING -p udp -m udp  --dport 873  -j CLASSIFY  --set-class 1:30
```

Figure 2

Written By: Jared Watkins – jared (over at) jaredwatkins (do t) com
Last updated: 04102009

The following shell script implements the rules defined above in a series of tc commands.

```
#!/bin/sh

# 90% of download
DOWNLINK=8400

# Dev should be the public interface you are 'transmitting' on
DEV=eth0

if [ "$1" = "status" ]
then
        clear
#       tc -s qdisc ls dev $DEV
        watch -n1 tc -s class ls dev $DEV
        exit
fi

# 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

if [ "$1" = "stop" ]
then
        exit
fi

#install root HTB, point default traffic to 1:40
tc qdisc add dev $DEV root handle 1: htb default 40

#shape everything at 90% $UPLINK speed to prevent modem queing
tc class add dev $DEV parent 1: classid 1:1 htb rate 460Kbit burst 6k

#voip class 1:10 - iax, sip, rtp, h323
tc class add dev $DEV parent 1:1 classid 1:10 htb rate 428Kbit ceil 428Kbit burst 6k prio 1

#high prio class 1:20 - DNS, IPSEC, SSH, Telnet, openvpn, dhcp
tc class add dev $DEV parent 1:1 classid 1:20 htb rate 64Kbit ceil 460Kbit burst 6k prio 2

#bulk class 1:30 - imap, pop, smtp, http, rsync, sling
tc class add dev $DEV parent 1:1 classid 1:30 htb rate 128Kbit ceil 460Kbit burst 6k prio 3

#default class 1:40
tc class add dev $DEV parent 1:1 classid 1:40 htb rate 38Kbit ceil 460Kbit burst 6k prio 4


#all 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
tc qdisc add dev $DEV parent 1:30 handle 30: sfq perturb 10
tc qdisc add dev $DEV parent 1:40 handle 40: sfq perturb 10


# These next few lines classify packets with specific TOS/port values set
#
# Classifying by TOS seems to be more reliable than by port
# Configure your Asterisk system to set a tos value of 0x18 to match!
tc filter add dev $DEV parent 1:0 protocol ip prio 10 u32 match ip tos 0x18 0xff flowid 1:10

#TOS Minimum Delay (ssh, NOT scp) in 1:20
tc filter add dev $DEV parent 1:0 protocol ip prio 20 u32 match ip tos 0x10 0xff flowid 1:20

#DNS in interactive class 1:20
tc filter add dev $DEV parent 1:0 protocol ip prio 21 u32 match ip sport 53 0xffff flowid 1:20
tc filter add dev $DEV parent 1:0 protocol ip prio 22 u32 match ip dport 53 0xffff flowid 1:20

#only give TCP ACK's higher priority if this connection is asymmetrical
#if [ ! $DOWNLINK = $UPLINK ]
#then
#give TCP ACK's higher priority in 1:20
tc filter add dev $DEV parent 1: protocol ip prio 23 u32 \
   match ip protocol 6 0xff \
   match u8 0x05 0x0f at 0 \
   match u16 0x0000 0xffc0 at 2 \
   match u8 0x10 0xff at 33 \
```

```
   flowid 1:20
#fi

#ICMP (ip protocol 1) in the interactive class 1:20
tc filter add dev $DEV parent 1: protocol ip prio 25 u32 match ip protocol 1 0xff flowid 1:20

########## downlink #############
# slow downloads down to somewhat less than the real speed  to prevent
# queuing at our ISP. Tune to see how high you can set it.
# ISPs tend to have *huge* queues to make sure big downloads are fast
#
# 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 100 u32 match ip src \
   0.0.0.0/0 police rate ${DOWNLINK}kbit burst 10k drop flowid :1
```

One of the tc rules matches on the TOS field of packets of interest for voip. To use this you should configure your asterisk sip.conf file to use 'tos=0x18' or for asterisk >= 1.4 tos_sip=0x10 tos_audio=0x18 tos_video=0x18

To activate the shaping rules you simply run the script. To see a real time display of what's happening run with the 'status' parameter. Figure 3 shows an example of a fully saturated link with one ULAW encoded VIOP call in progress.

```
class htb 1:10 parent 1:1 leaf 10: prio 1 rate 428000bit ceil 428000bit burst 6Kb cburst 1652b
 Sent 4767675 bytes 17879 pkt (dropped 0, overlimits 0 requeues 0)
 rate 90016bit 51pps backlog 0b 0p requeues 0
 lended: 17879 borrowed: 0 giants: 0
 tokens: 108354 ctokens: 26377

class htb 1:1 root rate 460000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 80212719 bytes 113345 pkt (dropped 0, overlimits 0 requeues 0)
 rate 463600bit 114pps backlog 0b 0p requeues 0
 lended: 80960 borrowed: 0 giants: 0
 tokens: 23485 ctokens: -52721

class htb 1:20 parent 1:1 leaf 20: prio 2 rate 64000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 107497 bytes 1554 pkt (dropped 0, overlimits 0 requeues 0)
 rate 488bit 1pps backlog 0b 0p requeues 0
 lended: 1554 borrowed: 0 giants: 0
 tokens: 742189 ctokens: 27055

class htb 1:30 parent 1:1 leaf 30: prio 3 rate 128000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 1006713 bytes 4875 pkt (dropped 0, overlimits 0 requeues 0)
 rate 7032bit 4pps backlog 0b 0p requeues 0
 lended: 4875 borrowed: 0 giants: 0
 tokens: 367579 ctokens: 26358

class htb 1:40 parent 1:1 leaf 40: prio 4 rate 38000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 74373250 bytes 89107 pkt (dropped 58, overlimits 0 requeues 0)
 rate 369736bit 58pps backlog 0b 70p requeues 0
 lended: 8077 borrowed: 80960 giants: 0
 tokens: -262991 ctokens: -25306
```

Figure 3

Figure 4 shows the same setup but with 3 active calls.

```
class htb 1:10 parent 1:1 leaf 10: prio 1 rate 428000bit ceil 428000bit burst 6Kb cburst 1652b
 Sent 14723752 bytes 58013 pkt (dropped 0, overlimits 0 requeues 0)
 rate 258736bit 151pps backlog 0b 0p requeues 0
 lended: 58013 borrowed: 0 giants: 0
 tokens: 106777 ctokens: 24800

class htb 1:1 root rate 460000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 184318307 bytes 274426 pkt (dropped 0, overlimits 0 requeues 0)
 rate 468352bit 190pps backlog 0b 0p requeues 0
 lended: 182896 borrowed: 0 giants: 0
 tokens: 61597 ctokens: -14609

class htb 1:20 parent 1:1 leaf 20: prio 2 rate 64000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 219947 bytes 3180 pkt (dropped 0, overlimits 0 requeues 0)
 rate 272bit 0pps backlog 0b 0p requeues 0
 lended: 3180 borrowed: 0 giants: 0
 tokens: 744141 ctokens: 27327

class htb 1:30 parent 1:1 leaf 30: prio 3 rate 128000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 2449300 bytes 11768 pkt (dropped 0, overlimits 0 requeues 0)
 rate 6256bit 4pps backlog 0b 0p requeues 0
 lended: 11768 borrowed: 0 giants: 0
 tokens: 371095 ctokens: 27055

class htb 1:40 parent 1:1 leaf 40: prio 4 rate 38000bit ceil 460000bit burst 6Kb cburst 1656b
 Sent 166907419 bytes 201521 pkt (dropped 154, overlimits 0 requeues 0)
 rate 200624bit 34pps backlog 0b 56p requeues 0
 lended: 18569 borrowed: 182896 giants: 0
 tokens: -191994 ctokens: -22371
```

Figure 4