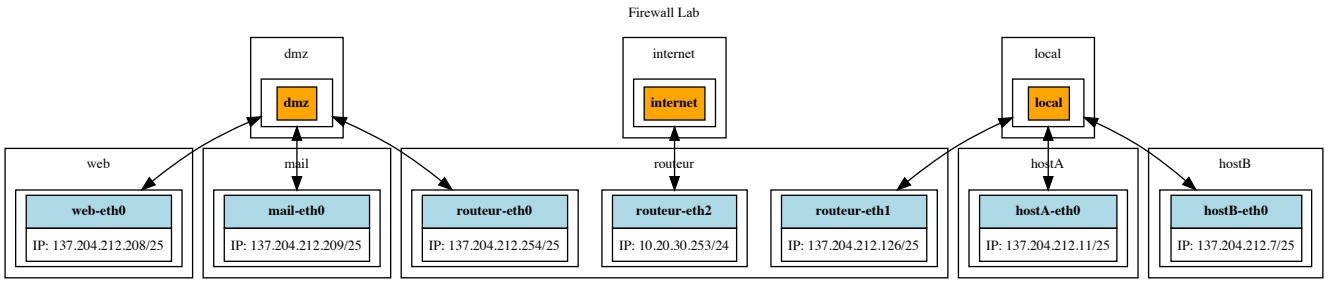
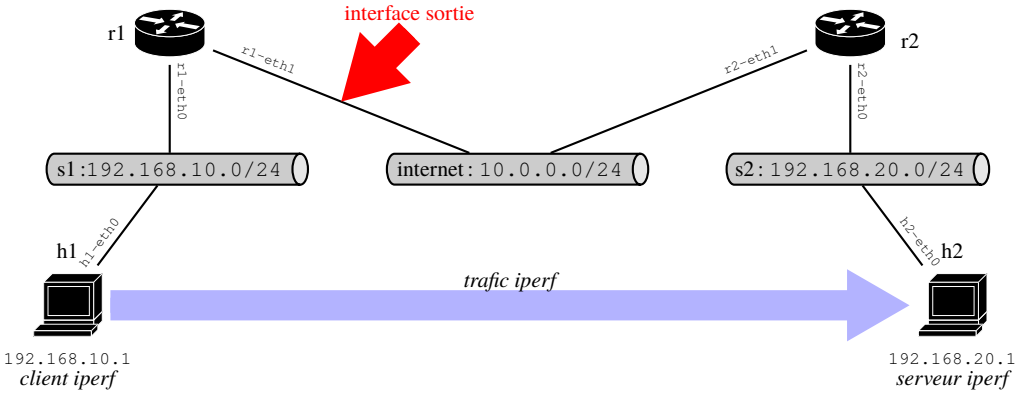
QoS

Démonstration de « *Traffic Shaping* »

Pour cette démonstration, on utilisera le réseau virtuel habituel mis en place à l'aide de `netns` et de switches `openvswitch` :



On va utiliser la commande `iperf` qui envoie du trafic TCP depuis le client, ici ce sera sur le netns `h1`, vers le serveur, placé sur `h2`.



L'interface de sortie du trafic est donc `r1-eth1` et le port de destination

Mesure sans QoS

Sur `h2`, on lance le serveur `iperf` :

xterm

```
pef@atmos:~/NET_LAB$ [h2] iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
```

Puis sur `h1` :

xterm

```
pef@atmos:~/NET_LAB$ [h1] iperf -c 192.168.20.1
------------------------------------------------------------
Client connecting to 192.168.20.1, TCP port 5001
TCP window size:  204 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 46796 connected with 192.168.20.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   118 MBytes  98.6 Mbits/sec
```

L'affichage change sur `h2` :

xterm
```
pef@atmos:~/NET_LAB$ [h2] iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.20.1 port 5001 connected with 192.168.10.1 port 46796
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.2 sec   118 MBytes  96.4 Mbits/sec
```

C'est la QoS définit dans le fichier du simulateur qui s'applique définie à la dernière ligne du du fichier `build_architecture:`

```
# mettre en place la limitation à 100Mbps
tc qdisc add dev internet-r2 root tbf rate 100Mbit latency 50ms burst 1M
```

Mesure avec QoS

ATTENTION

Lors de la configuration de la « *Queueing DISCipline* » `htb`, le débit, « *rate* » est exprimé en notation :

- `mbps` : c-à-d en megabytes per second, exemple : 20mbps ⟹ 160 megabits per second ;
- `mbit` : c-à-d en megabits per seconds, exemple : 20mbit ⟹ 20 megabits per second.

La configuration sur R1 de la QoS pour l'interface `r1-eth1` :

peut échouer s'il n'y a pas de QoS initialement « qdisc noqueue »

xterm
```
pef@atmos:~$ [r1] sudo tc qdisc del dev r1-eth1 root
pef@atmos:~$ [r1] sudo tc qdisc add dev r1-eth1 root handle 1: htb default 10
pef@atmos:~$ [r1] sudo tc class add dev r1-eth1 parent 1: classid 1:10 htb rate 20mbit
pef@atmos:~$ [r1] sudo tc class add dev r1-eth1 parent 1: classid 1:20 htb rate 10mbit
pef@atmos:~$ [r1] tc -d -s class show dev r1-eth1
class htb 1:10 root prio 0 quantum 200000 rate 20Mbit ceil 20Mbit linklayer ethernet
burst 1600b/1 mpu 0b cburst 1600b/1 mpu 0b level 0
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 10000 ctokens: 10000

class htb 1:20 root prio 0 quantum 125000 rate 10Mbit ceil 10Mbit linklayer ethernet
burst 1600b/1 mpu 0b cburst 1600b/1 mpu 0b level 0
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 20000 ctokens: 20000
```

La mise en place de la **classification** du trafic de la commande `iperf` :

xterm
```
pef@atmos:~$ [r1] sudo iptables -t mangle -F
pef@atmos:~$ [r1] sudo tc filter show dev r1-eth1
pef@atmos:~$ [r1] sudo iptables -t mangle -A PREROUTING -p tcp --dport 5001 -j MARK
--set-mark 1
pef@atmos:~$ [r1] sudo tc filter add dev r1-eth1 protocol ip parent 1: handle 1 fw
classid 1:20
pef@atmos:~$ [r1] sudo tc filter show dev r1-eth1
filter parent 1: protocol ip pref 49152 fw chain 0
filter parent 1: protocol ip pref 49152 fw chain 0 handle 0x1 classid 1:20
```

Le test de QoS depuis `h1` :

xterm
```
pef@atmos:~/NET_LAB$ [h1] iperf  -c 192.168.20.1
------------------------------------------------------------
Client connecting to 192.168.20.1, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 46820 connected with 192.168.20.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  13.9 MBytes  11.5 Mbits/sec
```

Et sur `h2` :

xterm
```
pef@atmos:~/NET_LAB$ [h2] iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.20.1 port 5001 connected with 192.168.10.1 port 46796
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.2 sec   118 MBytes  96.4 Mbits/sec
[  4] local 192.168.20.1 port 5001 connected with 192.168.10.1 port 46820
[  4]  0.0-12.2 sec  13.9 MBytes  9.57 Mbits/sec
```

Et pour l'état de « *traffic control* » sur `r1` :

xterm

```
pef@atmos:~/NET_LAB$ [r1] tc -d -s class show dev r1-eth1
class htb 1:10 root prio 0 quantum 200000 rate 20Mbit ceil 20Mbit linklayer ethernet
burst 1600b/1 mpu 0b cburst 1600b/1 mpu 0b level

 Sent 42 bytes 1 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
 lended: 1 borrowed: 0 giants: 0
 tokens: 9737 ctokens: 9737

class htb 1:20 root prio 0 quantum 125000 rate 10Mbit ceil 10Mbit linklayer ethernet
burst 1600b/1 mpu 0b cburst 1600b/1 mpu 0b level

 Sent 15212366 bytes 10051 pkt (dropped 0, overlimits 5106 requeues 0)
 backlog 0b 0p requeues 0
 lended: 5109 borrowed: 0 giants: 0
 tokens: 19175 ctokens: 19175
```

Et le firewall sur `r1` :

xterm

```
pef@atmos:~/NET_LAB$ [r1] sudo iptables -t mangle -nvL
Chain PREROUTING (policy ACCEPT 8726 packets, 13M bytes)
 pkts bytes target     prot opt in     out     source               destination
 4549   13M MARK       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp dpt:5001 MARK set 0x1

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 8726 packets, 13M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 8726 packets, 13M bytes)
 pkts bytes target     prot opt in     out     source               destination
```

On peut également vérifier que sans la mise en place du filtre avec « *tc filter* », le trafic global de l'interface r1-eth1 est bien limité à 20mbits :

xterm

```
pef@atmos:~/NET_LAB$ [h2] iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.20.1 port 5001 connected with 192.168.10.1 port 47074
[  4]  0.0-11.5 sec  26.1 MBytes  19.1 Mbits/sec
```

et sur `h1` :

xterm

```
pef@atmos:~/NET_LAB$ [h1] iperf  -c 192.168.20.1
------------------------------------------------------------
Client connecting to 192.168.20.1, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 47074 connected with 192.168.20.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.5 sec  26.1 MBytes  20.9 Mbits/sec
```

On peut également vérifier que la syntaxe utilisée dans la correction du TD est aussi correcte et suffisante pour appliqué le filtre sur `r1` (c-à-d mettre le trafic dans la file 20) :

xterm

```
pef@atmos:~/NET_LAB$ [r1] sudo tc filter add dev r1-eth1 handle 1 fw flowid 1:20
pef@atmos:~/NET_LAB$ [r1] tc filter show dev r1-eth1
filter parent 1: protocol all pref 49152 fw chain 0
filter parent 1: protocol all pref 49152 fw chain 0 handle 0x1 classid 1:20
```

Et le résultat sur h2 :

xterm

```
pef@atmos:~/NET_LAB$ [h2] iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.20.1 port 5001 connected with 192.168.10.1 port 47074
[  4]  0.0-11.5 sec  26.1 MBytes  19.1 Mbits/sec
[  4] local 192.168.20.1 port 5001 connected with 192.168.10.1 port 47076
[  4]  0.0-12.2 sec  13.9 MBytes  9.57 Mbits/sec
```

et sur `h1` :

xterm
```
pef@atmos:~/NET_LAB$ [h1] iperf  -c 192.168.20.1
------------------------------------------------------------
Client connecting to 192.168.20.1, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 47076 connected with 192.168.20.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec  13.9 MBytes  11.5 Mbits/sec
```

QoS sur la mauvaise interface, c-à-d l'interface d'entrée

Juste pour tester, on applique la QoS sur `r1-eth0` l'interface d'entrée du trafic :

xterm
```
pef@atmos:~/NET_LAB$ [r1] sudo iptables -t mangle -F
pef@atmos:~/NET_LAB$ [r1] sudo iptables -t mangle -A PREROUTING -p tcp --dport 5001 -j
MARK --set-mark 1
pef@atmos:~/NET_LAB$ [r1] sudo tc qdisc del dev r1-eth1 root
pef@atmos:~/NET_LAB$ [r1] tc -d -s class show dev r1-eth0
pef@atmos:~/NET_LAB$ [r1] sudo tc qdisc add dev r1-eth0 root handle 1: htb default 10
pef@atmos:~/NET_LAB$ [r1] sudo tc class add dev r1-eth0 parent 1: classid 1:20 htb rate
10mbit
pef@atmos:~/NET_LAB$ [r1] sudo tc class add dev r1-eth0 parent 1: classid 1:10 htb rate
20mbit
pef@atmos:~/NET_LAB$ [r1] sudo tc filter add dev r1-eth0 protocol ip parent 1: handle 1
fw classid 1:20
pef@atmos:~/NET_LAB$ [r1] tc -d -s class show dev r1-eth0
class htb 1:10 root prio 0 quantum 200000 rate 20Mbit ceil 20Mbit linklayer ethernet
burst 1600b/1 mpu 0b cburst 1600b/1 mpu 0b level 0
 Sent 640690 bytes 9607 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
 lended: 9607 borrowed: 0 giants: 0
 tokens: 9587 ctokens: 9587

class htb 1:20 root prio 0 quantum 125000 rate 10Mbit ceil 10Mbit linklayer ethernet
burst 1600b/1 mpu 0b cburst 1600b/1 mpu 0b level 0
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
 lended: 0 borrowed: 0 giants: 0
 tokens: 20000 ctokens: 20000
```

La QoS par défaut, sans classification, s'est appliquée sur le trafic en sortie de `r1` sur `r1-eth0`, c-à-d sur les « ACKs *» du trafic TCP de `h2` vers `h1`, ce qui n'a pas ralenti le trafic de `h1` vers `h2`.*

Les affichages sur `h1` :

xterm
```
pef@atmos:~/NET_LAB$ [h1] iperf  -c 192.168.20.1
------------------------------------------------------------
Client connecting to 192.168.20.1, TCP port 5001
TCP window size:  340 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 47058 connected with 192.168.20.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec   119 MBytes  98.6 Mbits/sec
```

Il n'y a pas eu d'effet puisque la QoS est appliquée sur la mauvaise interface...

Les affichages sur `h2` :

xterm
```
pef@atmos:~/NET_LAB$ [h2] iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.20.1 port 5001 connected with 192.168.10.1 port 47058
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.3 sec   119 MBytes  96.4 Mbits/sec
```

Il n'y a pas eu d'effet puisque la QoS est appliquée sur la mauvaise interface...

Le firewall a marqué les paquets :

xterm

```
pef@atmos:~/NET_LAB$ [r1] sudo iptables -t mangle -nvL
Chain PREROUTING (policy ACCEPT 19303 packets, 126M bytes)
 pkts bytes target     prot opt in     out     source               destination
 9697  125M MARK       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp dpt:5001 MARK set 0x1

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 19303 packets, 126M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 19303 packets, 126M bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Attention

Les cartes réseaux modernes réalisent une partie du traitement de la segmentation TCP directement, c-à-d indépendamment du CPU.
Ce travail s'appelle « *tcp-segmentation-offload* » pour « *décharger* » le CPU d'une partie du travail de re-agglomérer les segments TCP.
Pour voir si votre carte le fait :

xterm

```
pef@bmax:~/NETLAB$ [h1] ethtool -k h1-eth0 | grep 'tcp-segmentation-offload'
tcp-segmentation-offload: on
```

Cette « *dé-segmentation* » empêche la machine qui l'applique de voir la taille réel des segments TCP par un outil comme `tcpdump` ou `tshark`, car ils indiquent la taille de segments TCP agglomérés et pouvant alors avoir des tailles supérieures au MSS et à la MTU.
Pour permettre l'obtention des tailles réelles des segments TCP, il faut désactiver cette option :

xterm

```
pef@bmax:~/NETLAB$ [h1] sudo ethtool -K h1-eth0 tso off
```

Vous pouvez alors obtenir des infos sur votre trafic TCP pendant que vous lancez la commande `iperf` :

xterm

```
pef@bmax:~/NETLAB$ [h1] ss -i
Netid State Recv-Q Send-Q  Local Address:Port    Peer Address:Port Process
tcp   ESTAB 0      709520   192.168.10.1:40822   192.168.20.1:5001
         cubic wscale:10,10 rto:312 rtt:109.843/0.261 mss:1448 pmtu:1500 rcvmss:536
advmss:1448 cwnd:96 ssthresh:21 bytes_sent:2338520 bytes_acked:2200961
segs_out:1617 segs_in:812 data_segs_out:1615 send 10.1Mbps lastrcv:1840 pacing_rate
12.1Mbps delivery_rate 9.6Mbps delivered:1521 busy:1840ms unacked:95
rcv_space:14480 rcv_ssthresh:64088 notsent:571960 minrtt:0.366
```

Ces informations ont été obtenues avec QoS (on remarque que l'on a des infos sur le « slow start threshold *», que l'algorithme de gestion de la congestion est «* cubic *», la valeur du «* RTO *», le «* window scale *», etc.)*

xterm

```
pef@bmax:~/NETLAB$ [h1] ss -i
Netid State Recv-Q Send-Q  Local Address:Port    Peer Address:Port Process
tcp   ESTAB 0      1135232  192.168.10.1:40836   192.168.20.1:5001
         cubic wscale:10,10 rto:296 rtt:93.238/0.642 mss:1448 pmtu:1500 rcvmss:536
advmss:1448 cwnd:163 ssthresh:21 bytes_sent:4387440 bytes_acked:4152865
segs_out:3032 segs_in:1524 data_segs_out:3030 send 20.3Mbps lastrcv:1736
pacing_rate 24.3Mbps delivery_rate 19.1Mbps delivered:2869 busy:1736ms unacked:162
rcv_space:14480 rcv_ssthresh:64088 notsent:900656 minrtt:0.341
```

Ces informations ont été obtenues sans QoS.

Pour obtenir la taille des paquets TCP en « *direct* » :

xterm

```
pef@bmax:~/NETLAB$ [h1] sudo tshark -l -i h1-eth0 -Y 'tcp.len and
ip.dst==192.168.20.1' -e tcp.len -T fields
```

Les segments perdus :

xterm

```
pef@bmax:~/NETLAB$ [h1] sudo tshark -l -i h1-eth0 -Y 'tcp.analysis.lost_segment and
ip.dst==192.168.20.1' -e tcp.analysis.lost_segment -T fields
```

Pour avoir le débit en temps réel avec l'outil `bmon` :