Faculté des Sciences & Techniques — Université de Limoges

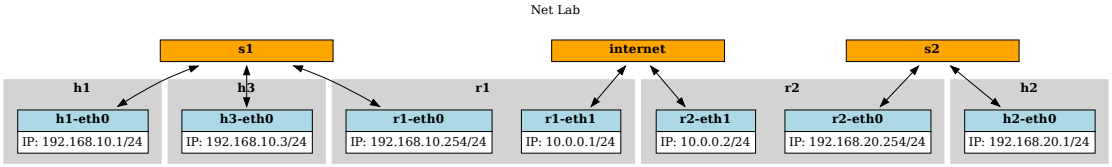
Master 1ère année

Admin Rés

Projet

QoS & NetFilter

Cadre d'expérimentation et de développement



Nous utiliserons le fichier habituel `build_architecture` de TP pour construire l'architecture de travail avec une modification à la fin :

```
#!/bin/bash -x

# créer les namespaces pour les hôtes
ip netns add h1
ip netns add h2
...
ip netns exec h3 ip r add default via 192.168.10.254

# activer le routage sur r1 et r2
ip netns exec r1 sudo sysctl net.ipv4.conf.all.forwarding=1
ip netns exec r2 sudo sysctl net.ipv4.conf.all.forwarding=1
```

Nous supprimerons les deux dernières lignes du fichier :

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

Outils à installer

Vous aurez besoin d'installer les outils suivants :

xterm
```
$ sudo apt install iperf hping3 tcpdump tshark
```

- L'outil `hping3` permet de « *forger* » des paquets, c-à-d de fabriquer des paquets avec le contenu que l'on veut, y compris en utilisant une adresse IP source qui n'est pas celle de la machine où l'on exécute la commande.
  Ces paquets sont envoyés indépendamment de la pile TCP/IP de la machine.
- Les outils `tcpdump` et `tshark` vont nous permettre d'écouter le trafic, avec pour `tshark` la possibilité de faire une analyse du trafic.
- L'outil `iperf` va servir à créer du trafic basé UDP et TCP et à en mesurer le débit.

Travail

- Le travail est à réaliser en **binôme**.
- Un **court rapport** est à produire avec des captures de résultat des différents outils pour illustrer votre configuration.
- Le **source** du programme Python est à remettre également.
- Une **archive** contenant tous les éléments du projet sera à envoyer par l'outil « *filesender* » de l'ENT à `bonnefoi@unilim.fr`.

Cadre des mesures

Le trafic va être généré depuis les hôtes `h1` et `h2` vers l'hôte `h3`.
L'outil utilisé sera `iperf` avec lequel on pourra créer :

- ▷ du trafic TCP et mesurer le débit qu'il peut atteindre au maximum ;
- ▷ du trafic UDP, où l'on pourra « *pousser* » du trafic en fixant un certain débit et mesurer, du côté du récepteur, c-à-d le serveur, le trafic et débit effectivement reçu (la mesure des pertes également) ;
- ▷ du trafic TCP et UDP avec une valeur choisie de TOS.

Ci joint une liste de commandes pour mener vos mesures :

xterm

```
# pour la surveillance de la TOS avec tcpdump
$ sudo tcpdump -i r1-eth1 -lnvv '(ip and (ip[1] & 0x02)  == 0x02)'

# pour la récupération du TOS avec tcpdump sur les paquets SYN
$ sudo tcpdump -i r1-eth1 -lnv "tcp[tcpflags] & (tcp-syn) != 0 and src host
192.168.10.1"

# pour l'envoi de paquets marqués avec iperf
$ iperf -c 192.168.20.1 -p 5002 -u -w10000 -l1472 -b2G -t20 -i.5 --tos 0x10 # en
UDP
$ iperf -c 192.168.20.1 -w10000 -l1472 -b500M -t20 -i.5 --tos 0x02 # en TCP

# pour la réception de trafic
$ iperf -s -u -i 1 -e # en UDP
$ iperf -s -i 1 -e # en TCP

# pour l'interception des paquets TCP dupliqués
$ tshark -l -i r2-eth0 -Y 'tcp.analysis.duplicate_ack' -f 'tcp and port 5001'

# pour voir les qdisc et la quantité de paquets traités
$ tc -s -p -d qdisc show dev ifb0

# pour voir les classes et la quantité de paquets traités avec un rafraîchissement
d'une fois par seconde
$ watch -n 1 sudo tc -s -p -d -g class show dev ifb0

# pour supprimer une qdisc
$ sudo tc qdisc del dev ifb0 root

# pour supprimer la qdisc ingress
$ sudo tc qdisc del dev r2-eth1 ingress

# pour supprimer un filtre par son numéro de priorité
$ sudo tc filter del dev ifb0 prio 1
```

QoS et trafic entrant « *ingress* »

Pour pouvoir appliquer de la QoS sur l'entrée du routeur `r2` nous aurons besoin « *d'inverser* » le sens du trafic sur une interface appelée `ifb`, pour « *Intermediate Functional Block* ».
Nous allons charger un module dans le noyau appelé `ifb` qui va créer une nouvelle interface réseau que l'on « *donnera* » au netns `r2`.

Voici le script, `r2_ifb_install`, à exécuter pour réaliser ce travail :

```
#!/bin/bash

modprobe -r ifb
modprobe ifb numifbs=1

ip link set ifb0 netns r2
ip netns exec r2 ip link set ifb0 up
ip netns exec r2 tc qdisc add dev r2-eth1 ingress handle ffff:
ip netns exec r2 tc filter add dev r2-eth1 parent ffff: matchall action mirred egress
redirect dev ifb0
```

Vous noterez que ce script doit être exécuté avec les droits `root` sur la machine elle-même et non pas dans un des netns de la simulation.

Question à répondre dans le rapport

Vous expliquerez ce que font ces règles et ce à quoi sert l'interface `ifb`

Nous allons tester tout d'abord la qdisc `red` en exécutant sur `r2` :

xterm
```
$ sudo tc qdisc add dev ifb0 root red limit 10000 min 8000 max 9000 avpkt 100 burst
81 harddrop adaptive bandwidth 10Mbit
```

Mesures à mettre dans le rapport

Mesurez votre débit max :

- ▷ avec iperf en TCP sans l'application de cette qdisc ;
- ▷ avec iperf en TCP avec l'application de cette qdisc ;
- ▷ avec iperf en UDP avec et sans cette qdisc.

Faites des copies d'écran des résultats et mesures obtenus avec iperf et commentez ces résultats.
Vous pourrez également analyser les paquets perdus en UDP (iperf en mode serveur) et les « *acks dupliqués* » en TCP (commande tshark).
Une capture des statistiques de la qdisc avec `tc qdisc show dev ifb0` est la bienvenue également.

QoS en ingress et classification par TOS

Nous allons maintenant passer au test lié à l'utilisation du TOS.
Voici le script, `ingress_classifier`, pour réaliser la configuration :

```
tc qdisc show dev ifb0
tc qdisc del dev ifb0 root

tc qdisc show dev ifb0
# créer les classes 1:1, 1:2, 1:3, pour les classes de trafic 0, 1 et 2
sudo tc qdisc add dev ifb0 root handle 1: prio

tc qdisc show dev ifb0
sudo tc qdisc add dev ifb0 parent 1:1 handle 10: sfq
sudo tc qdisc add dev ifb0 parent 1:2 handle 20: tbf rate 200mbit burst 10000 limit
10000
sudo tc qdisc add dev ifb0 parent 1:3 handle 30: tbf rate 50mbit burst 10000 limit
10000
tc qdisc show dev ifb0

tc filter add dev ifb0 parent 1:0 protocol ip prio 1 u32 match ip tos 0x10 0xff flo
wid 1:1
tc filter add dev ifb0 parent 1:0 protocol ip prio 3 u32 match ip tos 0x08 0xff flo
wid 1:3
```

Pour l'exécuter sur `r2`, vous pouvez utiliser :

xterm
```
$ sudo bash -x ingress_classifier
```

Analyse dans le rapport

Vous testerez l'efficacité de ces qdisc à l'aide d'iperf en TCP et UDP et avec les différentes valeurs de TOS choisies (`0x00`, `0x08` et `0x10`).
Vous capturerez également le « *graphe* » des classes et de leurs statistiques obtenues avec la commande `tc -s -p -d -g class show dev ifb0`.
Est-ce plus efficace que précédemment ?

QoS et « *traffic Shaping* »

Toujours sur `r2`, nous allons maintenant abandonner le trafic en « *ingress* » pour nous occuper du trafic en sortie « *egress* ».
Nous n'aurons donc plus besoin de l'interface `ifb0` et des opérations associées (`modprobe` et `tc filter`).
Voici le script mettant en place notre QoS :

```
tc qdisc add dev r2-eth0 root handle 1: htb default 10
tc class add dev r2-eth0 parent 1: classid 1:1 htb rate 90mbit burst 9m
tc class add dev r2-eth0 parent 1:1 classid 1:10 htb rate 40mbit ceil 90mbit burst 9m
tc class add dev r2-eth0 parent 1:1 classid 1:20 htb rate 20mbit ceil 50mbit burst 2m
tc class add dev r2-eth0 parent 1:1 classid 1:30 htb rate 10mbit ceil 20mbit burst 1m
tc filter add dev r2-eth0 parent 1:0 protocol ip prio 1 handle 1 fw flowid 1:10
tc filter add dev r2-eth0 parent 1:0 protocol ip prio 2 handle 2 fw flowid 1:20
tc filter add dev r2-eth0 parent 1:0 protocol ip prio 3 handle 3 fw flowid 1:30
```

À mettre dans le rapport

À l'aide du firewall NetFilter et de la cible `CLASSIFY`, utilisez les différents modes d'iperf pour tester les différentes classes.
Indiquez quelles règles de firewall vous avez utilisées pour rediriger le trafic vers les différentes classes.
Capturez à la fois les commandes iperfs et les résultats de la commande `tc -s -p -d -g class show dev r2-eth0`.

QoS à la demande

1 – Soit le programme suivant que vous exécuterez sur `r1` :

```
#!/usr/bin/python3

import subprocess

interface_réseau = b"wlo2"
filtre_écoute = b"'ip and (ip[1] & 0x03 == 0x03)'"

écoute_réseau = subprocess.Popen(b"tcpdump -c 1 -ln -i %s %s"%(interface_réseau,
filtre_écoute),shell = True, stdout = subprocess.PIPE, stderr = subprocess.PIPE)

print("Résultat du TCPDUMP :", écoute_réseau.stdout.read())
```

Et la commande `hping` suivante :

xterm

```
pef@bmax:~/PROJADMRES$ [h1] sudo hping3 -c 1 -V --syn --destport 80 --tos 03 --ttl
1 192.168.20.2
```

Vérifiez bien que lors de l'exécution de la commande `hping3`, le programme Python réagit bien à la réception du paquet « *forgé* ».

À mettre dans le rapport

Écrivez deux outils en Python :

- le premier exécuté sur `h1` permettant de générer un paquet permettant d'activer de la QoS sur `r2`, par l'intermédiaire d'une modification sur `r1`, pour le trafic provenant d'une machine du réseau `192.168.10.0/24` ;
- le second exécuté sur `r1` permettant d'intercepter ce paquet et de créer la règle de firewall sur `r1` permettant de faire la modification des paquets (marquage TOS) déclenchant la QoS demandée sur `r2` pour le trafic associé.

Pour « *coder* » les informations par le paquet :

- le paquet sera forgé par `hping3` ;
- son adresse IP d'origine est celle de la machine dont le trafic doit être impacté (par exemple `h1` ou `h3`) ;
- son adresse IP de destination est celle de la vraie machine destination (par exemple `h2`), mais son TTL doit le faire disparaitre sur `r2` ;
- son TOS correspond à la file de débit à appliquer en sortie de `r2`.

Vous mettrez le code dans votre archive de remise de projet et vous capturerez les preuves du bon fonctionnement de ces deux programmes : capture du paquet forgé, QoS appliquée.