Tester un protocole réseau en présence de perte de paquets

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 mars 2007

https://www.bortzmeyer.org/tester-protocoles-reseaux-avec-pertes.html

---

Lorsqu'on veut tester un protocole réseau ou bien un programme qui utilise un réseau, il peut être pratique de regarder ce qu'ils donnent avec un réseau de mauvaise qualité ; beaucoup de bogues ne se révèlent qu'alors. Une fonction pratique du noyau Linux, **Netem**, permet de le faire.

Il faut utiliser le système Netem <http://linux-net.osdl.org/index.php/Netem> (*"Network Emulation"*). Cela impose de l'avoir compilé dans le noyau :

```
 Networking -->
   Networking Options -->
     QoS and/or fair queuing -->
        Network emulator
```

Sur une machine Debian, vous pouvez regarder si le noyau a été compilé avec cette option :

```
% grep -i netem /boot/config-2.6.18-4-686
CONFIG_NET_SCH_NETEM=m
```

Si le code de Netem est en module, comme c'est le cas ici, il faut le charger :

```
modprobe sch_netem
```

Ensuite, il faut le programme `tc` qui se trouve dans le paquetage `iproute` (ou `iproute2`, cela dépend des systèmes).

tc est très complexe et mal documenté. Mais on trouve de nombreux textes en ligne. Ici, nous nous limiterons à son utilisation pour Netem. Commençons par un cas simple :

```
tc qdisc add dev  tap0 root netem delay 50ms loss 50%
```

(On peut annuler l'effet de cette commande avec `tc qdisc del dev tap0 root netem`.) Ce script crée une "qdisc" ("*queuing discipline*") Netem avec imposition d'un délai de 50 ms et, en prime, 50 % de perte de paquets. Elle s'applique à **tous** les paquets passant par l'interface tap0, ce qui est sans doute trop brutal. Si on veut sélectionner les paquets :

```
DEVICE="tap0"
INTERESTING="4"
DEST_PORT="7"
CLEAN_FIRST="YES"
USE_IPTABLES="YES"

if [ ! -z "$CLEAN_FIRST" ]; then
   tc qdisc del dev ${DEVICE} parent 1:2
   tc qdisc del dev ${DEVICE} root
   iptables -t mangle -F INPUT
   iptables -t mangle -F OUTPUT
fi

tc qdisc add dev ${DEVICE} root handle 1: prio

tc qdisc add dev ${DEVICE} parent 1:2 handle 4: netem delay 50ms loss 50%

if [ -z "$USE_IPTABLES" ]; then
    tc filter add dev ${DEVICE} protocol ip parent 1:0 prio 2 u32 \
         match ip dport ${DEST_PORT} 0xffff flowid 10:5
else
    tc filter add dev ${DEVICE} protocol ip parent 1:0 prio 2 \
        handle ${INTERESTING} fw flowid 10:5
    iptables -t mangle -A INPUT -i ${DEVICE} -p tcp --dport ${DEST_PORT} \
        -j MARK --set-mark ${INTERESTING}
    iptables -t mangle -A OUTPUT -o ${DEVICE} -p tcp --sport ${DEST_PORT} \
        -j MARK --set-mark ${INTERESTING}
fi
```

Ce script crée également une "qdisc" Netem avec imposition d'un délai de 50 ms et 50 % de perte de paquets.

L'envoi des paquets à cette qdisc (ici, uniquement les paquets à destination du port 7, celui du service "echo") peuvent se faire de deux façons (dans le script ci-dessus, cela dépend de la valeur de la variable `USE_IPTABLES` : on peut utiliser iptables, qui dispose de très nombreuses possibilités de sélection des paquets intéressants, ou bien on peut tout faire avec tc, ce qui dispense d'apprendre iptables mais offre moins de possibilités de choix.

Le résultat se voit nettement, ici avec echoping `<http://echoping.sourceforge.net/>`. Sans Netem, la ligne est très rapide :

```
Minimum time: 0.019670 seconds (3304525 bytes per sec.)
Maximum time: 0.043468 seconds (1495353 bytes per sec.)
Average time: 0.022324 seconds (2911665 bytes per sec.)
Standard deviation: 0.007052
Median time: 0.019978 seconds (3253579 bytes per sec.)
```

Avec les paramètres Netem ci-dessus :

```
Minimum time: 8.224572 seconds (7903 bytes per sec.)
Maximum time: 275.318547 seconds (236 bytes per sec.)
Average time: 58.501345 seconds (1111 bytes per sec.)
Standard deviation: 80.538300
Median time: 22.027636 seconds (2951 bytes per sec.)
```

Il existait aussi un module Netfilter, pom-base-random, qui faisait à peu près la même chose mais il semble avoir disparu.

Avec FreeBSD, dummynet <http://info.iet.unipi.it/~luigi/ip_dummynet> permet de faire à peu près la même chose et est apparu bien avant Netem. par exemple :

```
ipfw add 1000 pipe 1 tcp from judith to kiwi echo
ipfw pipe 1 config plr 0.5
```

et les paquets "echo" envoyés par judith à kiwi seront détruits une fois sur deux.