

Hálózati forgalom szabályozás (QoS) alapjai

Környezet leírása

1. Ubuntu 14.04 LTS tűzfal
 - a. eth0: DHCP, internet
 - i. Max letöltés: 4 mbit/s
 - ii. Max. feltöltés: 128 kbit/s
 - b. eth1: 192.168.200.1/24, intranet
2. Desktop
 - a. eth0: 192.168.200.10/24, gw 192.168.200.1, dns 192.168.200.1 8.8.8.8 8.8.4.4
 - b. "bigup" fájl: 98 MB, ezt fogjuk feltölteni a kimenő vonal terheléséhez

Tanmenet

1. ssh firewall
2. sudo -s
3. tc -s -d qdisc ls
 - a. PFIFO: a legrosszabb választás :(
4. **Lássunk egy SFQ-t...**
 - a. tc qdisc add dev eth0 root sfq perturb 10
 - b. tc qdisc add dev eth1 root sfq perturb 10
 - c. tc -s -d qdisc ls
5. **Classless vs. Classfull**
 - a. Classless: nem lehet "gyereke", azaz nem osztható tovább: csak a legutolsó ponton alkalmazzuk (pl. PFIFO, SFQ)
 - b. Classfull: lehet tovább osztani (pl. CBQ: Class Based Queueing, HTB: Hierarchical Token Bucket), azaz lehetnek osztályai. Ezek is lehetnek classfull vagy classless.
6. **Egyszerű forgalom prioritizálás példa: kimenő forgalom (eth0)**
 - a. Töröld a korábban betett SFQ-t:
 - i. tc qdisc del dev eth0 root
 - ii. tc qdisc del dev eth1 root
 - b. Adj hozzá egy prio schedulert:
 - i. tc qdisc add dev eth0 root handle 1: prio
 1. Ez létrehozott három osztályt:
 - a. tc class ls dev eth0
 2. Adj SFQ-t minden osztályhoz:
 - a. tc qdisc add dev eth0 parent 1:1 handle 10: sfq
 - b. tc qdisc add dev eth0 parent 1:2 handle 20: sfq
 - c. tc qdisc add dev eth0 parent 1:3 handle 30: sfq
 3. Nézzük hová megy a forgalom!
 - a. while ;; do clear ; tc -s qdisc ls dev eth0 ; sleep 1s ; done
 - b. rsync -v --progress --inplace bigup rsync://192.168.2.11/test/bigup
 - i. Ez láthatóan a 20:-ba megy
 - c. ping 192.168.2.1
 - i. Lassú..... és a 20:-ba megy
 - d. ssh -v slapic@192.168.2.11
 - i. Ez is lassú. De a forgalom a 10:-be megy.
 - e. Limitáld a 20:-as sebességét!
 - i. tc qdisc del dev eth0 parent 1:2 handle 20: sfq
 - ii. tc qdisc add dev eth0 parent 1:2 handle 20: tbf rate 60kbit buffer

1600 limit 3000

iii. while ;; do clear ; tc -s qdisc ls dev eth0 ; sleep 1s ; done

f. Lássuk ebből mi lett!

i. rsync -v --progress --inplace bigup rsync://192.168.2.11/test/bigup

ii. ssh -v slapic@192.168.2.11

1. ssh gyorsabb (belépés után)

iii. ping 192.168.2.1

1. Ez is jobb, de nem sokkal. TOS-t nem állít.

2. Állítsunk TOS-t:

a. ping -Q 0x10 192.168.2.1

b. Így gyors!

iv. Rsync pedig lelassult. Nagyon.

c. Ez egy egyszerű és gyors módszer, de van ennél jobb is.

7. Haladóbb módszer: HTB

a. Ezúttal a letöltési irányon teszteljük. Nézzük most ott mi történik!

i. Desztopon

1. rsync -v --progress --inplace rsync://192.168.2.11/test/bigdown bigdown

2. ping -Q 0x10 192.168.2.1

a. megint lassú...

3. ssh -v slapic@192.168.2.11

a. Nem olyan durván rossz, mint a kitömött kimenő vonalnál, de nem az igazi.

b. A root qdisc létrehozása, ezúttal a letöltési irányhoz (a létrehozott kimenő PRIO szabályozást nem bántom).

i. tc qdisc add dev eth1 root handle 1: htb default 30

c. Jöhetnek az osztályok.

i. Elsőnek egy globális limit a vonalhoz (4 mbit/s). Limitáljuk a névleges sebesség alá kicsivel:

1. tc class add dev eth1 parent 1: classid 1:1 htb rate 3900kbit burst 15k

a. Apró probléma (később foglalkozunk vele): így a tűzfalról a belső hálózatba menő forgalmat is limitálom).

ii. Osszuk fel a sávszélességet!

1. Az egyéb forgalmat lassítsuk le max. 2 mbit/s-re és garantáljunk neki 500kbit/s sebességet.

a. Tegyük az rsync forgalmát is ebbe!

2. Az ssh kapjon legalább 500 kbit/s sávszélességet és elhasználhatja az egészet (3900 kbit/s).

3. A böngészés (80-as port a példa kedvéért) kapjon legalább 2900 kbit/s sávszélességet és ez egyben a maximuma is legyen.

4. SOHA ne ossz fel többet, mint amennyi tényleg van! NEM fog működni a szabályozás ($a+b+c$ maximumjai = csatoló maximum, azaz 3900).

a. A max. túllépheti, a garantáltat biztosan megkapja, a többi eloszlik a garantáltak (rate) arányában a maximum erejéig.

iii. Hogyan?

1. Kell három class:

a. Böngészéshez: 1:10 (max. és garantált 2900kbit/s)

i. tc class add dev eth1 parent 1:1 classid 1:10 htb rate 2900kbit ceil 2900kbit burst 15k

b. SSH-nak 1:20 (max. a teljes, garantált 500kbit/s)

i. tc class add dev eth1 parent 1:1 classid 1:20 htb rate 500kbit ceil

3900kbit burst 15k

- c. Egyéb: 1:30 (alapértelmezett, max. 2mbit/s, garantált 500kbit/s)
 - i. `tc class add dev eth1 parent 1:1 classid 1:30 htb rate 500kbit ceil 2mbit burst 5k`
 - d. `tc qdisc ls dev eth1`
 - e. `tc -s class ls dev eth1`
 - i. Az 1:30-ban van most minden forgalom: ez a default.
2. Célszerű mindenhová betenni egy SFQ-t a végére:
- a. `tc qdisc add dev eth1 parent 1:10 handle 10: sfq perturb 10`
 - b. `tc qdisc add dev eth1 parent 1:20 handle 20: sfq perturb 10`
 - c. `tc qdisc add dev eth1 parent 1:30 handle 30: sfq perturb 10`
3. Lássuk mi lett ebből:
- a. `while ;; do clear ; tc -s qdisc ls dev eth1 ; sleep 1s ; done`
 - b. `rm -f bigdown ; rsync -v --progress --inplace rsync://192.168.2.11/test/bigdown bigdown`
 - i. A 30:-as viszi a forgalmat. Ez OK.
 - c. `ping 192.168.2.1`
 - i. Ez is jó: van szabad sávszél, ahová befér. Pedig ő is a 30:-ba megy, de mivel a vonal nincs kitömve és SFQ-van ott, szépen megy.
 - d. `ssh -v slapic@192.168.2.11`
 - i. Mint a villám, hasonló okokból. Pedig még át sem tettük dedikáltra...
4. Jöhetnek a szűrők:
- a. Mivel az alap a 30:-as, minden oda megy, amit nem irányítunk: ezzel nincs dolgunk.
 - b. Az SSH forgalom a 20-asba kell menjen, a célport (22) alapján:
 - i. `tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip sport 22 0xffff flowid 1:20`
 - 1. sport: a bejövő forgalom esetén a forrásport (szerveről jövő forgalom)
 - 2. Kimenő szabályozásnál értelem szerűen dport
 - ii. `while ;; do clear ; tc -s qdisc ls dev eth1 ; sleep 1s ; done`
 - 1. máris pörög (ssh-val vagyok a tűzfalon bent: erre is hat)
 - iii. `ssh -v slapic@192.168.2.11`
 - 1. Ez is oda megy (nem nő máshol a forgalom)
 - c. A böngészés a 10-esbe kell menjen a célport (80) alapján:
 - i. `tc filter add dev eth1 protocol ip parent 1:0 prio 5 u32 match ip sport 80 0xffff flowid 1:10`
 - 1. sport/dport hasonló logikával, mint az SSH-nál
 - ii. `while ;; do clear ; tc -s qdisc ls dev eth1 ; sleep 1s ; done`
 - iii. Lássunk egy weboldalt!
 - 1. <http://www.meetup.com/Linux-rendszergazda-online-offline-meetup/>
 - 2. Ment a 10-be is forgalom.

iv. Feltöltési irány szabályozása HTB-vel

- 1. `tc qdisc ls dev eth0`
- 2. `tc qdisc del dev eth0 root`
- 3. `tc qdisc ls dev eth0`
- 4. Alap HTB

- a. `tc qdisc add dev eth0 root handle 1: htb default 30`
5. Maximáljuk a sávszélességet kifelé kicsivel a valós alá (128kbit/s helyett csak 110):
 - a. `tc class add dev eth0 parent 1: classid 1:1 htb rate 110kbit burst 2k`
6. Osszuk fel a sávszélességet!
 - a. Az egyéb forgalmat lassítsuk le max. 50 kbit/s-re és garantáljunk neki 20kbit/s sebességet.
 - i. Tegyük az rsync forgalmát is ebbe!
 - b. Az ssh kapjon legalább 30 kbit/s sávszélességet és elhasználhatja az egészet (110 kbit/s).
 - c. A böngészés (80-as port a példa kedvéért) kapjon legalább 60 kbit/s sávszélességet és ez egyben a maximuma is legyen (igazából csak a HTTP kérések, POST adatok és fájl-feltöltések használják, a többi nem feltöltés lesz).
 - d. SOHA ne ossz fel többet, mint amennyi tényleg van! NEM fog működni a szabályozás (1+2+3 maximumjai = csatoló maximum, azaz 110).
 - i. A max. túllépheti, a garantáltat biztosan megkapja, a többi eloszlik a garantáltak (rate) arányában a maximum erejéig.
7. Kell három class:
 - a. Böngészéshez: 1:10 (max. 60kbit/s, garantált 60kbit/s)
 - i. `tc class add dev eth0 parent 1:1 classid 1:10 htb rate 60kbit ceil 60kbit burst 2k`
 - b. SSH-nak 1:20 (max. 110kbit/s, garantált 30kbit/s)
 - i. `tc class add dev eth0 parent 1:1 classid 1:20 htb rate 30kbit ceil 110kbit burst 2k`
 - c. Egyéb: 1:30 (alapértelmezett, max. 50kbit/s, garantált 20kbit/s)
 - i. `tc class add dev eth0 parent 1:1 classid 1:30 htb rate 20kbit ceil 50kbit burst 2k`
 - d. `tc qdisc ls dev eth0`
 - e. `tc -s class ls dev eth0`
 - i. Az 1:30-ban van most minden forgalom: ez a default.
8. Célszerű mindenhová betenni egy SFQ-t a végére:
 - a. `tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10`
 - b. `tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10`
 - c. `tc qdisc add dev eth0 parent 1:30 handle 30: sfq perturb 10`
9. Jöhetnek a szűrők:
 - a. Mivel az alap a 30:-as, minden oda megy, amit nem irányítunk: ezzel nincs dolgunk.
 - b. Az SSH forgalom a 20-asba kell menjen, a célport (22) alapján:
 - i. `tc filter add dev eth0 protocol ip parent 1:0 prio 5 u32 match ip dport 22 0xffff flowid 1:20`
 1. Itt most dport, mert a KIMENŐ forgalomnál nézzük.
 - c. A böngészés a 10-esbe kell menjen a célport (80) alapján:
 - i. `tc filter add dev eth0 protocol ip parent 1:0 prio 5 u32 match ip dport 80 0xffff flowid 1:10`
 1. sport/dport hasonló logikával, mint az SSH-nál
10. Tegyük be az IMCP-t is a 20:-ba!
 - a. `tc filter add dev eth0 parent 1:0 protocol ip prio 4 u32 match ip protocol 1 0xff flowid 1:20`

i. Az ICMP az 1-es protokoll

1. `grep -i icmp /etc/protocols`

11. Nem csak SSH: TOS minimum delay használata

a. Az SCP így nem lesz prioritizálva, ami hasznos lehet

b. `tc filter add dev eth0 parent 1:0 protocol ip prio 10 u32 match ip tos 0x10 0xff flowid 1:20`

i. A 0x10 TOS jelzés a “minimum delay”, ezt használják az interaktív programok, mint az SSH.

ii. A sport/dport 22 szűrőket persze ki kell vened!

1. De ne feledd, hogy az SSH csak a már létrejött kapcsolatban állít 0x10-et, addig nem :(

12. Célcím/forráscím szerinti szűrés

a. A 192.168.2.11 felé/felől menő forgalom menjen minden esetben a 30:-ba, akkor is ha ssh, icmp, stb.!

i. `tc filter add dev eth0 parent 1:0 protocol ip prio 5 u32 match ip dst 192.168.2.11 flowid 1:30`

ii. `tc filter add dev eth1 parent 1:0 protocol ip prio 5 u32 match ip src 192.168.2.11 flowid 1:30`

iii. A “prio” kisebb értéke biztosítja, hogy ez fog érvényesülni (a feldolgozás a prio alapján növekvő sorrendben megy).

13. Szűrők kombinálása “ÉS” alapon:

a. a “match” részek ismétlésével.

i. `match ip src 192.168.2.11/32 match ip sport 80`

14. IP protokoll szűrés:

i. `match ip protocol XXX 0xff`

ii. lásd /etc/protocols

15. Kombinálás tűzfallal:

a. FWMARK használatával lehet.

b. Szűrés:

i. `tc filter add dev eth1 protocol ip parent 1:0 prio 1 handle 6 fw flowid 1:30`

c. Jelölés, pl. netről jövő SMTP forgalom irányításához (saját SMTP szerverünkhöz jövő forgalom):

i. `iptables -A PREROUTING -t mangle -i eth0 -p tcp --dport smtp -j MARK --set-mark 6`

v. De mi lesz a tűzfalról a belső hálózatba menő forgalommal? Azt minek lassítjuk?!

1. Belső hálózat pl. 100mbit/s, így a limit legyen 90mbit/s.

a. `tc class add dev eth1 parent 1: classid 1:2 htb rate 90mbit burst 15k`

2. Egyéb szűrést itt nem akarunk, mehet bele egy SFQ:

a. `tc qdisc add dev eth1 parent 1:2 handle 200: sfq perturb 10`

3. `tc -s qdisc ls dev eth1`

a. még nem megy bele forgalom, ez OK

4. `tc filter add dev eth1 parent 1:0 protocol ip prio 1 u32 match ip src 192.168.200.1 flowid 1:2`

5. `while :; do clear ; tc -s qdisc ls dev eth1 ; sleep 1s ; done`

a. Az SSH kapcsolatunk már ezt “terheli”

8. Mi lesz a netről a tűzfalra jövő forgalommal? Arra nem hat az eth1 kimenő korlátja.

a. Itt csak ingress policing lehetőség van. Ez akkor érdekes, ha pl. proxy fut itt (TIPP: ne itt fusson proxy...).

- i. `tc qdisc add dev eth0 handle ffff: ingress`
- ii. `tc filter add dev eth0 parent ffff: protocol ip prio 50 u32 match ip src 0.0.0.0/0 police rate 110kbit burst 10k drop flowid :1`