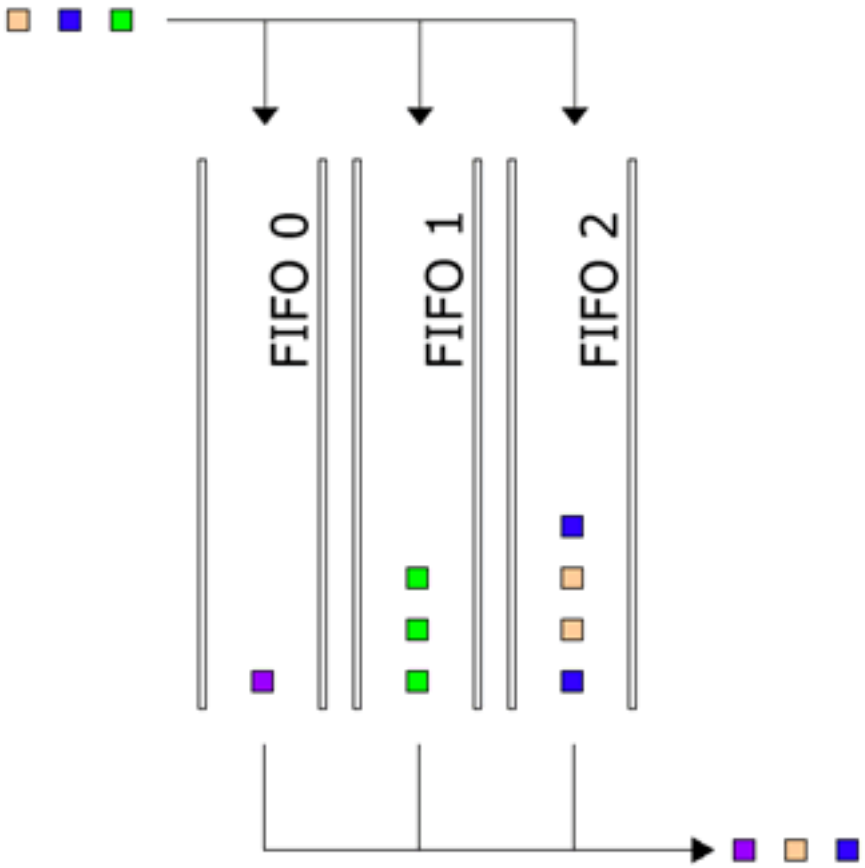
Filtriranje mrežnog prometa na aplikacijskoj razini pomoću Linux rješenja - 2.dio

U ovom članku sistematizirat ću osnovne metode upravljanja redovima čekanja na Linux operativnim sustavima, a to su besklasne metode upravljanja redovima.

Red čekanja (*queue*) je privremena memorija (*buffer*) ili lokacija, koja sadrži konačan broj paketa, koji čekaju da se nad njima obavi neka akcija. Svako mrežno sučelje ima svoj predefinirani red čekanja, a u njemu postoje 3 moguće akcije: ulazak paketa u red (*enqueue*), njegov izlazak iz reda (*dequeue*) i njegovo brisanje (*drop*). U najjednostavnijem se obliku paketi u redu čekanja prosljeđuju po FIFO principu i bez drugih mehanizama nije moguće kontrolirati njihovo ponašanje. Metode upravljanja redovima (*queuing disciplines - qdiscs*) su algoritmi kojima se kontrolira način ulaska paketa u redove i njihovog izlaska. Ukratko, potrebno je prepoznati željeni protokol tj. paket koji pripada interesantnom toku podataka (*match*), nekako ga označiti (*mark*) i onda ga na neki način oblikovati (*shape*).

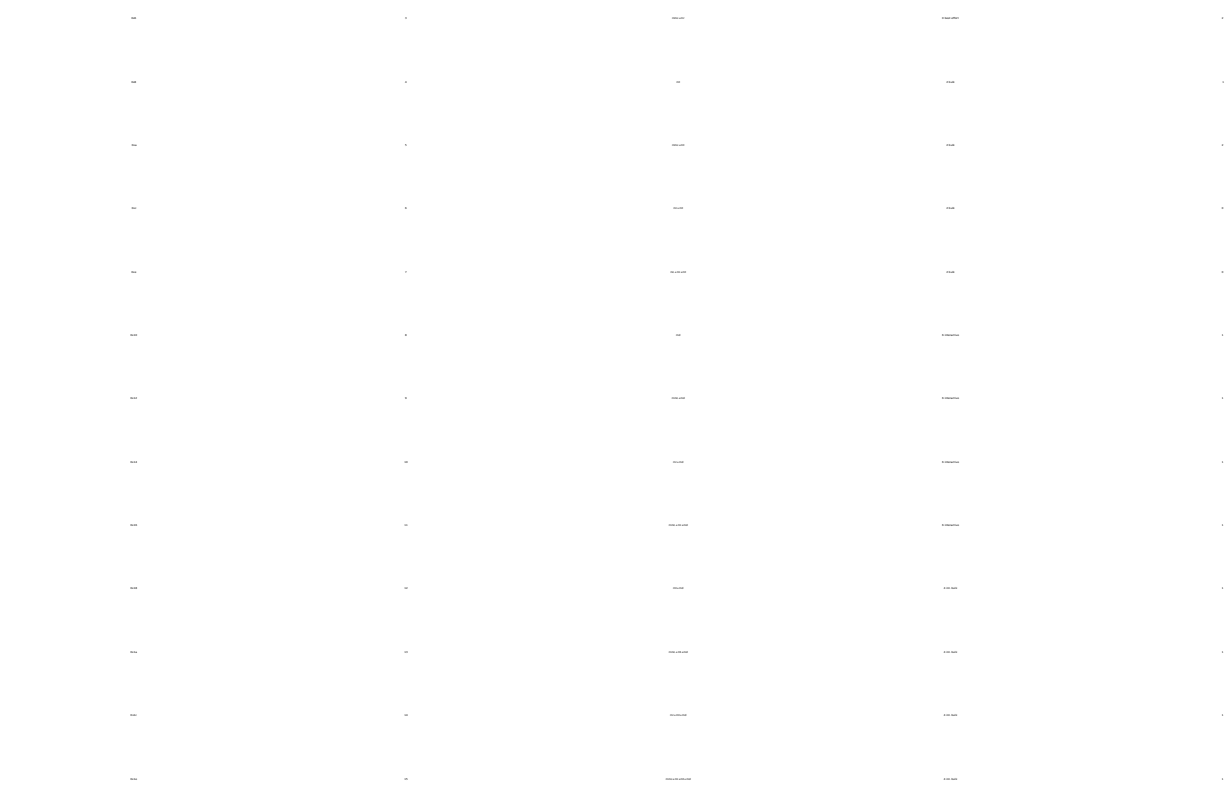
Besklasne metode za upravljanje
Besklasne metode (*classless qdiscs*) su osnovni mehanizmi upravljanja redovima čekanja na Linux operativnim sustavima i njiha se isključivo može kontrolirati izlazni (odlazni) mrežni promet. Te metode ne mogu sadržavati klase i njima nije moguće dodijeliti filtar. Pomoću ovih metoda moguće je oblikovati mrežni promet isključivo na cjelokupnom mrežnom sučelju, bez ikakve njegove podjele, a sama kontrola prometa se obavlja preraspodjelom paketa, unošenjem kašnjenja i brisanjem paketa.

Metoda pfifo_fast
Ovo je osnovna metoda upravljanja redom i bazirana je na first-in first-out (FIFO) principu, a ujedno je i predefinirana metoda upravljanja redom na Linuxu. Za razliku od bazne FIFO metode, pfifo_fast metoda ima mogućnost davanja prioriteta paketima na način da je glavni red podjeljen u tri pod reda u kojima vrijedi FIFO princip i svaki od njih ima svoj prioritet. Shema pfifo_fast metode je prikazana na slici 1.



Važno je naglasiti da pod red 0 ima najveći prioritet, a pod red 2 najmanji. Zbog toga se pod red 1 neće početi prazniti sve dok postoje paketi koji čekaju u pod redu 0. Isto vrijedi i za pod red 2, on neće biti ispražnjen sve dok postoje paketi u pod redu 1.
Kako je pfifo_fast metoda predefinirana i na njenu konfiguraciju nije moguće utjecati, ne postoje ni parametri za njeno podešavanje. Međutim, postoje dva parametra vezana uz ovu metodu, ali nijedan od njih nije moguće postavljati pomoću *tc* komandne linije. Prvi parametar jest *priomap* koji određuje prioritet paketa. Jezgra operativnog sustava čita TOS (Type of Service) polje u zaglavlju IP paketa i prema vrijednosti tog polja paketi se razvrstavaju u pod redove. Četiri od osam bitova TOS polja su definirana na način prikazan u tablici 1.

Linux operativni sustavi imaju predefiniranu interpretaciju svih kombinacija vrijednosti TOS polja i preslikavanje u pod redove na način prikazan u tablici 2.



Stupac "Bitovi" sadrži vrijednosti četiri TOS bita, a stupac "Značenje" Linux interpretaciju vrijednosti polja "Bitovi". Na primjer, vrijednost 15 u stupcu "Bitovi" znači da paket traži minimiziranje troška, maksimalnu pouzdanost, maksimalnu propusnost i minimalno kašnjenje, a po interpretaciji te vrijednosti, paket će biti preslikan u pod red 1.
Ovaj parametar također omogućuje postavljanje većeg prioriteta koji nisu povezani s TOS poljem u zaglavlju paketa, nego se postavljaju pomoću drugih mehanizama, a načini su opisani u RFC 1349.

Drugi parametar jest *txqueuelen*, a koji određuje duljinu reda. Njega je moguće postaviti prilikom konfiguracije mrežnog sučelja, npr. sa

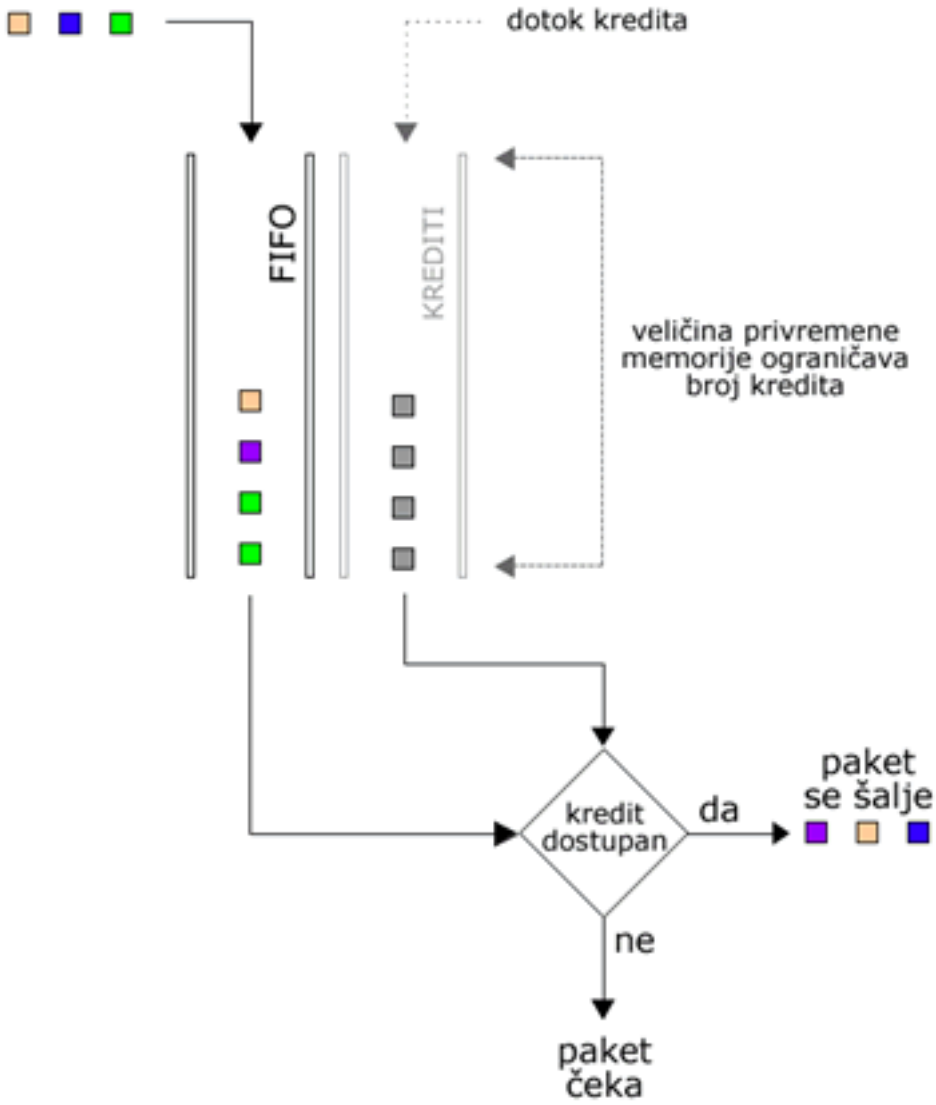
```
#ifconfig eth0 txqueuelen 10
```

Njegova trenutna vrijednost se može se provjeriti s *ifconfig* i *ip* naredbama.

Metoda filtra s kreditima
Metoda filtra s kreditima (Token Bucket Filter – TBF) je jednostavna metoda za upravljanje redom koja radi na način da propušta samo one pakete koji pristižu na mrežno sučelje ne većom brzinom od određene, uz mogućnost kratkotrajnih prekoračenja (*burst*). Ova je metoda prilično precizna i koristi malo procesorskog vremena tako da bi trebala biti prvi izbor u slučaju da se mrežno sučelje želi samo usporiti. Implementacija TBF metode sastoji se od privremene memorije (*bucket*) koja se konstantno puni imaginarnim podacima - kreditima za slanje (*token*) nekom određenom brzinom (*token rate*). Najvažniji parametar je veličina privremene memorije tj. broj imaginarnih podataka koji mogu biti pohranjeni u nju. Svaki kredit koji pristigne pokupi jedan dolazeći paket iz toka podataka i onda se briše iz privremene memorije. Povezujući ovaj algoritam s dva toka - podataka i kredita, dolazi se do tri moguće kombinacije:
- podaci stižu u TBF jednakom brzinom kao i krediti - u ovom slučaju svaki dolazeći paket ima svoj odgovarajući kredit i paket prolazi bez zadržavanja
- podaci stižu i TBF brzinom manjom od brzine kredita - samo dio kredita pokupi dolazeće pakete i krediti se akumuliraju do veličine privremene memorije - akumulirani
se krediti mogu iskoristiti za kratkotrajno slanje paketa brzinom većom od standardne brzine dolaska kredita (*burst*)
- podaci stižu u TBF brzinom većom od brzine kredita - ovo znači da će privremena memorija vrlo brzo ostati bez kredita i paketi ostaju u redu čekanja - u tom slučaju se

TBF počinje gušiti i ako paketi nastave stizati brzinom većom od brzine kredita, dolazeći paketi će se brisati (*drop*)

Shema TBF metode je prikazana na slici 2.



U samoj implementaciji krediti predstavljaju oktete, a ne pakete.
Tri parametra koja su uvijek dostupna za upotrebu uz TBF (neovisno o količini slobodnih krediti) su:
- limit ili latencija - limit je broj okteta koji mogu čekati slobodne kredite, a latency je maksimalno vrijeme koje paket može čekati slobodni kredit unutar TBF-a
- burst/buffer/maxburst - veličina privremene memorije u oktetima - ovo je najveća vrijednost u oktetima za koju će biti slobodnih kredita
- mpu - Minimal packet Unit određuje najmanju veličinu kredita za jedan paket, potrebno iz razloga što paket veličine nula okteta zauzima ne manje od 64 okteta
pojasne širine

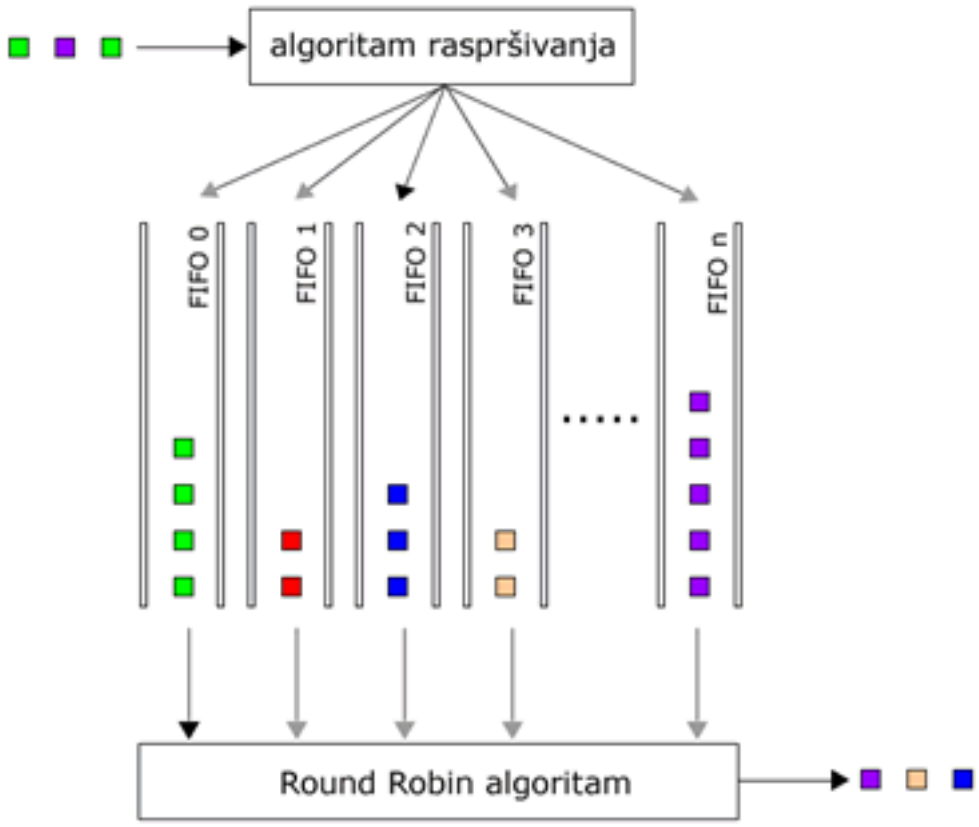
Ostali parametri su:
- rate - uzima se u obzir pri računanju maksimalnog vremena čekanja paketa
- peakrate - također se uzima u obzir pri izračunavanju maksimalnog vremena paketa, a služi pri određivanju brzine pražnjenja privremene memorije, maksimalno
1mbit/s zbog Linux ograničenja
- mtu/minburst - efektivno znači stvaranje nove privremene memorije, zbog ograničenja peakrate parametra

Primjer konfiguracije:

```
#tc qdics add dev ppp0 root tbf rate 200kbit latency 50ms burst 1540
```

Ovaj je primjer vrlo koristan u slučaju da se želi spojiti jedan mrežni uređaj s velikim redom (kao npr. DSL modem) na drugi, vrlo brzi mrežni uređaj (npr. mrežna kartica). U slučaju odlaznog prometa performanse uređaja (modema) drastično padaju, a razlog je taj što se red odlaznih paketa u modemu puni puno brže nego je uređaj sposoban poslati, a paketi koji pristižu ne mogu doći na red. Gornja linija smanjuje red odlaznih paketa, ali ne u samom uređaju (modemu), nego u jezgri operativnom sustava, tj. na mjestu gdje ga možemo kontrolirati. Brzinu od 200kbit treba zamijeniti stvarnom brzinom odlaznog prometa umanjenom za nekoliko postotaka.

Metoda stohastičnog pravednog reda
Metoda stohastičnog pravednog reda (Stohastic Fairness Queuing - SFQ) je jednostavna implementacija metode za upravljanje iz cijele obitelji ravnopravnih metoda za upravljanje. Manje je precizna od ostalih, ali je mnogo brža (potrebno je manje kalkulacija) uz zadržavanje jednakosti prema svim paketima.
Glavna karakteristika SFQ metode je tok koji se podudara sa TCP sesijom ili UDP tokom. Mrežni je promet podijeljen u velik broj FIFO redova, po jedan za svaki tok. Onda se on dalje preusmjerava tzv. Round Robin algoritmom, tj. svakom toku se naizmjenično daje jednaka šansa za slanje paketa. To rezultira vrlo ravnopravnim ponašanjem i isključuje mogućnost da jedan tok bude prioritetan. Ova metoda ne alocira red za svaki tok nego ima ugrađen algoritam raspršivanja koji raspodjeljuje promet preko konačnog broja redova. Međutim, kako je moguće da raspršivanje jedan tok bude više zastupljen od drugih, sami algoritam se mijenja često i nasumice i odatle naziv "stohastičan".
SFQ metoda je korisna samo u slučaju kad je mrežno sučelje stalno opterećeno do maksimuma jer, u suprotnom, nema reda i nema efekta koji proizlazi iz čekanja paketa i raspodjele njihovog redoslijeda. Ravnopravnost algoritma ovisi o broju redova i to na način da veći broj redova znači veću ravnopravnost. Shema SFQ metode je prikazana na slici 3.



Parametri SQF metode su:

- perturb - broj sekundi nakon kojeg će se promijeniti algoritam, preporučena vrijednost je 10 sekundi - ako se ne postavi, algoritam se nikad neće promijeniti
- quantum - količina okteta koja određuje koliko se tok može isprazniti prije nego sljedeći red počne s pražnjenjem
- limit - ukupni broj paketa koji mogu stati u red nakon kojeg ih SFQ počne brisati

Primjer konfiguracije:

```
# tc qdisc add dev ppp0 root sfq perturb 10
# tc –s –d qdisc ls qdisc sfq 800c: dec ppp0 quantum 1514b limit 12p flows 128/1024 p
erturb 10sec
```

Parametar 800c je automatski kreirana jedinstvena oznaka (handle), a limit 128 postavlja maksimalni broj paketa koji mogu čekati u redu. Ova konfiguracija je prikladna za slučaj da se želi postići ravnopravnost mrežnog prometa na računalu koje ima mrežno sučelje iste pojasne širine kao i mrežna veza na koju se spojeno, npr. obični modem.

Random Early Detection

Ova se metoda koristi pri upravljanju redovima čekanja na glavnim linkovima (backbone linkovima) koji imaju više od 100Mb pojasne širine
Normalni način rada usmjernika (routera) na internetu se zove tail-drop, a očituje se u stvaranju reda čekanja do neke veličine, a sav mrežni promet koji prelazi tu veličinu se briše. Taj je način vrlo nepravedan i može dovesti do retransmisijske sinkronizacije. To je pojava kad izbrisani mrežni promet, koji nije stao red čekanja usmjernika, uzrokuje zakašnjelu retransmisiju koja prepuni već zagušeni red čekanja. Da se se backbone usmjernici mogli nositi s kratkotrajnim zagušenjima, obično imaju implementirane velike redove čekanja. Međutim, iako su veliki redovi čekanja dobri za propusnost, mogu prilično povećati latenciju i uzrokovati praskovito ponašanje TCP protokola prilikom zagušenja.
Problemi s tail-drop-om na internetu se povećavaju jer raste upotreba aplikacija koje nisu "prijateljski raspoložene" prema mreži. Tu uskače Linux kernel koji nudi RED, Random Early Detection. Neki ga zovu i Random Early Drop jer taj naziv ukratko opisuje njegov način rada. Sami RED nije potpuno rješenje za cijeli tail-drop problem jer aplikacije koje loše implementiraju proces prilagođavanja retransmisije (backoff procedure) još uvijek mogu dobiti veći postotak pojasne širine, ali s RED metodom ne utječu toliko propusnost i latenciju ostalih konekcija. RED statistički briše pakete iz svih tokova podataka prije nego se red čekanja u potpunosti napuni. Na taj se način backbone link usporava na suptilniji način i sprječava se retransmisijska sinkronizacija. RED također pomaže TCP protokolu u pronalaženju pravedne brzine na način da se neki paketi brišu ranije, što drži red čekanja manjim i latenciju pod kontrolom. Vjerojatnost da se neki paket izbriše iz određene konekcije je proporcionalan pojasnoj širini koju ta konekcija koristi, a ne broju paketa koje šalje. RED je najbolji za upotrebu na backbone linkovima gdje se ne može priuštiti kompleksnost praćenja pojedine konekcije radi pravednog upravljanja redom čekanja.

Parametri kojima je moguće kontrolirati RED moetodu su:
- min - određuje minimalnu veličinu reda, izraženu u bajtovima, nakon koje će se paketi početi brisati
- max - određuje tzv. soft maximum, vrijednost koju će algoritam pokušati ne prijeći
- burst - određuje maksimalni broj paketa koji mogu proći u kratkotrajnom prekoračenju
- limit - vrijednost koja predstavlja sigurnosnu točku, iza koje će RED preći u tail-drop način rada
- avpkt - prosječna veličina paketa

Parametar min se može izračunati na način da se najveća prihvatljiva latencija pomnoži s dostupnom pojasnom širinom. Ako se parametar postavi na premalenu vrijednost, smanjit će se propusnost, a prevelika vrijednost će smanjiti latenciju. Parametar max bi trebao biti najmanje dva puta veći od min vrijednosti. Parametar burst određuje ponašanje RED algoritma u situacijama kratkotrajnih prekoračenja. On bi trebao biti veći od vrijednosti dobivene dijeljenjem min i avpkt parametara. Parametar limit se obično postavlja na vrijednost osam puta veću od max parametra, dok parametar avpkt postavljen na vrijednost 1000 radi uredno na linkovima koji imaju MTU od 1500 bajtova.

Zaključak

Sve besklasne metode su jednostavne metode koje upravljaju mrežnim prometom na način da ga

preraspodjeljuju, usporavaju (tj. unose kašnjenje) ili brišu. Ukratko:
- za jednostavno usporavanje mrežnom prometa upotrijebiti TBF metodu koja pouzdano radi na velikim pojasnim širinama ako se dobro odredi veličina privremene memorije (bucket)
- u slučaju da je link konstantno maksimalno opterećen, a želja je da sav mrežni promet bude ravnopravan, upotrijebiti SFQ metodu
- ako ste vlasnik backbone linka i znate što radite, upotrijebite RED
- ako se mrežni promet prosljeđuje, upotrijebiti TBF na sučelju na koje se promet proslijeđuje

Linkovi:
Linux Advanced Routing and Traffic Control [1]
RED Gateways for Congestion Avoidance [2]

- Logirajte [3] se za dodavanje komentara

pet, 2007-11-30 13:19 - Mirko Lovričević**Kategorije:** Mreža [4]
Vote: 0

No votes yet

Source URL: https://sysportal.carnet.hr/node/320

Links
[1] http://lartc.org
[2] http://www.icir.org/floyd/papers/red/
[3] https://sysportal.carnet.hr/sysportallogin
[4] https://sysportal.carnet.hr/taxonomy/term/29