praktyczne zastosowania mechanizmów QoS, Linuxowe HTB

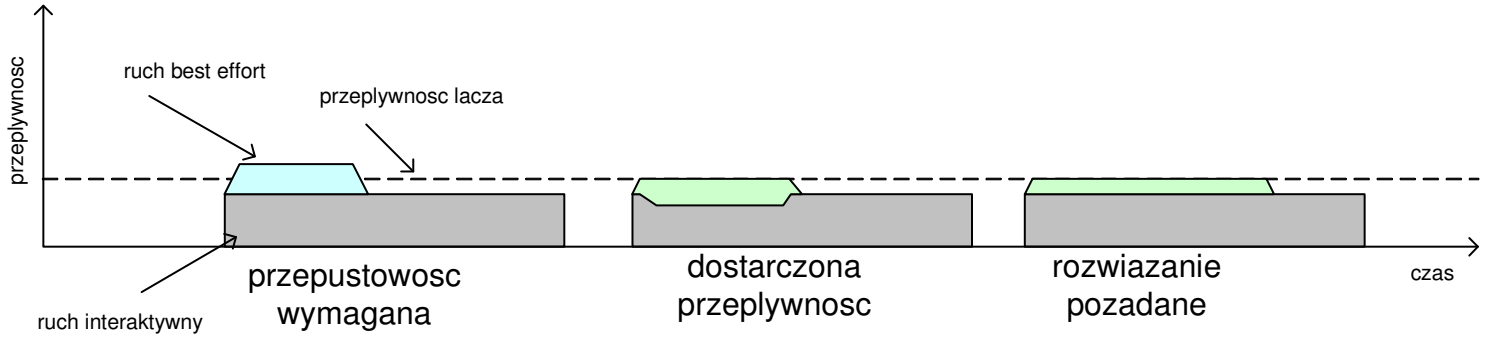
paweł kudzia

QoS czyli..

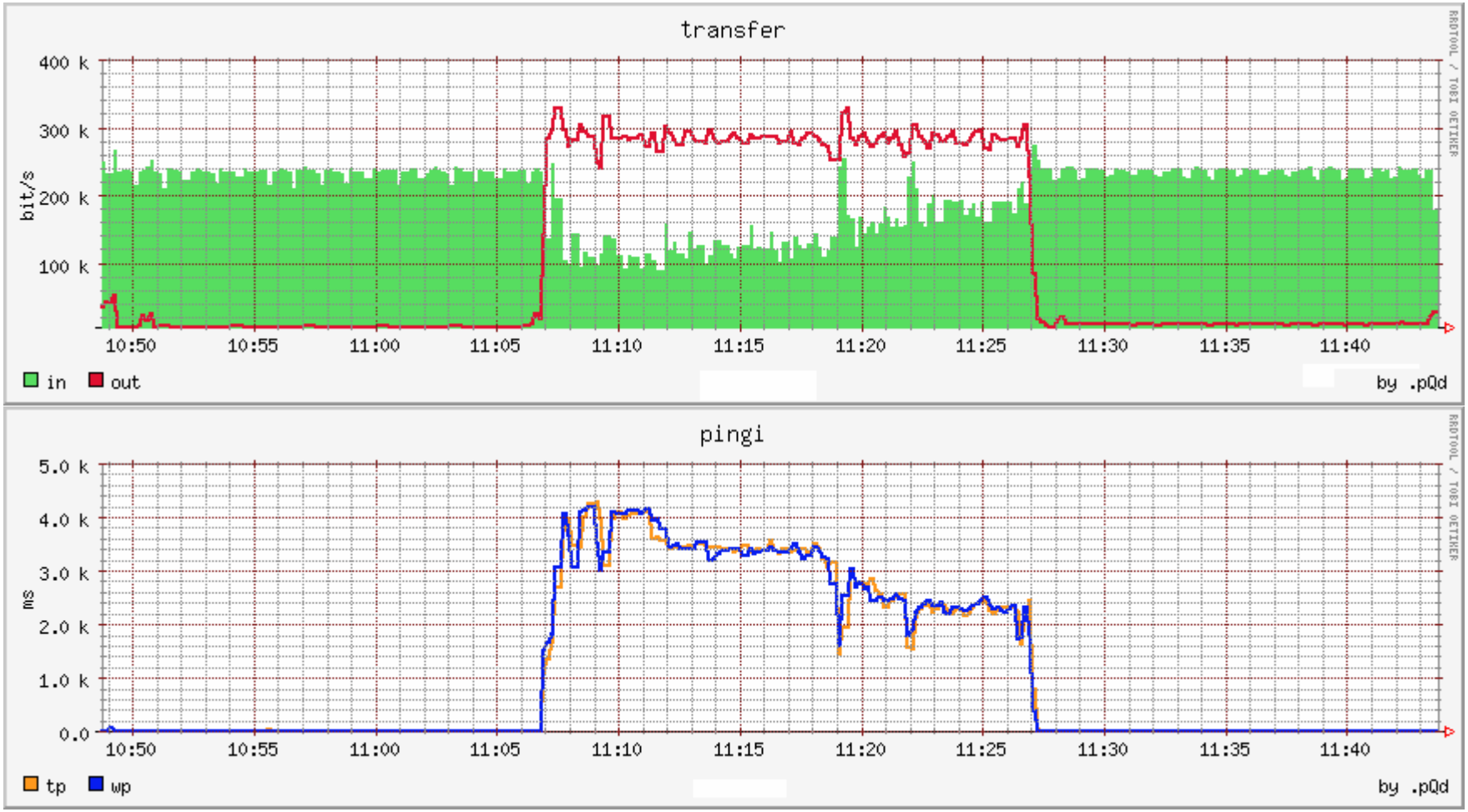
racjonalne wykorzystanie dostępnego zasobu poprzez wyróżnienie klas ruchu traktowanych w różny sposób.

dostosowanie przepływności dostarczanej poszczególnym klasom usług do faktycznych potrzeb z uwzględnieniem pewnych priorytetów.

problemy z obsługą best effort

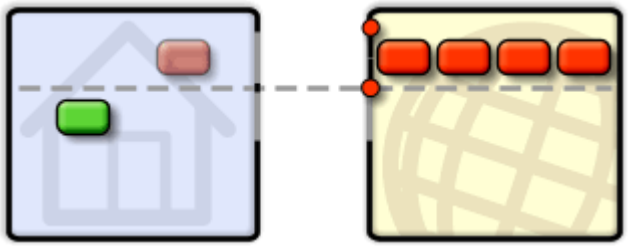


wpływ uplaodu na download

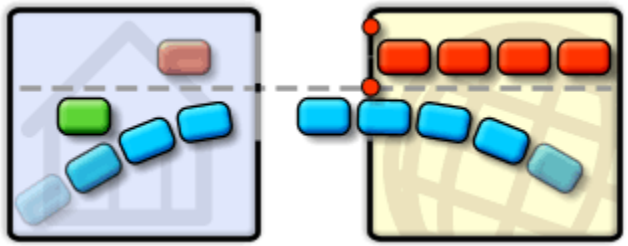


problem opóźnionych ACK

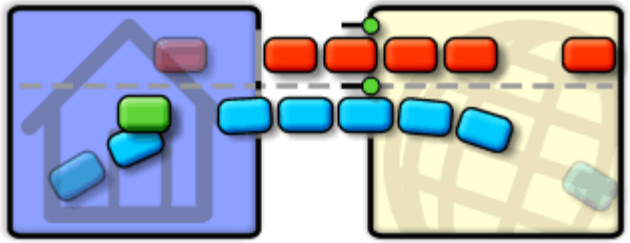
sam download:



download i upload:

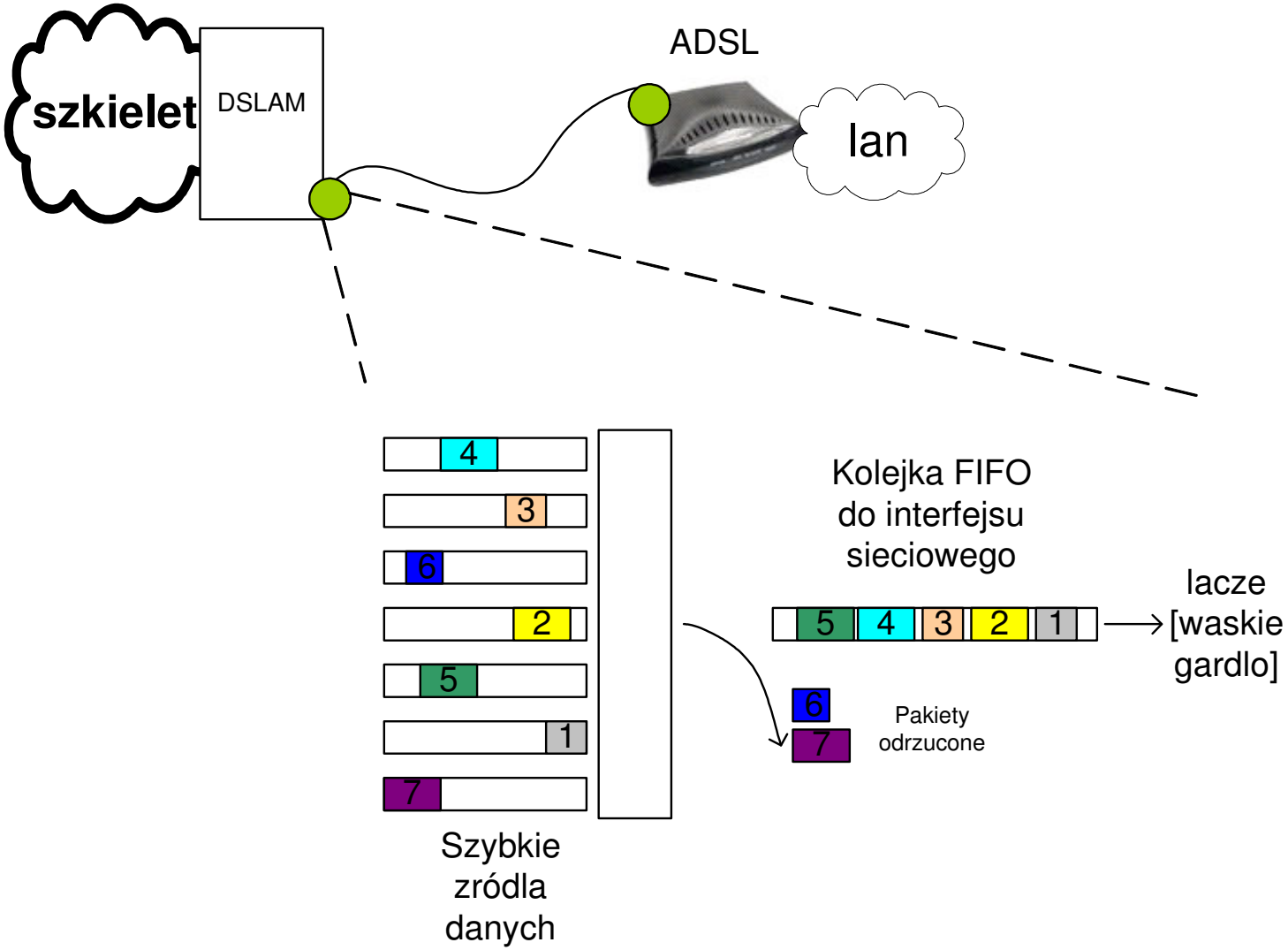


download i upload z priorytetyzacją ACK:



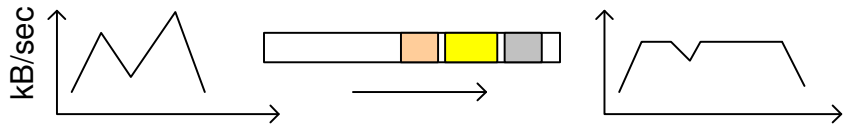
Animacja z www.cfos.de

problemy z kolejką fifo

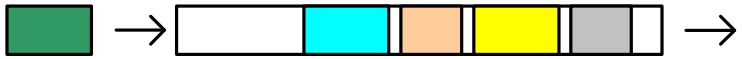


kolejki…

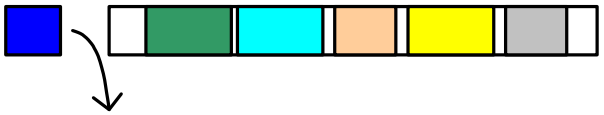
+ buforują dane napływające zbyt szybko



- …ale wprowadzają **opóźnienie** [zmienne]



- ..i mają skończony rozmiar.
  po zapełnieniu kolejki pakiety są **odrzucane.**
  nie można przewidzieć, które pakiety zostaną odrzucone.

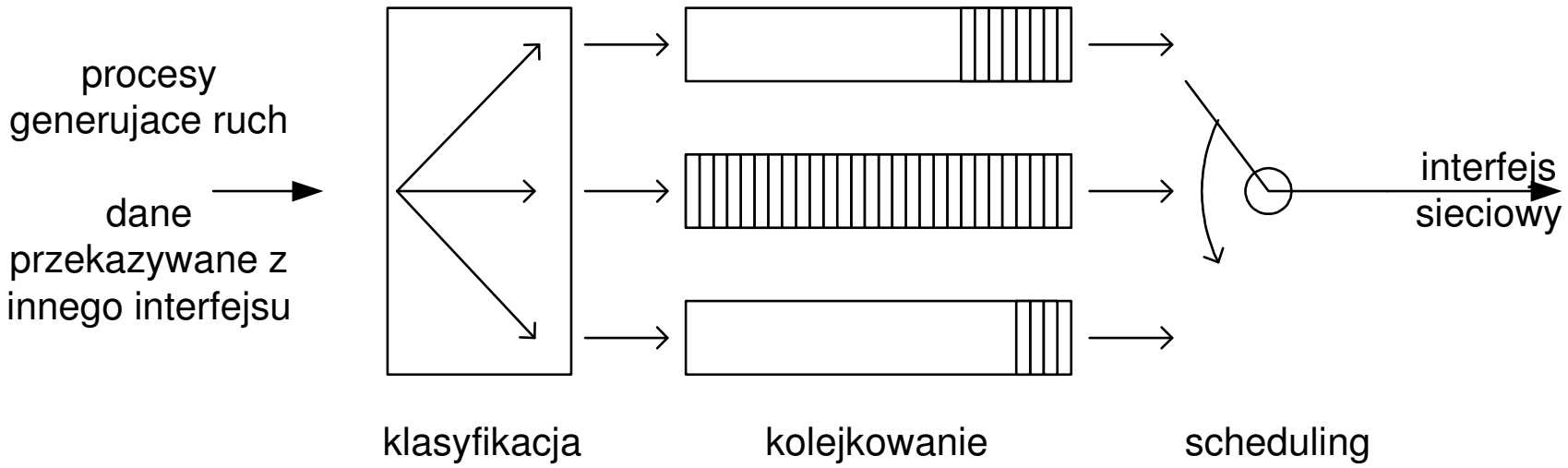


takie rozwiązanie nie jest ‘przyjazne’ dla protokołu TCP [lawinowe retransmisje z wielu źródeł w momencie gdy pakiety zaczynają być odrzucane].
rozwiązanie? RED.

parametry QoS

- **przepływność** – odpowiedniki CIR i EIR, priorytet przydzielania pasma ponad CIR
- opóźnienie
- zmienność opóźnienia [jitter]
- straty pakietów

QoS - etapy


procesy
generujace ruch
dane
przekazywane z
innego interfejsu
interfejs
sieciowy
klasyfikacja
kolejkowanie
scheduling

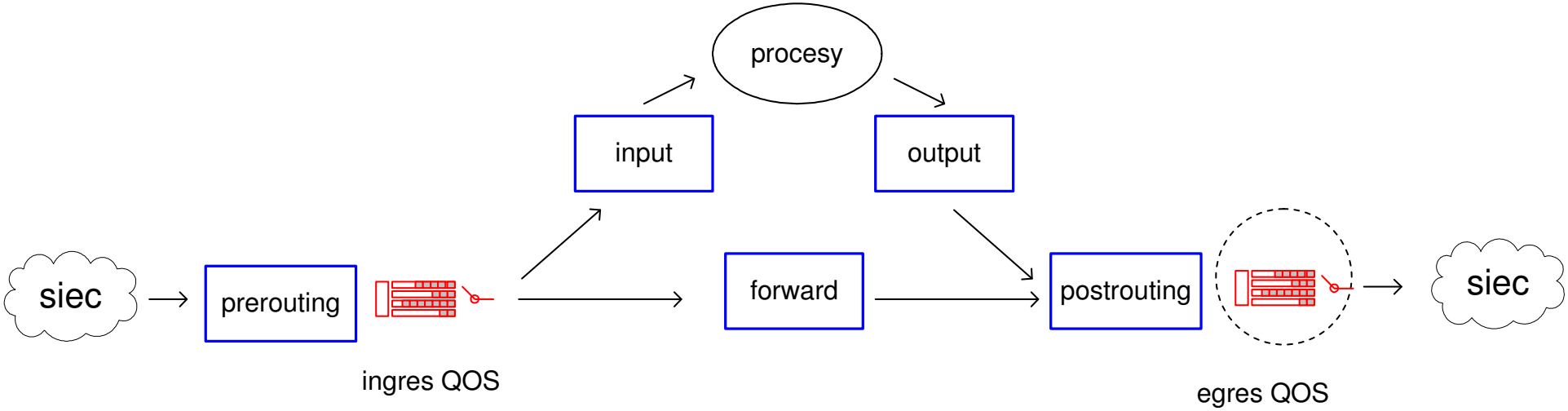
- klasyfikacja
  - na poziomie zawartości pakietu [zazwyczaj pola nagłówka] przyporządkowuje pakiet do wybranej klasy obsługi
- kolejkowanie
  - przechowywanie pakietów w oczekiwaniu na transmisję
- scheduling
  - wybór kolejki z której pakiet zostanie pobrany i wysłany ‘do sieci’.

kolejkowanie i scheduling często są łączone razem i nazywane dyscyplinami kolejkowania [‘queuing disciplines’].

mechanizmy dostępne w Linuxie

- elastyczne klasyfikatory
  - pola nagłówka [porty, źródło, cel, TOS].
  - reguły opisane poleceniem **`tc`** i **`iptables`**
- rozmaite dyscypliny kolejkowania :
  - bezklasowe [FIFO, proste ograniczenie przepływności - TBF, sprawiedliwe – SFQ],
  - oparte o klasy, pozwalające budować struktury drzewiaste [CBQ, HTB]

droga pakietu w jądrze Linuxa



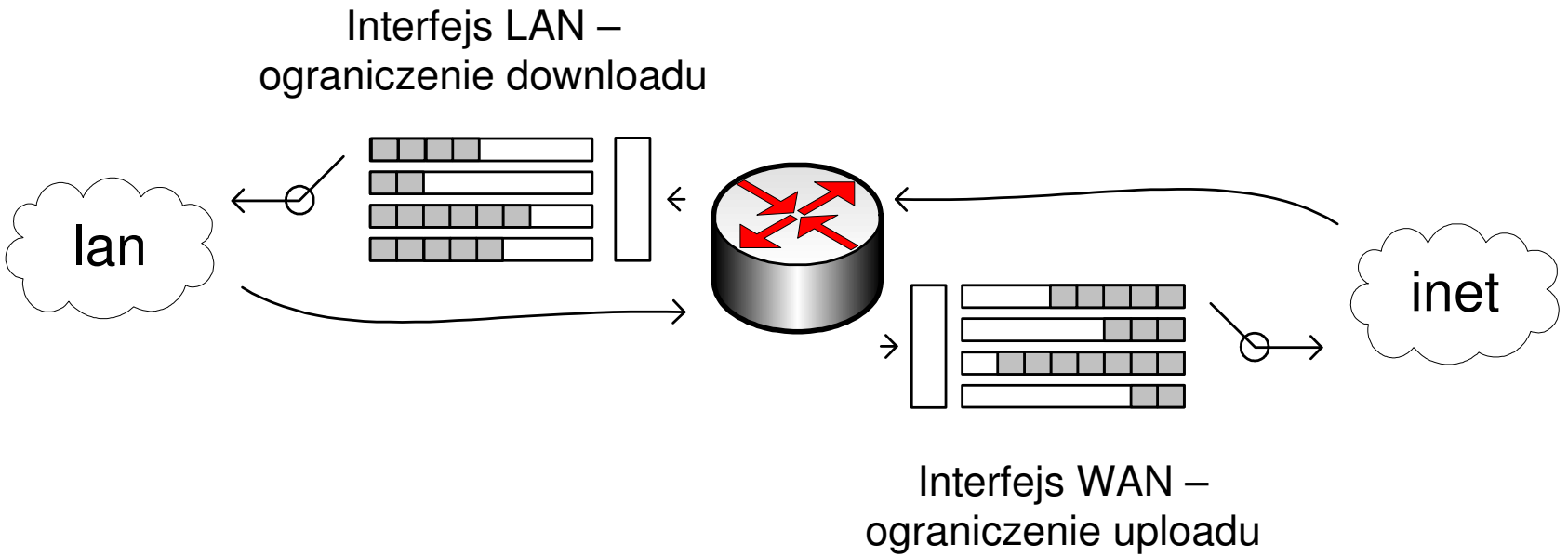
filtr pakietów – kontrola poprzez **`iptables`**
QoS – kontrolowany poleceniem **`tc`**

Na podstawie www.docum.org

miejsce kształtowania ruch

standardowe jądra dostarczają znacznie bogatszy zestaw narzędzi do kształtowania ruchu wychodzącego [egress] niż przychodzącego [ingress].

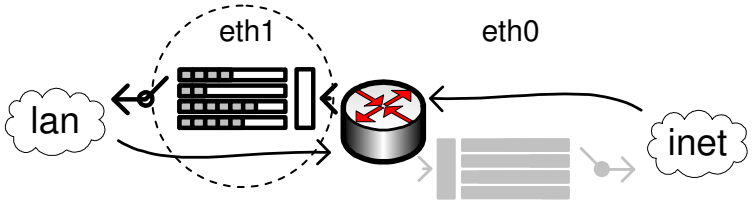
najczęściej ogranicza się ruch wychodzący z routera.

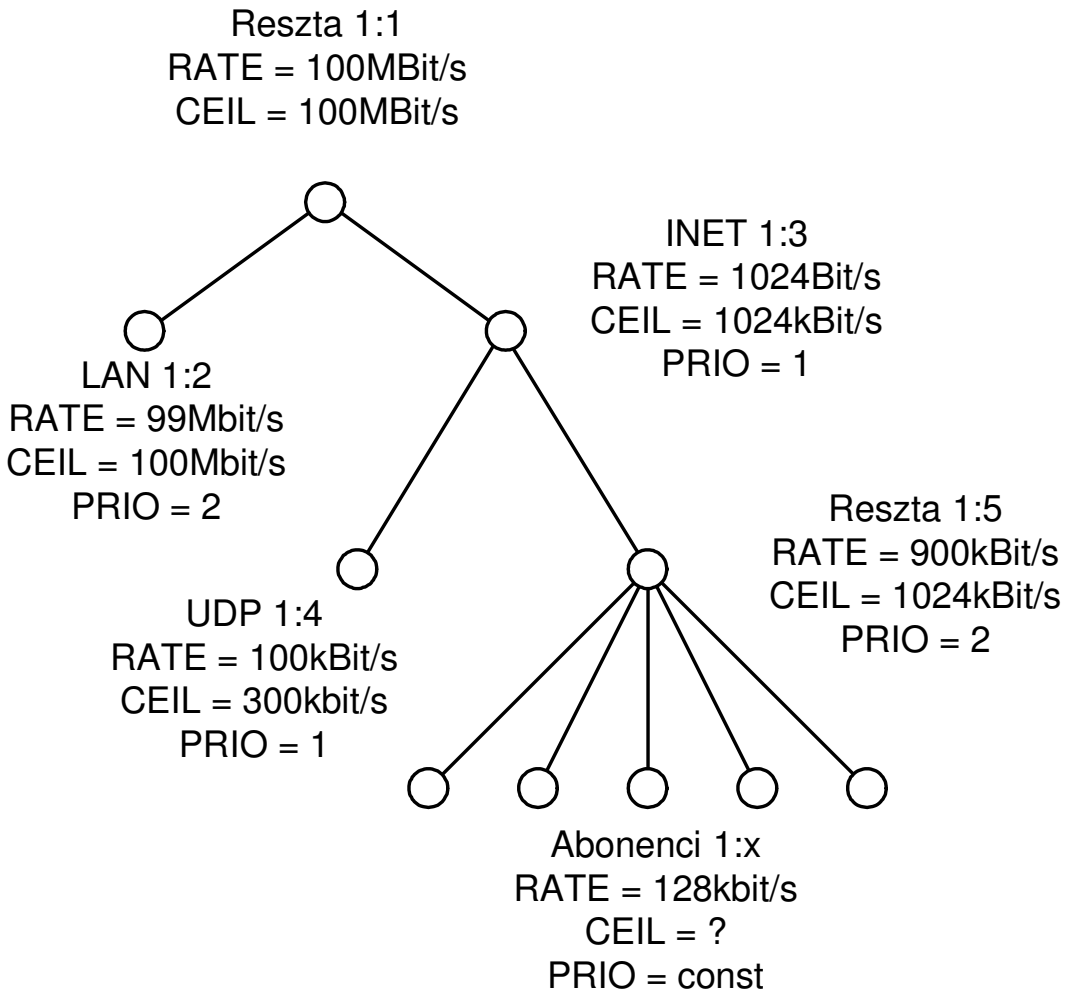


konkrety - HTB

- klasy powiązane ze sobą hierarchicznie
- korzeniem drzewa jest interfejs sieciowy
- klasy-dzieci mogą ‘pożyczać’ między sobą niewykorzystane pasmo
- parametry QoS opisujące klasę
  - **RATE**
  - **CEIL**
  - **PRIO**

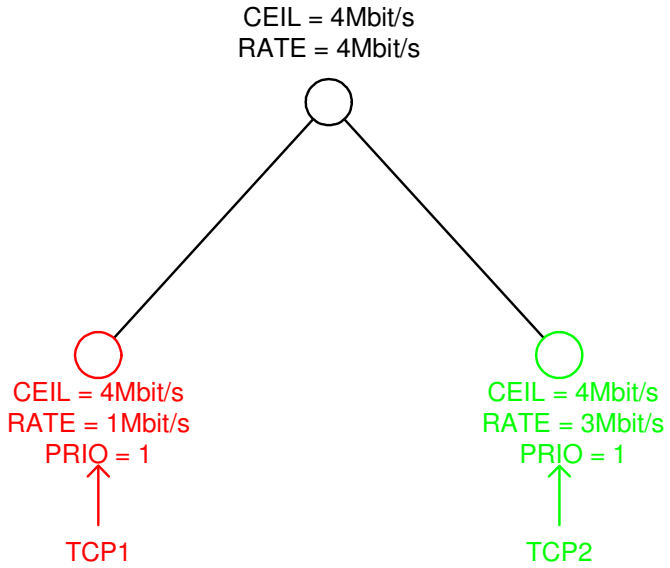
drzefko

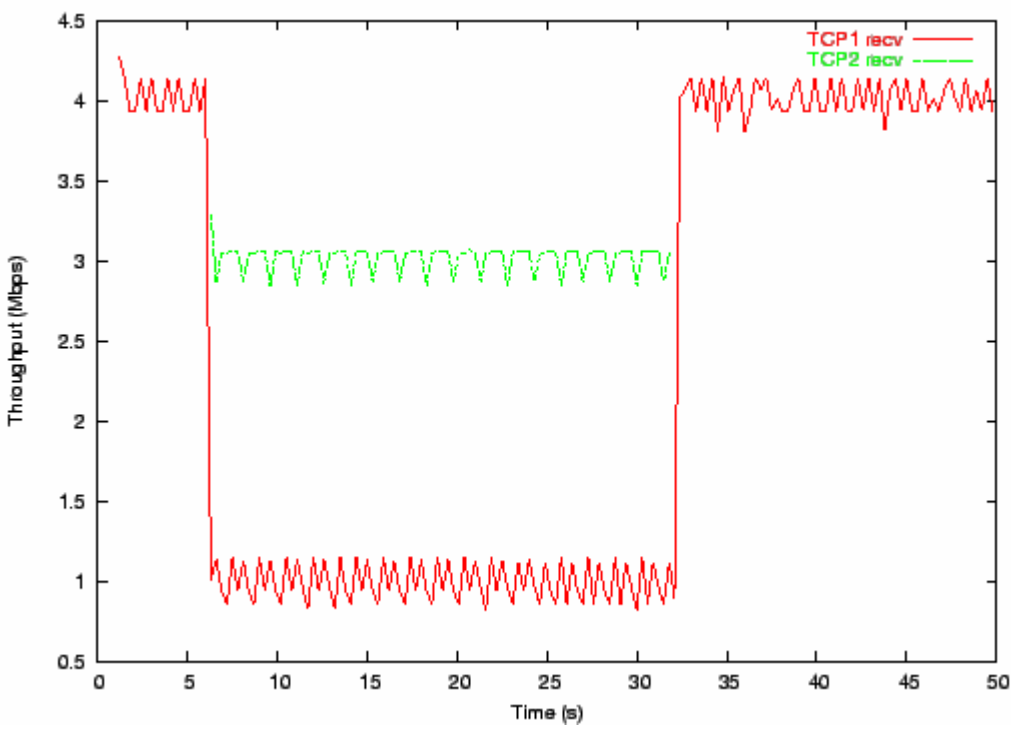

eth1
eth0
lan
inet


Reszta 1:1
RATE = 100MBit/s
CEIL = 100MBit/s
INET 1:3
RATE = 1024Bit/s
CEIL = 1024kBit/s
PRIO = 1
LAN 1:2
RATE = 99Mbit/s
CEIL = 100Mbit/s
PRIO = 2
Reszta 1:5
RATE = 900kBit/s
CEIL = 1024kBit/s
PRIO = 2
UDP 1:4
RATE = 100kBit/s
CEIL = 300kbit/s
PRIO = 1
Abonenci 1:x
RATE = 128kbit/s
CEIL = ?
PRIO = const

‘pożyczanie’ pasma

- występuje pomiędzy klasami mającymi tego samego rodzica
- im niższe **PRIO** tym wyższy priorytet przydzielania pasma

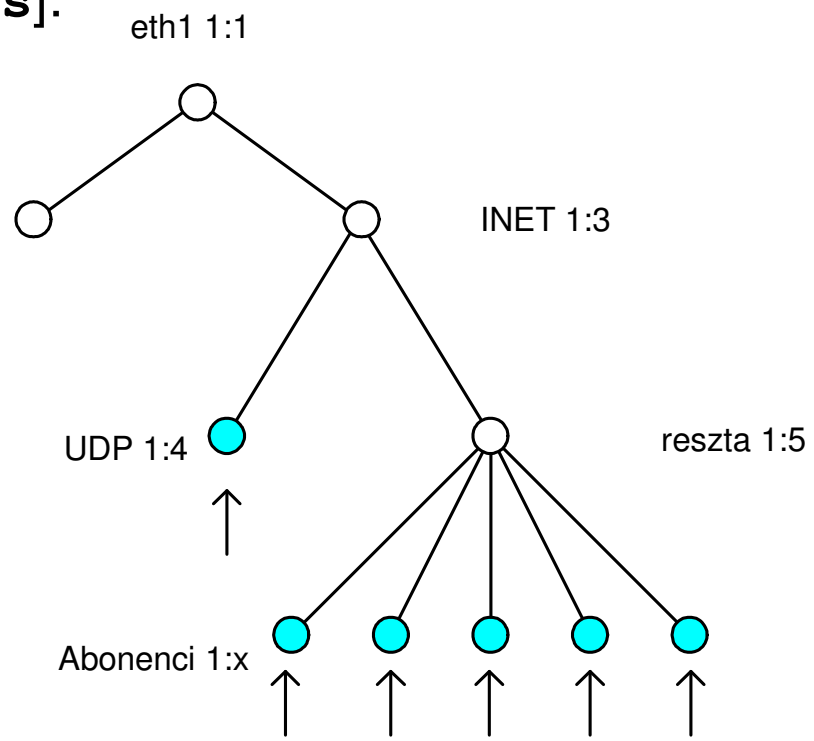




Na podstawie http://lifc.univ-fcomte.fr/~fuin

klasyfikacja

- przydzielanie pakietów do konkretnych klas na podstawie zdefiniowanych filtrów
- do filtrowania możemy wykorzystać :
  - filtr u32 kontrolujący pola nagłówka [**`tc`**]
  - znaczniki [marks] nadawane w tablicach **`mangle`** lub nowy cel ‘CLASSIFY’ w łańcuchach [**`iptables`**]:
    - **`PREROUTING`** [tylko znczniki]
    - **`POSTROUTING`**,
    - **`FORWARD`**,
    - **`OUTPUT`**,



klasyfikacja - przykłady

- iptables

```
iptables \
-t mangle -A POSTROUTING \
-i eth0 -s 81.2.3.4 [.inne.filty.] \
-j CLASSIFY --set-class 1:3
```

- tc:

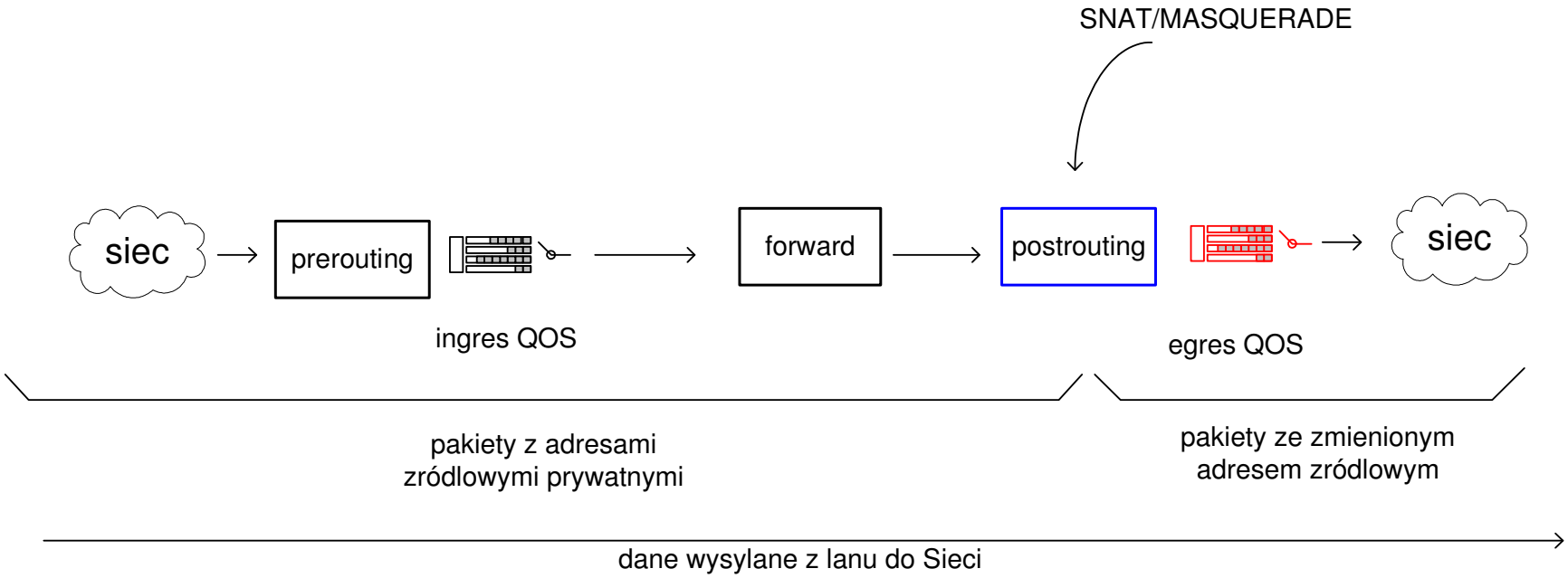
```
tc filter add dev eth0 parent 1:0 protocol ip \ prio
100 u32 match ip src 81.2.3.4 classid 1:3
```

- iptables [mark] + tc:

```
iptables -t mangle -A PREROUTING \
-i eth0 -s 81.2.3.4 \
-j MARK --set-mark 1234

tc filter add dev eth0 parent 1:0 protocol ip \ prio
2 handle 1234 fw classid 1:3
```

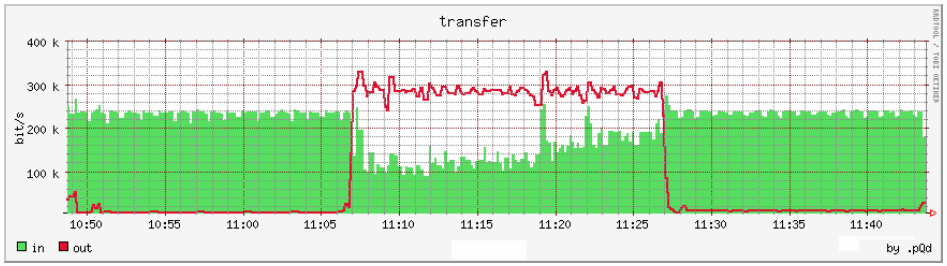
po co ten mark ?



- w przypadku stosowania translacji adresów źródłowych [NAT'a] informacja o adresie nadawcy z siecilokalnej tracona jest w po przejściu pakietu przez łańcuch postrouting tablicy nat
- kolejki egres widzą pakiety ciągle z tym samym adresem źródłowym [publicznym].
- rozwiązanie – znakowanie pakietów [znacznik przechowywany jest w pamięci systemu aż do opuszczenia pkt interfejsem sieciowym].

przykład praktyczny

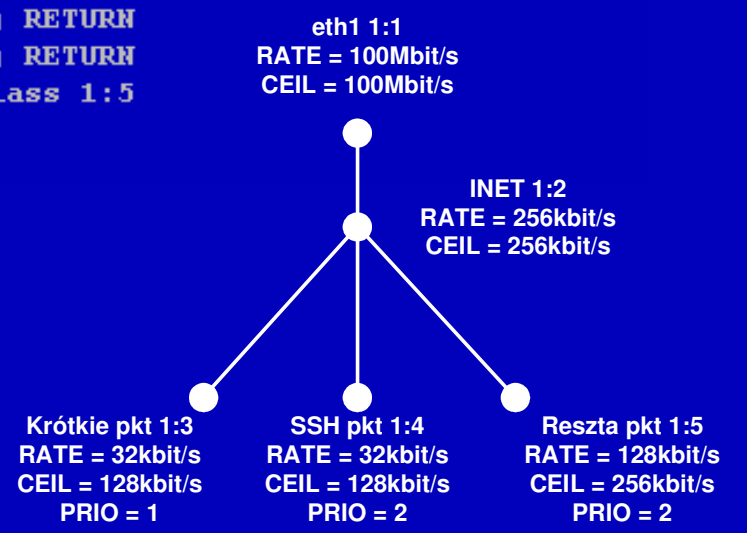
- cel: likwidacja efektu wysyconego uploadu



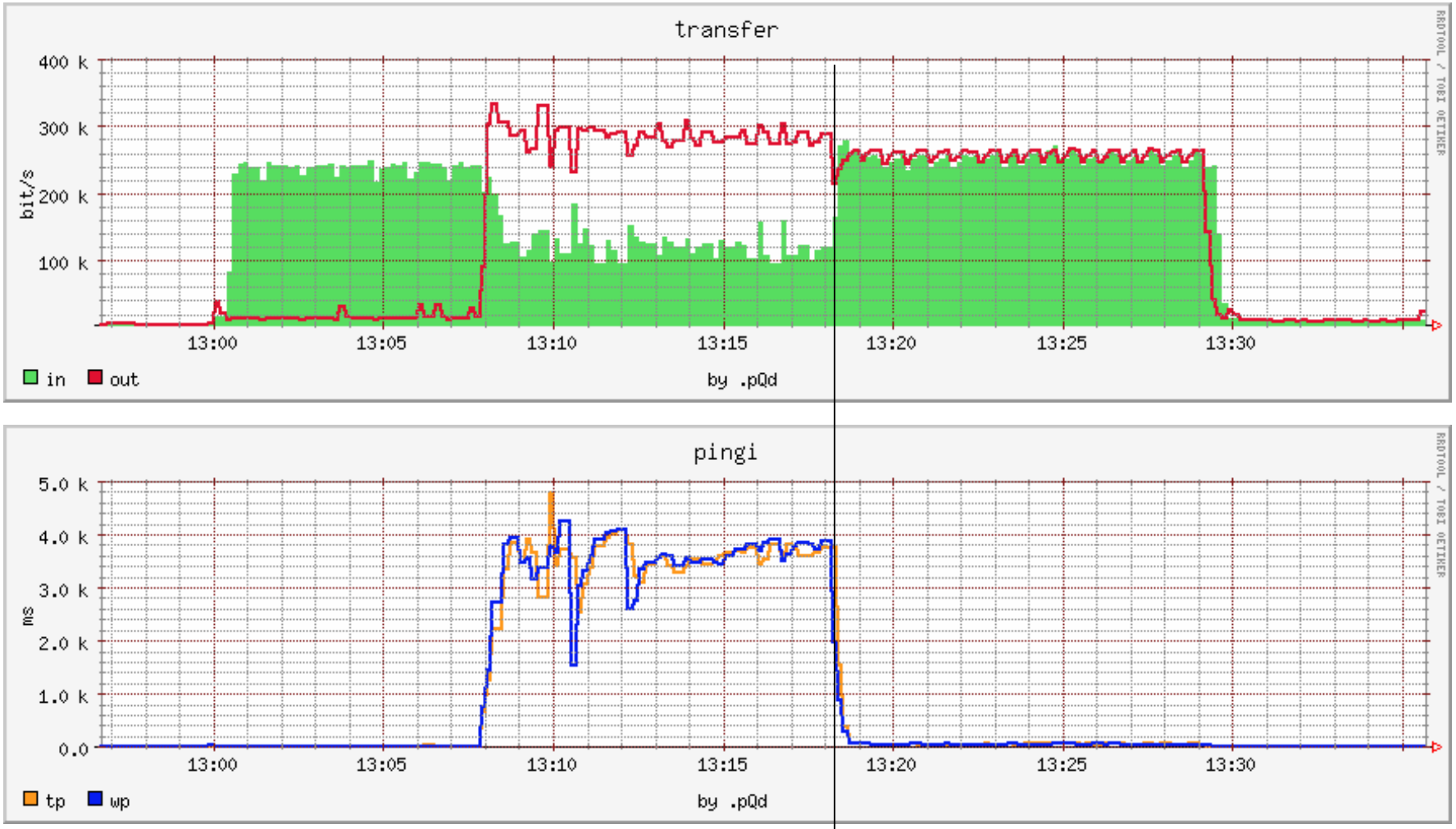
- sposób realizacji:
  - wyróżnienie kolejki o wyższym priorytecie dla małych pakietów [w szczególności pakietów z ustawioną flagą ACK, nie niosących danych, VoIP].
  - wyróżnienie kolejki dla ruchu o podwyższonym priorytecie [ssh]
  - wyróżnienie kolejki dla pozostałego ruchu [reszta uploadu]

skrypt

```
#!/bin/bash
tc qdisc del root dev eth1
tc qdisc add dev eth1 root handle 1:0 htb default 5
tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit ceil 100Mbit
# class for upload
tc class add dev eth1 parent 1:1 classid 1:2 htb rate 256kbit ceil 256kbit
# hi priority [short packets, including ack/syn/fin]
tc class add dev eth1 parent 1:2 classid 1:3 htb rate 32kbit ceil 128kbit prio 1
# medium priority [ssh]
tc class add dev eth1 parent 1:2 classid 1:4 htb rate 32kbit ceil 64kbit prio 2
# lo priority [everything else]
tc class add dev eth1 parent 1:2 classid 1:5 htb rate 128kbit ceil 256kbit prio 2
# attach queing disciplines
tc qdisc add dev eth1 parent 1:3 sfq perturb 10
tc qdisc add dev eth1 parent 1:4 sfq perturb 10
tc qdisc add dev eth1 parent 1:5 sfq perturb 10
# filter traffic
iptables -t mangle -F POSTROUTING
iptables -t mangle -A POSTROUTING -o eth1 -m length --length 0:128 -j CLASSIFY --set-class 1:3
iptables -t mangle -A POSTROUTING -o eth1 -m length --length 0:128 -j RETURN
iptables -t mangle -A POSTROUTING -o eth1 -p tcp --sport 22 -j CLASSIFY --set-class 1:4
iptables -t mangle -A POSTROUTING -o eth1 -p tcp --dport 22 -j CLASSIFY --set-class 1:4
iptables -t mangle -A POSTROUTING -o eth1 -p tcp --sport 22 -j RETURN
iptables -t mangle -A POSTROUTING -o eth1 -p tcp --dport 22 -j RETURN
iptables -t mangle -A POSTROUTING -o eth1 -j CLASSIFY --set-class 1:5
iptables -t mangle -A POSTROUTING -o eth1 -j RETURN
```

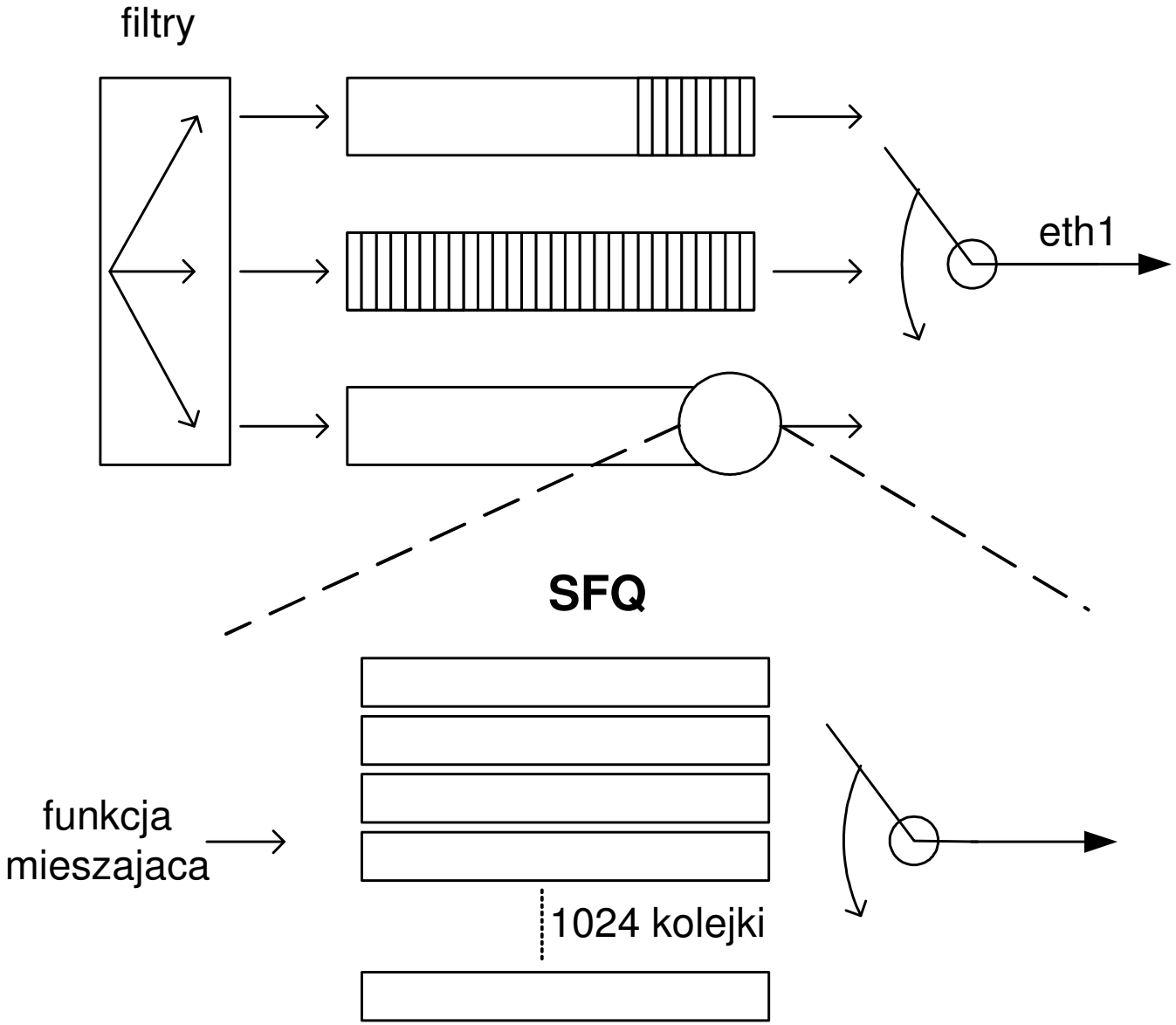


rezultat

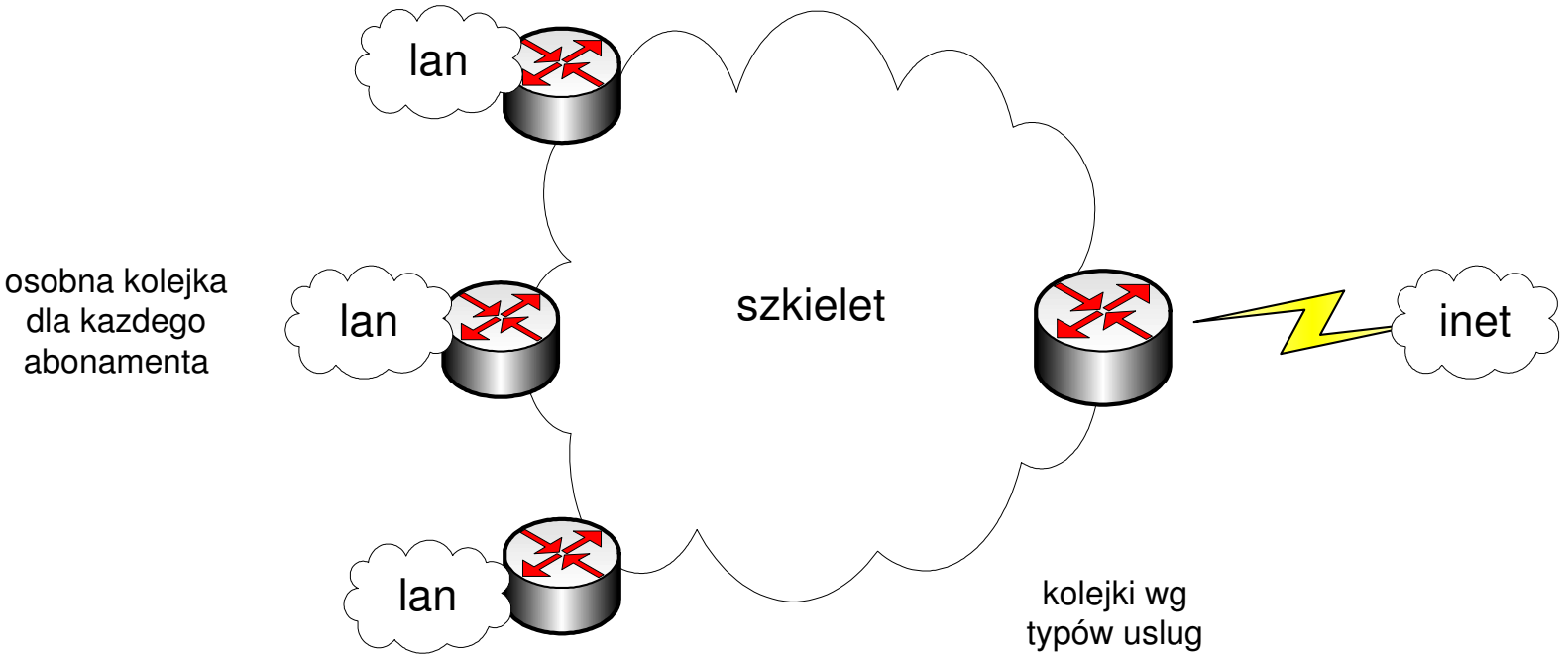


moment uruchomienia QoS'a priorytetyzującego krótkie pakiety

SFQ - sprawiedliwość w obrębie jednej kolejki



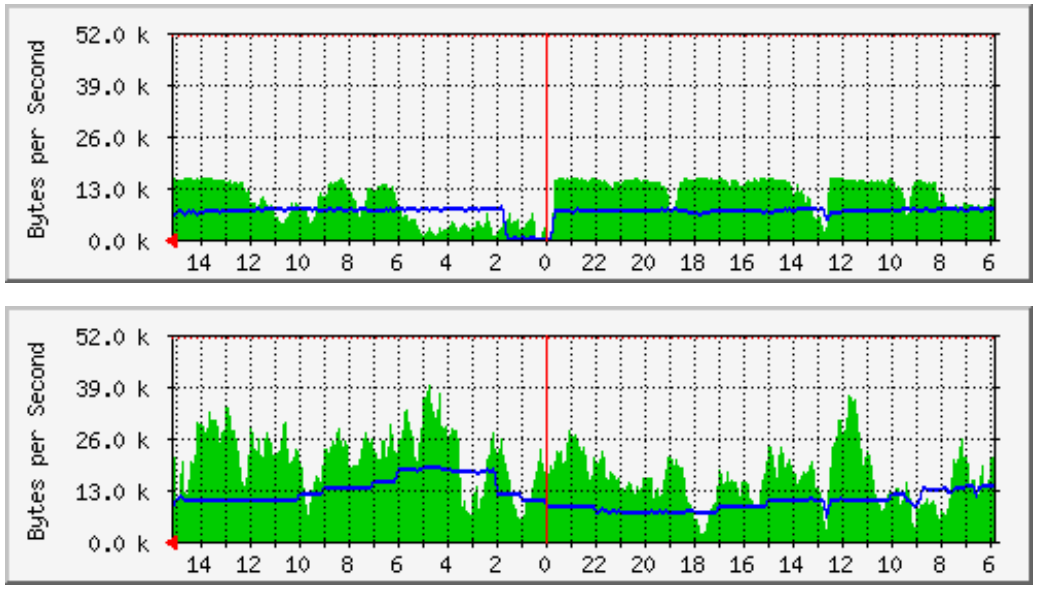
inne przykłady zastosowania



routery dostępowe

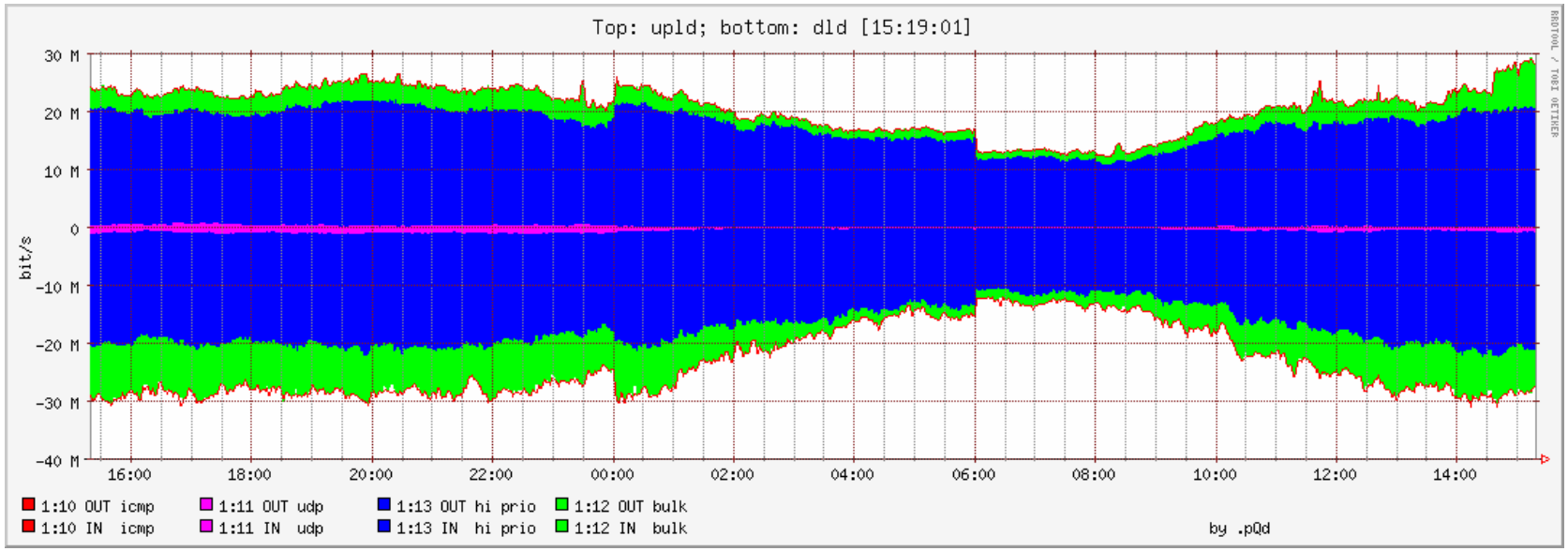
kolejki wynikające z typu abonamentu.

[manipulacjia wartościami CEIL i PRIO]



router brzegowy

- wyróżniono 4 klasy ruchu:
  - ruch TCP z portem źródłowym/docelowym poniżej 1024 lub portem 3389 [RDP]
  - ruch icmp lub pakiety < 128B
  - ruch UDP
  - cała reszta.
- parametry RATE, CEIL i PRIO zostały dobrane empirycznie / na podstawie obserwacji.



routery dostępowe

P4 / Athlony 2-3GHz, 512MB ram, Linux 2.6.8-2.6.11

- routing [~50 tras, OSPF]
- hub DC++, statystyki, w3cache
- QoS dla łącze **2-16Mbit/s**, podział pasma wg adresu źródłowego/docelowego [intserv]. Max. liczba kolejek: ~690 IN, ~690 OUT.

obciążenie jest akceptowalne - w godzinach szczytu load ~1.5-2.

router brzegowy

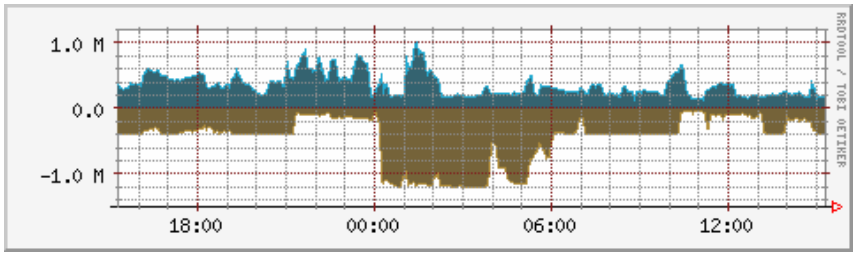
P4 3GHz, 1024MB RAM, Linux 2.6.8:

- routing [~50 tras, OSPF]
- obsługa poczty, hub DC++, hosting [%^&*(# !]
- QoS dla łącze **31Mbit/s**, Podział przepływności wg typu usługi [diffserv]. 4 IN, 4 kolejki OUT:

brak znaczącego obciążenia procesora. Load ~0.5.

praktyki „operatorów”

+ zmiana parametrów QoS w zależności od pory dnia



– dyskryminacja ruchu z sport > 1024
– dysrkyminacja ruchu na portach nieznanych [biała lista : 22, 25, 53, 80, 110, 143, 993 995, 3389, icmp, udp]
– dyskryminacja ruchu z określonych AS’ów [TP]

podsumowanie

QoS jest potrzebny gdy:

- wymagania przewyższają dostępne zasoby
- łączymy ze sobą sieci o różnych przepływnościach

Linux pozwala na realizację QoS na małą i średnią skalę dzięki mechanizmom dostępnym w standardowych kernelach.

źródła informacji

Linux Advanced Routing And Traffic Control
http://lartc.org/

strona domowa HTB
http://luxik.cdi.cz/~devik/qos/htb/

Guide to IP Layer Network Administration with Linux
http://linux-ip.net/

HTB - Strażnik Trafficu
http://linio.boo.pl/htb.html

Sterowanie przepływem danych w Linuksie
http://echelon.pl/pubs/NET4_tc.pdf

Shaping bandwidth with Linux
http://lifc.univ-fcomte.fr/~fuin

http://gogle.com

materiały do laborki

[dla studiów dziennych]

- http://pqd.one.pl/qos/
- http://pqd.anv.pl/qos/