

QoS — jakość usług w sieciach IP

Bartłomiej Świercz

Katedra Mikroelektroniki i Technik Informatycznych

Łódź, 27 maja 2008

Wykład został przygotowany w oparciu o materiały:

„Sterowanie przepływem danych w Linuxie” , Paweł Krawczyk,
kravietz@echelon.pl

„HTB strażnik trafficu” , linio@terramail.pl

QoS (ang. Quality of Service) — wymagania nałożone na połączenie komunikacyjne realizowane przez daną sieć telekomunikacyjną.

Aby zapewnić QoS, stosowane są następujące mechanizmy:

- kształtowanie i ograniczanie przepustowości
- zapewnienie sprawiedliwego dostępu do zasobów
- nadawanie odpowiednich priorytetów poszczególnym pakietom wędrującym przez sieć
- zarządzanie opóźnieniami w przesyłaniu danych
- zarządzanie buforowaniem nadmiarowych pakietów: DRR, WFQ, WRR
- określenie charakterystyki gubienia pakietów
- unikanie przeciążeń: Connection Admission Control (CAC), Usage Parameter Control (UPC)

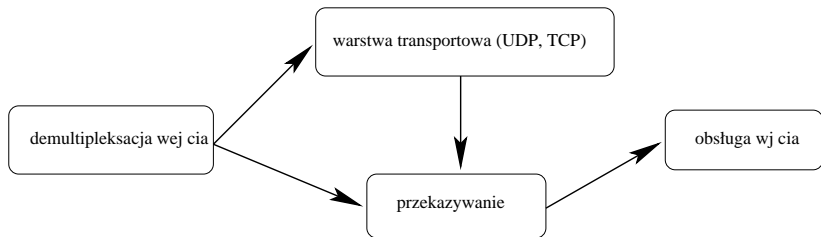
- Przepustowość (Bandwidth).
- Opóźnienie stałe i zmienne (Delays).
- Fluktuacje i wahania opóźnień (Jitters)
- Odrzucanie pakietów (Drops).

Przepustowość liczymy jako minimalne pasmo danego łącza na całym torze komunikacyjnym podzielonym przez ilość sesji współdzielących dane łącze.

- Stałe:
 - opóźnienie szeregowania,
 - opóźnienie propagacji,
 - opóźnienie związane z kodowaniem,
 - opóźnienie pakietowania (głównie RTP),
 - opóźnienie wprowadzane przez bufor zapobiegające fluktuacjom opóźnień.
- Zmienne:
 - opóźnienie kolejkowania,
 - opóźnienie przetwarzania i przekazywania
 - opóźnienia związane z kształtowaniem ruchu

- Najlepsze z możliwych dostarczenie danych.
- Model usług zróżnicowanych.
- Model usług zintegrowanych.

Router — droga przepływu danych

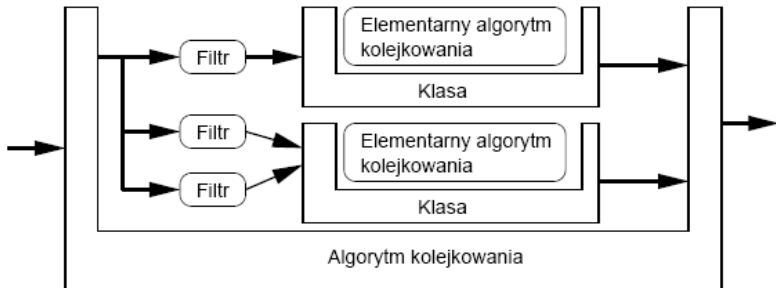


W nomenklaturze angielskiej określany jako *queueing discipline* termin ten ma trochę szersze znaczenie niż wynika to z polskiego tłumaczenia. Ogólnie rzecz biorąc, algorytm kolejkowania decyduje w jakiej kolejności przeznaczane są do wysłania pakiety znajdujące się aktualnie w kolejce. Istotne jest, że może on być albo jednym z elementarnych algorytmów opisanych poniżej, albo stanowić złożoną strukturę, podzieloną na klasy z przyporządkowanymi wieloma algorytmami elementarnymi.

Prosta kolejka z rysunku to zazwyczaj FIFO. Taki algorytm jest używany domyślnie na interfejsach Ethernet, PPP i innych, obsługiwanych przez Linuksa. Równie dobrze może to być jednak jeden z algorytmów opisanych dalej: TBF, SFQ itp.

Kolejka złożona

Kolejka złożona może być bardzo rozbudowaną strukturą, zawierającą w sobie kolejki proste oraz inne elementy, takie jak filtry i klasy.



Filtry , odpowiadające za przyporządkowanie pakietów do odpowiednich klas na podstawie wybranych parametrów, takich jak adres źródłowy, docelowy, protokołów i wiele innych.

Klasy , posiadające różne priorytety i stanowiące właściwe rozróżnienie między różnymi rodzajami ruchu.

Elementarne algorytmy kolejowania , decydujące o sposobie obsługi pakietów, które trafiły do danej klasy.

Pakiet wchodzący do złożonej kolejki jest klasyfikowany do określonej klasy przez filtr, kierujący się wybranymi parametrami pakietu. W zależności od rodzaju filtra mogą to być: adres źródłowy i docelowy pakietu, port, protokół, ToS itp. W chwili obecnej w linuxowej implementacji QoS dostępne są trzy podstawowe filtry: *route*, *fw* oraz *u32*.

Filtr oparty o tablice routingu. Każda trasa w tablicy routingu może mieć przypisane oznaczenie kolejki, do której mają być kierowane pakiety kierowane według tej trasy. Filtr route pozwala na klasyfikację pakietów ze względu na te same parametry, które są używane podczas wybierania dla niego trasy w tablicy routingu, a więc adresu docelowego (lub źródłowego, przy zastosowaniu routingu rozszerzonego). Jego największą zaletą jest szybkość oraz mały narzut czasowy podczas klasyfikacji, wynikającej z dużej efektywności operacji na tablicy routingu.

Filtr firewall opiera się o zaznaczanie pakietów przez firewall wbudowany w jądro. W przypadku ipchains do konfiguracji regułki filtra należy dodać opcję `-m`, której parametrem jest liczba stanowiąca oznaczenie pakietu. Jeśli przetwarzany przez firewall pakiet pasuje do danej regułki, to zostaje on oznaczony podaną liczbą. Faktycznie oznaczenie polega na ustawieniu pola `skb->priority` w pakiecie na podaną wartość. Warto zaznaczyć, że pole to jest także ustawiane przez jądro w zależności od ToS pakietu.

Istotna jest interpretacja oznaczenia. Jest ono liczbą 32-bitową, której starsze 16 bitów określa kolejność do której ma być skierowany pakiet, a młodsze klasę w obrębie tej kolejki. Przykładowo, chcąc skierować pakiet do klasy 1:3 powinniśmy użyć parametru `-m 0x10003`. Filtr `fw` pozwala na klasyfikację pakietów według wszystkich parametrów rozpoznawanych przez firewall. W odróżnieniu od filtra `route` są to więc także informacje z protokołów wyższych niż TCP i UDP: numery portów, typy pakietów ICMP itp.

Najbardziej złożony z filtrów dostępnych w Linuksie. Jest w całości oparty o tablice haszujące, co zapewnia wydajność nawet przy bardzo złożonych zbiorach reguł. Posiada największe możliwości jeśli chodzi o wybór kryteriów, które muszą spełniać klasyfikowane pakiety.

Elementarne algorytmy kolejowania w jądrze Linux

```
[lap:~] cd /usr/src/linux/net/sched
```

```
[lap:sched] ls sch*
```

```
sch_api.c          sch_cbq.c          sch_generic.c
sch_atm.c          sch_dsmark.c       sch_generic.o
sch_blackhole.c   sch_fifo.c         sch_gred.c
sch_hfsc.c         sch_netem.c        sch_sfq.c
sch_htb.c          sch_prio.c         sch_tbf.c
sch_ingress.c     sch_red.c          sch_teql.c
```

Konfiguracja jądra Linux

Linux Kernel v2.6.16.18 Configuration

Linux Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < >

```
Code maturity level options --->
General setup --->
Loadable module support --->
Block layer --->
Processor type and features --->
Power management options (ACPI, APM) --->
Bus options (PCI, PCMCIA, EISA, MCA, ISA) --->
Executable file formats --->
Networking --->
Device Drivers --->
```

v(+)

<Select> < Exit > < Help >

Linux Kernel v2.6.16.18 Configuration

Networking

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < >

```
--- Networking support
[*] Networking options --->
[ ] Amateur Radio support --->
<M> IrDA (infrared) subsystem support --->
<M> Bluetooth subsystem support --->
<M> Generic IEEE 802.11 Networking Stack
[ ] Enable full debugging output
--- IEEE 802.11 WEP encryption (802.1x)
<M> IEEE 802.11i CCMP support
<M> IEEE 802.11i TKIP encryption
```

<Select> < Exit > < Help >

Linux Kernel v2.6.16.18 Configuration

Networking options

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < >

^(-)

- < > **A**NSI/IEEE 802.2 LLC type 2 Support
- < > **T**he IPX protocol
- < > **A**ppletalk protocol support
- < > **C**CITT X.25 Packet Layer (EXPERIMENTAL)
- < > **L**APB Data Link Driver (EXPERIMENTAL)
- [] **F**rame Diverter (EXPERIMENTAL)
- < > **A**corn Econet/AUN protocols (EXPERIMENTAL)
- < > **M**AN router
- QoS and/or fair queueing** --->
- Network testing --->

< **Select** >

< **Exit** >

< **Help** >

Linux Kernel v2.6.16.18 Configuration

QoS and/or fair queueing

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [] excluded <M> module < >

[*] QoS and/or fair queueing

- Packet scheduler clock source (gettimeofday) --->
- Queueing/Scheduling
- < > Class Based Queueing (CBQ) (NEW)
- < > Hierarchical Token Bucket (HTB) (NEW)
- < > Hierarchical Fair Service Curve (HFSC) (NEW)
- < > Multi Band Priority Queueing (PRIO) (NEW)
- < > Random Early Detection (RED) (NEW)
- < > Stochastic Fairness Queueing (SFQ) (NEW)
- < > True Link Equalizer (TEQL) (NEW)

v(+)

<Select>

< Exit >

< Help >

Kolejka FIFO (First In, First Out, nazywana także *drop-tail*)
Najczęściej stosowana, nie tylko zresztą w ruterach. Prosta kolejka pakietów, przesuujących się do wyjścia. Limitowana wyłącznie przez wydajność interfejsu wyjściowego. Jedynym jej parametrem jest wielkość, mierzona w bajtach dla bfifo (byte FIFO) lub pakietach dla pfifo (packet FIFO).

Algorytm TBF (Token Bucket Filter) to prosta kolejka wypuszczająca wyłącznie pakiety poniżej ustalonego administracyjnego natężenia przepływu, z możliwością buforowania chwilowych przeciążeń. Implementacja TBF posiada bufor (kubetek, ang. bucket), do którego wpadają żetony z natężeniem (ang. token rate) określonym parametrem *rate*. Rozmiar kubetka (ilość żetonów, które może pomieścić) jest określany parametrem *buffer*.

Jeżeli dane wchodzi do TBF z natężeniem przepływu równym natężeniu przepływu żetonów to wówczas każda porcja danych ma swój odpowiadający żeton i przechodzi przez filtr.

Jeżeli dane wchodzi do TBF z natężeniem przepływu mniejszym niż żetony, ponieważ tylko część żetonów jest zabierana przez wychodzące dane, to kubetek wypełnia się niewykorzystanymi żetonami (ale nie może ich być więcej niż wielkość *buffer*). Mogą zostać one wykorzystane w przyszłości, w przypadku chwilowego przeciążenia (ang. burst).

Jeśli natężenie danych jest większe niż ustalone natężenie żetonów, to mamy do czynienia z przeciążeniem filtra. W tej sytuacji dane mogą być wysyłane dopóki nie zostaną zużyte wszystkie żetony, które mogły się tam nagromadzić w okresie poprzedzającym przeciążenie. Jeśli w kubku nie ma już żetonów, pakiety są kasowane.

Jak wynika z wcześniejszego opisu, parametr *buffer* określa, jak długo TBF będzie w stanie buforować dane przekraczające parametr *rate* w wypadku przeciążenia. Drugi istotny parametr to limit, określający wielkość kolejki w bajtach. Algorytm TBF znajduje powszechne zastosowanie do ograniczania pasma oraz regulowania natężenia przepływu danych w protokołach typu RSVP i innych, służących do gwarantowania przydziału zasobów sieciowych (patrz RFC 2215).

PRIO (Simple Priority Queueing) to prosta kolejka umożliwiająca preferowanie określonych pakietów. Składa się z kilku kolejek, z których zawsze najpierw obsługiwane są te o wyższym priorytecie. W razie przekroczenia limitu wielkości kolejek, pierwsze kasowane są pakiety z kolejki o najniższym priorytecie. Ten rodzaj kolejkowania ma zastosowanie w sytuacji, kiedy określony rodzaj ruchu musi posiadać absolutne pierwszeństwo i nie nadaje się do optymalizacji wykorzystania łącza.

SFQ (Stochastic Fairness Queueing) to prosta i posiadająca niewielkie wymagania obliczeniowe odmiana algorytmu sprawiedliwego kolejkowania (ang. fair queueing). Kolejka jest w tym algorytmie rozpatrywana jako składająca się z ciągów pakietów zwanych konwersacjami (conversations) lub strumieniami (flows). Do jednej konwersacji należą pakiety posiadające takie same adresy źródłowe i docelowe oraz protokół w nagłówku IP. Przykładowo, do samodzielnych konwersacji należeć będą połączenie TCP, ciągły strumień danych UDP (NFS, Quake) między dwoma hostami itp. Każda z konwersacji jest obsługiwana sprawiedliwie i cyklicznie (round-robin), to znaczy w każdym przebiegu wysyłane jest po jednym pakiecie z każdego ciągu. Konsekwencją jest również to, że najszybciej obsługiwane są konwersacje krótkie, czyli stanowiące małe obciążenie dla sieci. W przypadku przepełnienia natomiast gubione są pakiety z końca kolejki (stad tail-drop).

Parametrami SFQ są: interwał między przeliczanie funkcji haszującej (perturb) oraz jednostka kolejkowania (quantum). Ta ostatnia wartość powinna być większa lub równa MTU obowiązującemu na danym interfejsie. Regularne przeliczanie funkcji haszującej jest konieczne ze względu na jej prostotę, która może powodować kolizje czyli identyczne wyniki dla różnych konwersacji. Parametr ten podaje się w sekundach i powinien od być uzależniony od natężenia oraz charakterystyki ruchu — im więcej podobnych konwersacji przechodzi przez łącze, tym częściej powinna być przeliczana funkcja haszująca. W praktyce stosuje się czasy w przedziale 5 – 20 sekund.

RED (Random Early Detection) to algorytm mający na celu unikanie przeciążeń przez wykorzystanie mechanizmów istniejących już w protokole TCP. RED przewiduje wystąpienie przeciążenia łącza i gubi pakiety, sygnalizując tym samym nadawcy, że powinien ograniczyć transmisję. W przypadku protokołu TCP jest to założenie jak najbardziej poprawne, a po zakończeniu przeciążenia TCP potrafi automatycznie powrócić do optymalnej prędkości wysyłania pakietów. W związku z tym RED znajduje zastosowanie jako algorytm w podklasach CBQ, przeznaczonych dla ruchu TCP.

Charakterystyczna cecha RED jest to, że w odróżnieniu od pozostałych algorytmów, próbuje on zapobiegać przeciążeniu zanim ono wystąpi, a nie tylko zminimalizować jego skutki w czasie jego trwania. Pierwszym krokiem po przyjęciu nowego pakietu przez RED jest obliczenie średniego rozmiaru kolejki (w bajtach) będącego funkcją poprzedniej średniej oraz obecnego rozmiaru. Na tej podstawie — oraz dwóch administracyjnie ustalonych parametrów min i max — RED oblicza prawdopodobieństwo z jakim pakiet powinien zostać odrzucony. Prawdopodobieństwo to rośnie wraz ze wzrostem nasycenia łącza, aż do porzucenia pakietu. Algorytm RED jest opisany m. in. w RFC 2309.

CBQ (Class Based Queueing) jest algorytmem, stanowiącym podstawę do podziału przepustowości łącza (link sharing) oraz szkielet pozwalający na wykorzystanie wszystkich wyżej opisanych algorytmów elementarnych.

W CBQ ruch jest dzielony na kilka kolejek, zwanych dalej klasami, charakteryzujących się priorytetem oraz przydzieloną przepustowością. Siła CBQ jest to, że pakiety mogą być rozdzielane do kolejek–klas na podstawie dowolnych kryteriów za pomocą opisanych wcześniej filtrów. Natomiast zamiast domyślnie ustawianych kolejek FIFO, składowymi CBQ mogą być dowolne z opisanych wcześniej algorytmów elementarnych.

- Podział sumarycznej przepustowości łącza na kilka części (klas), przydzielonych według potrzeb określonym rodzajom usług, adresom IP itp.
- Przesyłanie pakietów z różnym pierwszeństwem w zależności od tych samych parametrów.
- Przydzielenie odpowiednim rodzajom ruchu właściwych algorytmów kolejkowania, np. dla ruchu masowego algorytmu SFQ, dla ruchu TCP algorytmu RED itp.

Klasy CBQ charakteryzuje kilka podstawowych parametrów, które są poniżej opisane w składni stosowanej przez polecenie tc:

- parent** — identyfikator kolejki macierzystej, do której dana klasa przynależy. Podawany w postaci KOLEJKA:0.
- classid** — identyfikator danej klasy podawany w postaci KOLEJKA:KLASA, gdzie KOLEJKA jest numerem kolejki do której dana klasa przynależy, a KLASA numerem deklarowanej klasy. Ponieważ kolejka macierzysta jest i tak podawana w parametrze *parent*, wystarczy podać sam numer klasy, np. classid :3.

prio — priorytet danej klasy, podawany jako liczba całkowita z przedziału 1–10. Im mniejsza wartość, tym większe pierwszeństwo będzie miała dana klasa. Priorytet pozwala podzielić klasy według wzrastającego pierwszeństwa w obsłudze danych przekazywanych przez daną klasę. Przykładowo, klasom przenoszącym ruch interaktywny (TELNET, SSH) można ustawić większy priorytet niż klasom z ruchem masowym (FTP). Oczywiście, liczbowe wartości priorytetów dla klas interaktywnych powinny być mniejsze niż dla klas masowych.

bandwidth — sumaryczna przepustowość interfejsu, w którym konfigurowane są klasy. Podawana jako liczba z przyrostkiem określającym jednostkę, np. dla Ethernetu 10Base-T 10mbit. Inne dopuszczalne jednostki to bps, kbps (odpowiednio bajty oraz kilobajty na sekundę, kbit oraz mbit (kilobity i megabity na sekundę)).

- rate** — przepustowość pasma przydzielonego danej klasie, stanowiąca ułamek przepustowości interfejsu. Jednostki są identyczne jak w parametrze bandwidth.
- weight** — względna waga danej klasy, która dla wszystkich klas powinna być wartością proporcjonalną do rate. W praktyce można przyjmować wartości dziesięciokrotnie mniejsze niż rate, ale parametr ten nie jest niezbędny i można zawsze używać wartości 1.

- avpkt** — średnia wielkość pakietów przesyłanych w tej klasie, w bajtach. Wielkość tę można wyznaczyć eksperymentalnie, ale przeważnie będzie ona stanowić 50-60% MTU danego interfejsu.
- mput** — minimalna wielkość pakietów, które będą przesyłane przez daną klasę, w bajtach. Mniejsze pakiety nie będą podlegały klasyfikacji.

- allot** — suma MTU oraz nagłówka warstwy łącza danego interfejsu w bajtach. Przykładowo, dla zwykłego Ethernetu (10Base-T) wartość ta wynosi 1514 (1500 + 14 nagłówka Ethernet), a dla połączenia PPP z MTU 576: 580 (576 + 4 bajty nagłówka PPP).
- maxburst** — parametr określający dopuszczalne chwilowe przeciążenie danej klasy (burstiness). Parametr ten jest ściśle związany z samym algorytmem CBQ.
- est** — dwa parametry miernika natężenia ruchu (rate estimator) pracującego w danej klasie, podawane w postaci $est\ X\ Y$. Miernik określa średnie natężenie przepływu danych w danej klasie przez okres X sekund ze stałą czasową Y sekund. W praktyce typowymi wartościami jest 1 i 8.

HTB jest stosunkowo nowym algorytmem kolejkowania. Posiada podobne możliwości (i parametry) co algorytm CBQ. Główne cechy odróżniające algorytm HTB od algorytmu CBQ to:

- jest szybszy (szybsze wysyłanie pakietów, mniejsze obciążenie serwera),
- jest bardziej dokładny,
- łatwiej zrozumieć jego składnię i sam sposób działania.

- „HTB strażnik trafficu”, `linio@terramail.pl`
- `http://www.docum.org/`