Kształtowanie ruch w sieciach Linux

1. Wprowadzenie

Wymagania wstępne: wykonanie ćwiczenia „Statyczny wybór trasy w systemie Linux”.
Potrzeba sterowania ruchem w sieciach komputerowych wynika głównie z faktu, że obecnie zapotrzebowanie na usługi sieciowe rośnie w szybszym tempie, niż aktualne możliwości sieci. Sterowanie takie można podzielić na dwie kategorie:

- **kształtowanie ruchu** (ang. *traffic shaping*) – zgłaszane do wysłania pakiety mogą zostać opóźnione lub odrzucone dla zachowania określonych właściwości ruchu (np. aktualnej przepustowości);
- **wyrównywanie obciążenia** (ang. *load balancing*) – dla zgłaszanych pakietów należy wybrać drogę w sieci uwzględniając politykę równomiernego obciążenia łączy.

Jednym z protokołów wspierających kontrolę przepływu pakietów jest protokół TCP, który pozwala transmitującym systemom dostosować się do poziomu ruchu i warunków w sieci. Jednak aby w pełni kształtować ruch w sieci konieczne są dodatkowe mechanizmy kontrolne; wykorzystuje się zatem w tym celu głównie tzw. algorytmy kolejkowania pakietów (ang. *queueing discipline*) – często nazywane w skrócie kolejkami. Kolejki obsługiwane przez interfejsy sieciowe można podzielić na dwie grupy:

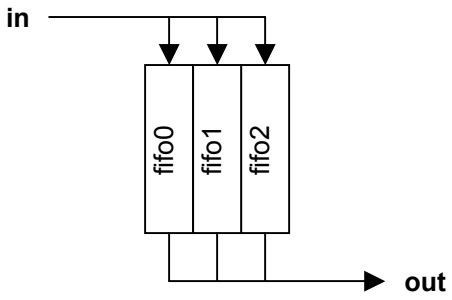
- **kolejki proste** – takie jak np. FIFO, które określają w jako sposób pakiety są traktowane przez system, bez możliwości zagnieżdżania innych kolejek;
- **kolejki złożone** – które mogą być rozbudowaną strukturą, zawierającą inne kolejki oraz filtry i klasy.

Oprócz algorytmów kolejkowania kształtowanie ruchu można także osiągnąć stosując w sieci odpowiednie mechanizmy zapewniania jakości (ang. *quality of service*). Przykładami takich rozwiązań są: **IntServ** (ang. *integrated services*) oraz **DiffServ** (ang. *differentaited services*). Pierwsze z tych rozwiązań – w praktyce realizowane przez protokół RSVP (ang. *resource reservation protocol*) – polega na zestawieniu w sieci połączenia gwarantującego żądane właściwości jakościowe. Niestety wymaga to aby wszystkie węzły pośrednie w sieci (rutery) obsługiwały to rozwiązanie; dodatkowo w dużych sieciach mogą pojawić się problemy z utrzymywaniem zestawionych połączeń oraz z szybko wyczerpującymi się zasobami. Uwzględniając te problemy, drugie rozwiązanie – DiffServ – nie wymaga zestawiania *a priori* połączenia w sieci, ale opiera się na dzieleniu pakietów na klasy przy ich wysyłaniu. Wówczas każdy węzeł pośredni zobowiązany jest traktować wszystkie pakiety danej klasy z jednakowymi parametrami jakościowymi.

1.1 Podstawowe kolejki proste

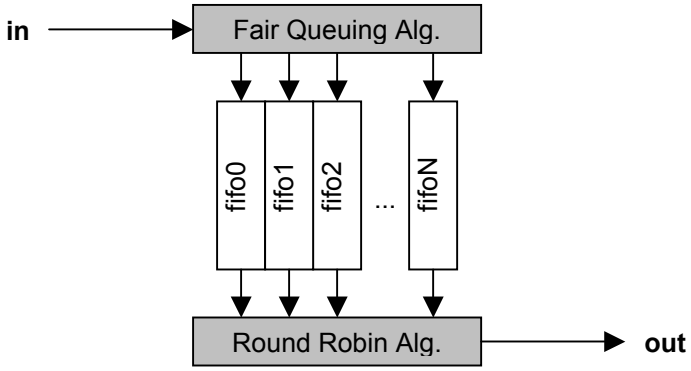
Jedną z najprostszych kolejek tego typu jest powszechnie znana kolejka FIFO, jednak oprócz niej do kształtowania ruchu wykorzystuje się także inne kolejki, z których podstawowe znaczenie mają: *pfifio_fast*, *SFQ* oraz *TBF*. Oto ich krótka charakterystyka:

- **pfifo_fast** – domyślna kolejka dla interfejsów sieciowych w systemie Linux, składająca się z trzech kolejek FIFO numerowanych od 0 (najwyższy priorytet) do 2 (najniższy priorytet); każda kolejka FIFO działa niezależnie, tj. jeśli w paśmie 0 są jeszcze pakiety, to pasmo 1 nie zostanie obsłużone; pakiety trafiają do poszczególnych kolejek FIFO na podstawie tzw. mapy priorytetów, uwzględniającej m.in. pole TOS pakietów IP; kolejka **pfifo_fast** jest przedstawiona na rysunku nr 1;



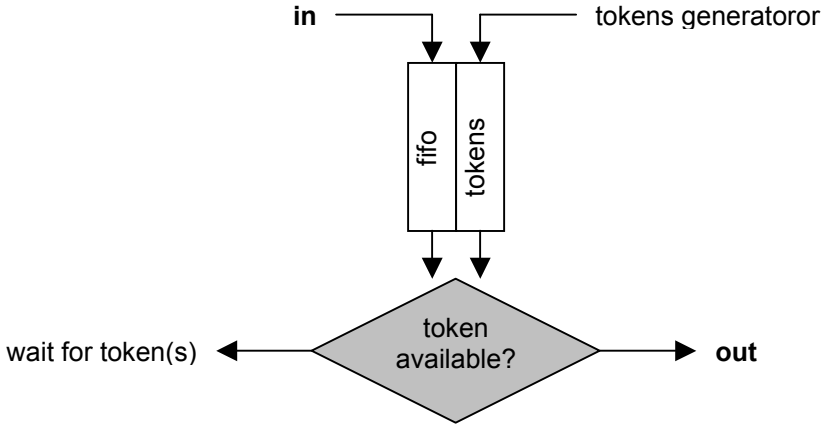
Rysunek nr 1. Kolejka **pfifo_fast**.

- **SFQ** (ang. *stochastic fair queuing*) – składa się z wielu kolejek FIFO; do każdej z tych kolejek trafiają pakiety z takim samym adresem źródłowym, docelowym (łącznie z numerami portów) i protokołem warstwy transportowej; pakiety z kolejek FIFO pobierane są zgodnie z algorytmem *round robin*; kolejka **SFQ** jest przedstawiona na rysunku nr 2.



Rysunek nr 2. Kolejka **SFQ**.

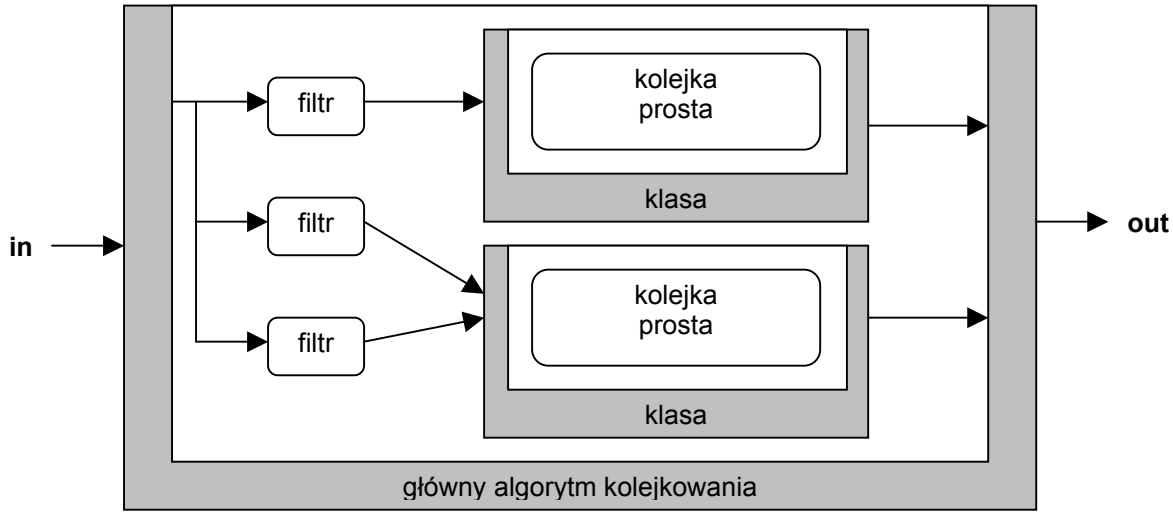
- **TBF** (ang. *token bucket filter*) – działa w oparciu o żetony (ang. *token*), które sterują ruchem pakietów w kolejce; każdy pakiet oczekuje na pojawienie się w kolejce określonej liczby żetonów, jeśli takie się pojawią pakiet może opuścić kolejkę; kolejka **TBF** jest przedstawiona na rysunku nr 3.



Rysunek nr 3. Kolejka **TBF**.

1.2 Kolejki złożone

W skład kolejki złożonej wchodzą: kolejki proste, filtry oraz klasy. Klasy są kontenerami dla innych dyscyplin kolejkowania; tworzą one strukturę drzewa, tj. każda klasa posiada jednego *rodzica* oraz jednego lub wielu *potomków*. Każda klasa posiada swój identyfikator (uchwyt), składający się z dwóch części: starszej (określający rodzica) oraz młodszej (musi być unikalny w skali *potomków* tej samej klasy) – np. **1:0**, co można zapisać także jako **1:**, jest to identyfikator klasy korzenia. Podstawową metodą przydzielania ruchu do konkretnych klas jest zastosowanie filtrów pakietów, które dokonują klasyfikacji na podstawie wybranych parametrów, takich jak np. adres źródłowy lub docelowy, nr portów itp. Kolejki proste w klasach decydują o sposobie obsługi pakietów, które trafiły do danej klasy. Ogólną postać kolejki prezentuje rysunek nr 4.



Rysunek nr 4. Kolejka złożona.

Kiedy jądro systemu zdecyduje się pobrać pakiet z kolejki złożonej, to pakiet ten będzie pochodził z jeden z dyscyplin kolejkowania należącej do jednej z klas. Tak więc kolejka złożona może priorytetyzować pewną część ruchu poprzez usiłowanie zdjęcia pakietów z jednej ze swoich kolejek wcześniej niż z innych (główny algorytm kolejkowania).

Przykładami kolejek złożonych są: *PRIO* (ang. *simple priority queueing*) – podobna w założenia do kolejki *pfifo_fast*; *WRR* (ang. *weighted round robin*) – działa na zasadzie rozdzielania dostępnego pasma w odpowiednich proporcjach wedle potrzeb; *CBQ* (ang. *class based queueing*) – jedna z bardziej rozbudowanych kolejek złożonych, potrafiąca klasyfikować ruch na podstawie zadeklarowanego pasma oraz umożliwiająca klasą wykorzystywanie nieużywanego pasma innych klas; *HTB* (ang. *hierarchical token bucket*) – obecnie najistotniejsza kolejka złożona w środowisku Linux, oparta jest na podobnym mechanizmie żetonów co kolejka *TBF*. Oto podstawowe możliwości kolejki *HTB*:

- dzielenie łącza na klasy o żądanej gwarantowanej przepustowości z możliwością wykorzystywania nieużywanego pasma innych klas;
- funkcja definiowania priorytetów klas;
- dzielenie hierarchiczne – każda klasa może być podzielona na wiele podklas.

1.3 Narzędzie `tc`

Program **`tc`** (wchodzący w skład pakietu **`iproute2`**), jest narzędziem pozwalającym realizować w systemie Linux kolejki oraz filtry dla kolejek złożonych. Ogólna postać polecenia **`tc`** jest następująca:

- dla kolejek:

`tc qdisc [add | change | replace | link] dev DEV [parent qdisc-id | root] [handle qdisc-id] qdisc [qdisc specific parameters]`

- dla klas:

`tc class [add | change | replace] dev DEV parent qdisc-id [classid class-id] qdisc [qdisc specific parameters]`

- dla filtrów:

`tc filter [add | change | replace] dev DEV [parent qdisc-id | root] protocol protocol prio priority filter-type [filtertype specific parameters] flowid flow-id`

Należy zaznaczyć, że zakładane kolejki przy pomocy omawianego narzędzia zawsze dotyczą interfejsu wyjściowego**!** Podstawowe parametry dla kolejki *HTB* to:

- **`rate`** – gwarantowana minimalna przepustowość dla klasy;
- **`celi`** – maksymalna przepustowość dla klasy.

Do filtrowania pakietów służy filtr **`u32`**, oto jego podstawowe selektory:

- **`mach ip src`** – źródłowy adres IP;
- **`mach ip dst`** – docelowy adres IP;
- **`match [udp | tcp] src`** – port źródłowy;
- **`match [udp | tcp] dst`** – port docelowy.

Przykłady poleceń `tc`:

- **`tc [qdisc | filter | class]`** – wyświetlenie informacji o kolejkach/filtrach/klasach;
- **`tc qdisc del root dev eth0`** – usunięcie dyscypliny kolejkowania z interfejsu eth0;
- **`tc qdisc add dev eth0 root handle 1:0 htb`** – dodanie kolejki *HTB* do interfejsu eth0;
- **`tc class add dev eth0 parent 1:1 classid 1:2 htb rate 5mbit celi 10mbit`** – dodanie nowej klasy z kolejką *HTB* oraz minimalną przepustowością 5mbit I maksymalna przepustowością 10mbit
- **`tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 10.0.0.1 flowid 1:2`** – dodanie nowego filtra **`u32`** dla pakietów adresowanych pod wskazany adres IP.

2. Organizacja, wymagany sprzęt, oprogramowanie

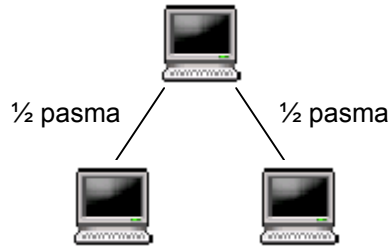
- Zadanie wykonuje grupa 3-osobowa;
- sprzęt: 3 komputery PC;
- oprogramowanie: system Linux z pakietem **`iproute2`**.

3. Zadania

1. Należy skonfigurować podział pasma zgodnie z rysunkiem nr 5 wykorzystując narzędzie **`tc`** oraz filtr **`u32`** (minimalnie ½ pasma, maksymalnie całe pasmo).
2. Dla powyższej konfiguracji wprowadzić do klas *potomnych* kolejkę *SFQ*.

3. Skonfigurować podział pasma na komputerze na dwie klasy dla dwóch usług: www oraz ftp.

Poprawność wszystkich konfiguracji zweryfikować programem **`netperf.`**



Rysunek nr 5. Konfiguracja do zadania.

4. Pytania sprawdzające

1. Dlaczego kolejkowanie dotyczy ruchu wychodzącego?
2. Czy możliwe jest kolejkowanie ruchu wchodzącego do interfejsu sieciowego?
3. Wyjaśnij zasady zarządzania pasmem w przełącznikach sieci Ethernet zgodne ze standardem IEEE 802.1p.

5. Literatura

1. HTB Linux queuing discipline manual – user guide
   http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm
2. „Kształtowanie ruchu i zaawansowany routing” B.Hubert (tłum. Ł.Bromski)
   http://lukasz.bromirski.net/docs/translations/lartc-pl.html
3. Podręcznik systemowy **`man`** dla polecenia **`tc`**, dla filtrów **`tc-filters`** oraz dla kolejki HTB **`tc-htb`**.