Problem kolejkowania dostępu czyli zarządzanie przepustowością sieci

Piotr Misiuda 4FD L08

Wstęp

Chciałbym aby mój projekt nie był czystym teoretycznym rozważaniem na temat jak to działa , a po co , a dlaczego. Postaram się zawrzeć w nim wiedze praktyczną oraz podstawowe wiadomości teoretyczne tak aby każdy debianowy średniak mógł dzięki niemu zastosować omówione rozwiązania na serverze. W tym sensie jest to bardziej instrukcja obsługi plus podstawowe wiadomości teoretyczne.

Kolejkowanie dostępu ma sens gdy nasz server (system Debian oraz SDI) dierżawi dostęp do internetu innym komputerom. Jak wiadomo ludzie kożystają z sieci , jedni intensywniej inni mniej.Prawdziwą zmorą administratorów są programy typu P2P (z angielskiego peer to peer , punkt to punkt) czyli Kazaa , Direct Connect. Powodują one przeciążanie całego łącza i pogorszenie parametrów sieciowych całej sieci. Innymi słowy jedna osoba ściąga filmy kazaą a reszta cierpi z powodu złego działania sieci.
W tym wypadku trzeba pomyśleć CBQ czyli pasowaniu łącza bądź inaczej kolejkowaniu dostępu.

Poniżej przedstawiam łatwy sposób przydzielania użytkownikom pasm o równej przepustowości. Jeżeli jest wolne pasmo (ktoś nie korzysta z sieci) inny pobiera z jego przepustowości. Gdy użytkownik zacznie ponownie korzystać z sieci pasmo jest odbierane osobie która je "pożyczyła".

Algorytm sotosowany był dla servera posiadającego SDI jako punkt dostępowy do sieci. Łatwo go jednak przerobić na inne typy łącz.

W wypadku SDI wygląda to tak : mamy łącze 112,5Kbit. Czyli każdemu użytkownikowi (sześć osób) ma przypadać 18,75Kbit jeżeli w danej chwili wszyscy korzystają z internetu(112,5 / 6 = 18.75). To jest prędkość teoretyczna, bo wiadomo że taka raczej rzadko się zdarza. Jeżeli jeden z nich idzie wyłącza komputer , a drugi potrzebuje więcej, więc pobiera z tej niewykorzystanej kolejki.

Praktyka

Aby działało nasz server musi posiadać pakiet iproute , jeżeli go nie mamy wykonujemy polecenie apt-get install iproute.
Po zainstalowaniu wykujemy komendę tc qdisc , jeżeli w odpowiedzi otrzymamy komunikaty błędów oznacza to , że musimy przekompilować jądro.

Należ dodać do jądra w opcjach : Networking Options -> QoS and/or fair queuing -> wszystkie moduły.

Tworzenie kolejki Root

Kolejki są po to w nich oczekiwać na dostęp do internetu .Czyli tworzymy je, aby obsługiwały nasz ruch wychodzący z sieci do internetu. Zasada jest taka że im wolniej pakiet wyślemy tym wolniej do nas pakiet przyjdzie.

jak mawia PCkurier :

"..Zastosowanie odpowiednich algorytmów pozwala buforować przechodzący przez router ruch. Nadchodząca z szybkiego łącza transmisja może być - przed "wtłoczeniem" jej w łącze o mniejszej przepustowości - spowalniana z pewnymi priorytetami, odrzucanie pakietów wykraczających poza limit docelowego łącza również może odbywać się w sposób zbytnio nie krzywdzący wybranych użytkowników. Możliwe jest zarządzanie tylko ruchem wychodzącym, choć niektóre operacje - np. odrzucanie pakietów w warunkach przeciążenia sieci - odbywają się po stronie wejścia..."

Krótko mówiąc CBQ dba o to, by łącze było przez odpowiedni czas bezczynne i dzięki temu prawdziwa przepustowość spadała do skonfigurowanej wielkości. Żeby to robić kalkuluje głównie czas, który powinien upłynąć pomiędzy pakietami.

Musimy stworzyć kolejkę główną naszego ruchu. Będzie to kolejka, do której będą dołączane inne, stworzone już dla konkretnego użytkownika Kolejkę te można nazwać kolejka root

Wydajemy polecenie które tworzy nam główną kolejkę:

tc qdisc add dev eth0 root handle 1:0 cbq bandwidth 100Mbit rate 112.5Kbit perturb 10 avpkt 1000

Gdzie:

- "***tc qdisc add dev eth0 root handle 1:0***" tworzy kolejke główna
- "***cbq***" - oznacza jaki algorytm kolejkowania będzie stosowany przez kernel
- "***bandwidth***" - szybkość interfejsu czyli tutaj karta - 100Mbitowa
- "***rate***" - wielkość do jakiej zmniejszamy szybkość interfejsu
- "**perturb**" - okresowa rekonfiguracja funkcji mieszającej , zalecana wartość 10 sekund
- "***avpkt***" - średnia wielkość pakietu (dla SDI przyjmujemy 1000)

Sprawdzamy czy kolejka działa wywołując polecenie:

tc -s qdisc ls

W wyniku powinniśmy otrzymać:

qdisc cbq 1: dev eth0 rate 14400bps (bounded,isolated) prio no-transmit

Sent 60 bytes 4 pkts (dropped 0, overlimits 0) borrowed 0 overactions 0 avgidle 56888 undertime 0

Oznacza to że kolejka została założona. Kolejny etap to tworzenie filtrów.

Tworzenie filtra - root

tc filter add dev eth0 parent 1:0 prio 10 protocol ip u32 divisor 16

Polecenie to zakłada główny filtr bez niego nie można zakładać filtrów do kolejek dzieci.

Tworzenie klas - podział pasma

Przykład pokazuje dwa komputery i przyznany im limit 18Kbit , oparty jest tylko na przydzieleniu użytkownikowi odpowiedniego pasma.

klasa przykładowy na eth0 192.168.1.8

Poniższa regułka zakłada klasę, która ma ograniczenie pasma 18Kbit, posiada priorytet 5, udostępnia swoje niewykorzystane pasmo innym potrzebującym oraz sama nie pożycza swojego pasma.

tc class add dev eth0 parent 1:0 classid 1:2 cbq bandwidth 100Mbit rate 18Kbit weight 1 prio 5 allot 1514 maxburst 20 avpkt 1000 bounded sharing

To polecenie przydziela klasie algorytm SFQ, czyli "sprawiedliwy podział pasma"
Dla ruchu HTTP i FTP (czyli masowego) zalecany jest algorytm RED

tc qdisc add dev eth0 parent 1:2 sfq perturb 10

Poniżej troszkę materiału o SFQ zaczerpniętego z PCkurier'a
Stochastic Fairness Queuing (SFQ)
Protokół "sprawiedliwego" podziału pasma, który dzieli pakiety na kolejki obsługiwane algorytmem round-robin (tzn. w określonej kolejności, w tych samych porcjach, w koło), przy czym pakiety są rozdzielane do różnych kolejek według tzw. konwersacji, czyli związane z tą samą transmisją, o identycznym adresie źródłowym, docelowym i protokole w warstwie powyżej IP. Przepełnienie kolejek powoduje utratę pakietów nadchodzących do kolejek. Round-robin daje szanse mniejszym transmisjom. SFQ działa efektywnie tylko tam, gdzie istnieje przeciążenie transmisją - bez tego kolejki są opróżnione i nie jest możliwe obsługiwanie ich po kolei. Ponieważ kolejek jest mniej niż konwersacji, rozbicie konwersacji na kolejki odbywa się pseudolosowo na podstawie algorytmu mieszającego. SFQ wymaga interwencji administratora w zakresie dwóch parametrów pracy:
- "**perturb**" - okresowa rekonfiguracja funkcji mieszającej dla zapewnienia losowego rozmieszczenia konwersacji w kolejkach; zalecana wartość 10 sekund,
- "**quantum**" - minimalna porcja bajtów opuszczająca kolejkę przed obsługą następnej kolejki; nie może być mniejsza od MTU danego interfejsu.

I część prawie najważniejsza, bez której nie będzie tego podziału. Bo to właśnie filtr decyduje co zrobić z pakietami, które w tym przypadku przychodzą do komputera, a wychodzą z serwera.

tc filter add dev eth0 parent 1:0 prio 10 u32 match ip dst 192.168.1.8 flowid 1:2

tc filter add dev eth0 parent 1:0 prio 10 u32 match ip src 192.168.1.1 flowid 1:2

przykładowa klasa na eth0 192.168.1.10

Druga klasa, która różni się tym od poprzedniej że, sama nie udostępnia swojego pasma(isolated), ale pożycza od innych(borrow).

tc class add dev eth0 parent 1:0 classid 1:3 cbq bandwidth 100Mbit rate 18Kbit weight 1 prio 5 allot 1514 maxburst 20 avpkt 1000 borrow isolated

tc qdisc add dev eth0 parent 1:3 sfq perturb 10

tc filter add dev eth0 parent 1:0 prio 10 u32 match ip dst 192.168.1.10 flowid 1:3

tc filter add dev eth0 parent 1:0 prio 10 u32 match ip src 192.168.1.1 flowid 1:3

Czy to działa ?

Sprawdzenie całości. Wpisujemy

tc -s qdisc ls

1. qdisc sfq 80c8: dev eth0 quantum 1514b perturb 10sec

Sent 818096 bytes 1172 pkts (dropped 0, overlimits 0) backlog 13p

2. qdisc sfq 80c4: dev eth0 quantum 1514b perturb 10sec

Sent 908233 bytes 923 pkts (dropped 0, overlimits 0) backlog 4p

3. qdisc cbq 1: dev eth0 rate 14400bps (bounded,isolated) prio no-transmit

Sent 1732622 bytes 2133 pkts (dropped 0, overlimits 2911) backlog 17p borrowed 0 overactions 0 avgidle 56888 undertime 0

Jak widać w każdej kolejce jest pokazana ilość danych przesłanych(czyli ruch jest kolejkowany wg. naszych ustawień), a w podsumowaniu również ilość pakietów przekraczających narzucony limit - overlimits.

Na koniec

CBQ jest narzędziem trudnym nieprzyjaznym i skomplikowanym , daje jednak możliwość przekształcenia zwykłego komputera klasy PC w router sieciowy za który musielibyśmy zapłacić dosyć drogo. Pozwala kierować i zarządzać ruchem w sieci dzięki czemu staje się narzędziem dla administratorów chcących stosować profesjonalne rozwiązania na nieprofesjonalnym sprzęcie.

Część materiałów (niewielka) została zaczerpnięta z PCkurier'a ,reszta stanowi doświadczenia zdobyte przez autora oraz znajome osoby zajmujące się podobnymi zagadnieniami