linio@terramail.pl

HTB – strażnik trafficu

wersja 0.7

1. Ale o co chodzi ?

Angielskie „traffic control” przetłumaczyć można jako kontrolowanie ruchu. W tym przypadku ruch tworzą pakiety przesyłane po sieci natomiast kontrolowanie dotyczy ograniczenia jego szybkości. Z grubsza i w teorii tyle.

W praktyce chodzi o to, aby:
- ograniczyć buractwo korzystające z programów typu Kazaa
- zapewnić sprawiedliwy (przeważnie) dostęp do Internetu
- nie dopuścić do sytuacji w której jeden z użytkowników blokuje innym dostęp do czegokolwiek poprzez jednoczesne ściąganie 3 filmów i 10 mpTrójek
- zapewnienie gwarantowanej szybkości połączenia z Siecią serwera(ów)

Nie będę się dalej rozwodził nad pierdołami, ponieważ Ty – administrator – i tak wiesz co chcesz osiągnąć.

2. HTB

HTB jest stosunkowo nowym algorytmem kolejkowania. Mówiąc wprost, to od niego zależy zapewnienie dokładnie takiego podziału pasma, jakiego sobie zażyczyłeś. Podobne możliwości jak opisywany tutaj HTB posiada CBQ. Dlaczego więc wybrałem HTB i polecam go Tobie ? Ponieważ HTB:

- jest szybszy (szybsze wysłanie pakietów, mniejsze obciążenie serwerka)
- jest bardziej dokładny (można powiedzieć dużo bardziej dokładny)
- łatwiej zrozumieć jego składnię i sam sposób działania

Tyle plusów wystarczy, teraz z drugiej beczki. Zagadnienie zarządzania przepustowością sieci jest złożone. Dlatego podaje dodatkowe źródła informacji, po które powinieneś sięgnąć:

- http://echelon.pl/kravietz/ – dokument pt. „Sterowanie przepływem danych w Linuxie”. Praca raczej teoretyczna ale niezbędna. Nie wczuwaj się za bardzo w opis CBQ. Język: polski, przeczytaj przynajmniej 5 razy.

- http://luxik.cdi.cz/~devik/qos/htb/ - strona domowa HTB. Ściągaj i czytaj wszystko, bez ograniczeń ilościowych. Język: angielski.

- http://docum.org/ - dużo dodatkowych informacji i przykłady. Czytaj ! Język: angielski.

Powyżej podałem najciekawsze (moim zdaniem) źródła informacji na temat kształtowania ruchu w kontekście HTB. Jeśli będziesz miał pytania bez odpowiedzi, poszukiwania zacznij od ww. adresów.

3. Wyzwanie: instalacja HTB i odpowiednich narzędzi

Instalacja HTB nie jest normalna. Już wyjaśniam: obecnie HTB obsługują wyłącznie jądra serii 2.4.x. Zdziwiony ? Ja też, i jeśli mam być szczery HTB to jedyny powód, dla którego zmieniłem jajko 2.2 na 2.4. Kolejna wiadomość: dopiero jajko 2.4.20 ma włączoną obsługę HTB, dla poprzednich należy je patchować. Zalecam instalację przynajmniej 2.4.20. Myślę, że to dobry moment abyś siadł do kompa i zaczął ściąganie najnowszego jajka (http://www.kernel.org).

Jajkiem zajmiemy się za chwilę, teraz kolej na resztę potrzebnego softu. Mowa tu o pakiecie iproute2. Pocieraj więc pod ftp://sunsite.icm.edu.pl/pub/linux/iproute/.

Lecimy dalej: sedno sprawy. Ze strony domowej HTB pobierz... htb (http://luxik.cdi.cz/~devik/qos/htb/). Okej.

Kolejne, ostatnie już narzędzie, jest opcjonalne ale zalecane. Mowa tu o programie ttcp. Świetnie się sprawdza jako tester szybkości sieci a jego obsługa jest banalnie prosta. Adres: ftp://ftp.arl.mil/pub/ttcp/. Jak go tam nie znajdziesz, wejdź do jakiejś wyszukiwarki plików (np. www.internauci.pl) i wpisz ttcp.

Cyrk dopiero się zaczyna. Zaczniemy od końca:

- rozpakuj ttcp.tar.gz i wydaj polecenie: gcc ttcp.c –o ttcp
- zobaczysz kilka ostrzeżeń a następnie binarkę
- skopiuj ją w jakieś sensowne miejsce (np. /usr/local/bin/)

ttcp mamy z głowy. Teraz htb i iproute2.

- rozpakuj htb a ujrzysz dwa pliki .diff. To może oznaczać tylko patchowanie.
- jeśli masz jajko => 2.4.20 pozostanie Ci tylko iproute2 do spaczowania.
- tu pojawia się drobny problem. Gdy powstawała ta praca, iproute2 należało spaczować. Być może od tego czasu dorzucono w końcu obsługę HTB. Zobacz do pliku Relnotes. Jeśli ujrzysz gdzieś tam HTB, szafa gra. Jeśli nie, wejdź do katalogu iproute2/ i wydaj polecenie: patch –p1 < ../*ścieżka/do*/htb..._tc.diff.
- pozostało uruchomić make i make install. W razie problemów możesz pogrzebać w pliku Makefile – jest tam kilka rzeczy do zmiany. U mnie (Slack 8.0) nie trzeba było nigdzie grzebać.

Teraz zakładam, że masz jajko i iproute2 (dokładnie programik tc) które obsługują HTB. Czas na kompilację jajka. Ogólne zalecenie jest takie: w Networking Options włącz QoS i zaznaczaj ile wlezie. Na początek wszystko wrzuć na stałe do jajka (nie rób modułów, dopóki nie sprawdzisz, że działa). Z tego, co się orientuję należy mieć tam także włączone ...netlink socket. I jeśli mam być szczery - Bóg jeden wie co jeszcze. Jeśli nie jesteś pewien, bierz wszystko co w opisie (Help) posiada „traffic control”, „traffic shaping”, „HTB” itp.

Potem SZZ (Skompiluj, Zainstaluj, Zrebootuj). Uruchom system na nowym jajku i wykonaj polecenia:

```
tc qdisc show
tc filter show
tc class show
```

Jeśli któreś z nich zwróci komunikat w stylu „....LINK: no such device" pominąłeś coś w jajku. Dodaj parę sensownych rzeczy, potem SZZ i jeszcze raz.

Jeśli ktoś posiada listę obowiązkowych opcji dla jajka potrzebnych do uruchomienia HTB, proszę o mejla. To był przyspieszony kurs konfiguracji Linuxa tak, aby mógł on pełnić funkcje rozdzielacza (sic!) pasma z wykorzystaniem HTB.

Poprawna instalacja tego wszystkiego potrafi być naprawdę upierdliwa. Tak czy siak, w końcu się uda. Przejdźmy do konkretów...

4. Zaczynamy

A teraz kilka ważnych informacji, które powinny zmienić Twój sposób myślenia na omawiany temat. HTB (podobnie jak CBQ) umożliwia jedynie ograniczenie wychodzącego ruchu.

Chyba jesteś trochę skołowany, ale powoli... Załóżmy taką sytuację:

```
( Internet ) ---------- eth1 SERWER eth0 ---------- LAN
```

W przypadku modemów (np. SDI od największego legalnego złodzieja w kraju – TPsa) miejsce eth1 zastępuje ppp0.

Skoro możesz jedynie ograniczać ruch wychodzący, to operacje na eth1/ppp0 dotyczyć będą ograniczenia ruchu z serwera do Internetu... Przeważnie nie o to chodzi...

Ale spójrz na to z drugiej strony: ograniczenia nakładane na eth0 dotyczyć będą ruchu z serwera do sieci lokalnej. A ponieważ serwer jest bramką do Inetu i cały ruch z Netu do LANu przechodzi właśnie przez serwer, dlatego też chcąc ograniczać Internet, działasz na interfejsie, przez który Internet „wychodzi" z serwka do LANu.

Innymi słowy: działając na interfejsie po stronie LAN ograniczasz Internet dla sieci LAN (wszystko, co jest ściągane przez sieć lokalną podlega ograniczeniu). Działając na interfejsie po stronie Internetu, ograniczasz wszystko, co do niego wychodzi, np. z serwera.

Prawdopodobnie od początku czaisz o co chodzi, ale chcę uniknąć nieporozumień. Pewnie masz w tej chwili 128 pytań na temat tego wszystkiego, ale spokojnie...

Przyspieszony kurs obsługi ttcp.

Nie jest to program wymagany, ale bardzo przydatny. Oprócz niego będziesz potrzebował dwóch Linuxów połączonych siecią. Na jednym i drugim instalujesz programik ttcp.

- po stronie odbiorcy piszesz: ttcp –r –s
- po stronie nadawcy: ttcp –t –s <nazwa_kompa_odbiorcy>

Najlepiej to sprawdzić w sieci lokalnej ze względu na stosunkowo dużą ilość domyślnie przesyłanych pakietów. Inne przydatne parametry programu (pamiętaj, że większość z nich należy zmienić zarówno po stronie nadawcy i odbiorcy):

-p <numer> :zmiana numeru portu (domyślnie 5001)
-l <długość> :długość buforów, wysyłanych do / pobieranych z, sieci (8192)
-n <ilość> : ilość buforów wysyłanych do / pobranych z sieci (2048)
-f <format> : jednostki danych wyjściowych (domyślnie kilobajty)

Myślę, że jest to dobry moment, aby poruszyć temat bitów, bajtów oraz przepustowości sieci. O ile podstawową jednostką w przypadku pamięci RAM czy dysków twardych jest bajt, o tyle w przypadku sieci korzysta się z bitów. Dla przypomnienia: 1 bajt = 8 bitów :)

Najczęściej spotykane sieci lokalne mają szybkość 10 lub 100 megabitów. Z Internetem łączymy się najczęściej z szybkością 56 kilobitów (modem rulez), 112 kilobitów (SDI rulez) lub np. 1 megabita (Polpak-T rulez). A wszystko to najczęściej poprzez Tpsa („to ludzka rzecz – okradać").

Powinieneś wiedzieć, z jaką szybkością działa Twoja sieć lokalna oraz jaka jest maksymalna szybkość Twojego połączenia z Netem (w tym przypadku ważne jest także czy i jak rozkłada się ta szybkość – np. 12 kbit na wychodzące a 100 kbit na przychodzące - choć prawdopodobnie nie masz ograniczeń w tym stylu).

Moja sieć lokalna zbudowana jest na kablu koncentrycznym (kto go jeszcze pamięta, kto go jeszcze używa ?) o teoretycznej szybkości 10 mbit / sek. Za chwilę to sprawdzę z wykorzystaniem ttcp i dwóch Linuxów.

Odbiorca: **ttcp –r –s –f mbit**
Nadawca: **ttcp –t –s <nazwa_odbiorcy>**

Oto weryfikacja szybkości działania mojego LANu (komunikaty odbiorcy):

ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001 tcp
ttrp-r: socket
ttcp-r: accept from 192.168.2.2
ttcp-r: 16777216 bytes in 14.85 real seconds = ***8.62 Mbit/sec*** *+++*
[i tu różne pierdoły]

Chyba wystarczająco dokładny pomiar...

Posunę się do stwierdzenia, że aby uruchomić HTB należy wykonać dwa kroki. Ale za to jakie... Pierwszym z nich jest podział „szybkości wychodzącej” na mniejsze, odpowiadające konkretnym usługom, odbiorcom itp. Drugim ruchem jest założenie odpowiednich filtrów. Filtry odpowiadają za to, aby odpowiednie pakiety wpadały do odpowiednich „podszybkości”, zdefiniowanych w pierwszym kroku.

Podsumowując: w pierwszym kroku konfigurujemy dostępne pasmo dla konkretnych użytkowników czy usług (czyli „dzielimy” łącze). W drugim kroku dbamy o to, żeby odpowiednie pakiety trafiły do właściwego „podłącza”.

Kurwa, język polski jest piękny, ale jak czytam „podłącza” czy „podszybkości” to chce mi się rzygać. Jak się znajdzie ktoś mądry, niech mi podeśle polskie odpowiedniki tych cholernych wynalazków... Dobra, chyba znalazłem: „podszybkości” i te drugie to kolejki i klasy. Czyli dzielimy całe pasmo, np. 56 kbitów na kolejkę i klasy.

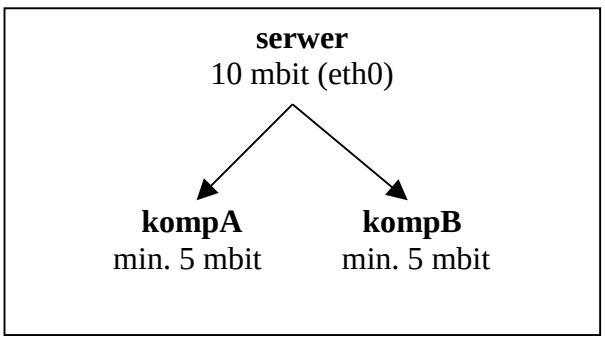
5. Najtrudniejszy pierwszy krok

Jak wspomniałem wcześniej, najpierw musimy podzielić całą przepustowość łącza na kolejkę a ją na klasy. Potem, za pomocą filtrów skierujemy do nich pakiety.

Sytuacja A – prosty przykład, dużo konkretów

Serwer (192.168.2.1) podłączony jest do LAN z szybkością 10 mbit/sek. Klientami są dwa komputery: kompA (192.168.2.2) i kompB (192.168.2.3). Chcemy im dać po 50% łącza (5mbit/sek) ale w ten sposób, że jeśli kompA nie robi nic, kompB dostaje całe łącze. Gdy kompA zaczyna coś robić, szybkość połączenia kompB spada do wyjściowych 50%. I na odwrót.

Schemacik:



Magiczne polecenia:

```
(1)  tc qdisc del root dev eth0
(2)  tc qdisc add dev eth0 root handle 1:0 htb

(3)  tc class add dev eth0 parent 1:0 classid 1:1 htb rate 10mbit ceil 10mbit

(4)  tc class add dev eth0 parent 1:1 classid 1:2 htb rate 5mbit ceil 10mbit
(5)  tc class add dev eth0 parent 1:1 classid 1:3 htb rate 5mbit ceil 10mbit

(6)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 192.168.2.2 flowid 1:2
(7)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 192.168.2.3 flowid 1:3
```

Wygląda strasznie, ale kwestia rozbudowy tego czegoś do bardziej skomplikowanych rzeczy jest prosta jak obsługa notatnika w Windows. Tak więc jedziemy:

Ogólnie:

(1) – kasuje poprzednie ustawienia kolejek, filtrów itd. Bardzo zalecane.
(2) – zakłada główną kolejkę na interfejsie eth0.
(3) – zakłada główną klasę z maksymalną możliwą przepustowością na głównej kolejce.
(4) i (5) – tworzą właściwe podklasy (z odpowiednią przepustowością) dla kompA i kompB.
(6) i (7) – kierują pakiety do odpowiednich kolejek

Szczegółowo:

(1) i (2)

- tc: program pochodzi z pakietu iproute2. Jest narzędziem do konfiguracji kontroli przepływu danych (**t**raffic **c**ontrol).
- qdisc: (ang. q*ueueing discipline*) można przetłumaczyć jako sposób kolejkowania, określona dyscyplina której podlega kolejka.
- del: usuń
- add: dodaj
- dev: (ang. *device*) urządzenie, tutaj: interfejs, karta sieciowa
- root: tutaj: główna, „kolejka – rodzic”
- handle: uchwyt, tutaj: identyfikator kolejki
- 1:0: w tym przypadku identyfikatorem głównej kolejki jest 1:0
- htb: to już chyba zostało wyjaśnione

Podsumowanie: utworzyliśmy główną kolejkę (1:0) dla interfejsu eth0.

(3)

- class: klasa, wykorzystywana przy podziale pasma, głównej kolejki na mniejsze. Ponieważ na razie nie mamy żadnej klasy, w tym miejscu tworzona jest główna klasa, która otrzymuje pasmo swojego rodzica.
- parent: wskazuje rodzica klasy. Argumentem jest identyfikator, uchwyt nadrzędnej kolejki lub klasy.
- 1:0: w tym przypadku rodzicem dla klasy głównej jest kolejka o identyfikatorze 1:0. Czyli tworzymy klasę, która będzie dzielić, zarządzać pasmem swojego rodzica.

- classid: identyfikator klasy. Przez niego będziemy odwoływać się do danej klasy.
- 1:1: to jest właśnie identyfikator aktualnie dodawanej klasy.
- rate: jego argumentem jest minimalna, gwarantowana szybkość klasy.
- 10mbit: w naszym przypadku oznacza to, że klasa 1:1 ma zagwarantowaną, minimalną szybkość 10 megabitów.
- ceil: maksymalna szybkość, jaka dostępna będzie dla danej klasy.
- 10mbit: w naszym przypadku jest to maksymalna przepustowość dla kolejki 1:1. Domyślnie i tak wartość rate = ceil, jednak chcę uniknąć nieporozumień, w ten sposób łatwiej zapanować nad rozrastającym się drzewem klas (o tym później).

Podsumowanie: utworzyliśmy klasę 1:1, której rodzicem jest kolejka 1:0. Zagwarantowaliśmy jej minimalną i maksymalną przepustowość (w tym przypadku są one równe).

(4) i (5)

Nic nowego. Utworzyliśmy dwie podkolejki (1:2 oraz 1:3), której rodzicem jest nasza klasa główna (1:1). Zagwarantowaliśmy im przynajmniej 5mbit przepustowości, czyli podzieliliśmy przepustowość łącza na dwie równe części.

Maksymalna szybkość każdej z nich to 10mbit. HTB zacznie przydzielać jednej z nich pasmo sąsiedniej TYLKO wtedy, gdy ta sąsiednia nie będzie wykorzystywana w całości. Ustawienie w jednym lub drugim przypadku parametru ceil na rate oznaczać będzie, że nawet gdy sąsiednie pasmo jest wolne, kolejka i tak ograniczona będzie do wartości ceil. Jest to wykorzystywane głównie przez ISP, gdy ich klient płaci za określoną szybkość. Za więcej musi wtedy dopłacić. W naszym przypadku nie żałujemy pasma userom. Jeszcze raz zaznaczę, że kolejka dostanie więcej pasma niż jej rate wtedy i tylko wtedy :), gdy jej ceil > rate oraz równoległe kolejki nie wykorzystują swojego pasma w całości. Gdy chcemy na stałe ograniczyć jej maksymalną szybkość, ustalamy ceil = rate.

Podsumowanie: stworzyliśmy ostatnie, dla naszego przykładu kolejki spełniające założenia dotyczące przepustowości. Są to 1:2 oraz 1:3.

I to jest właściwie koniec pierwszego etapu. Kolejki są, jednak nic do nich nie trafia. Czas na etap drugi: skierować odpowiednie pakiety do odpowiednich kolejek.

(6) i (7)

I tym właśnie zajmują się filtry. W moich przykładach wykorzystuje filtr u32, który posiada największe możliości. O innych możesz przeczytać w dokumencie P. Krawczyka.

- filter: oznacza, że będziemy definiować filter. To jest filtr.
- protocol: jego argument definiuje protokół, którego filtr dotyczy.
- ip: w tym przypadku jest to stary, dobry IP (ten od TCP/UDP)
- parent: w przypadku filtra, wskazuje główną kolejkę klasy.
- u32: nazwa filtra, który będziemy wykorzystywać
- match: dotyczy właśnie u32. Oznacza tyle, co „dopasuj”
- ip dst: argumentem jest adres IP przeznaczenia, czyli odbiorcy. W naszym przypadku są to adresy kompA i kompB.
- flowid: jego argument wskazuje docelową klasę, która spełni kryterium filtra. W naszym przypadku są to klasy 1:2 i 1:3.

Podsumowanie: ponieważ cały czas operujemy na danych wychodzących z danego interfejsu, najprościej zdefiniować filtr dla adresów docelowych. Tak więc dane wysyłane do kompA trafiają do klasy 1:2 (z całym dobrodziejstwem jej inwentarza) a pakiety przeznaczone do kompB lądują w klasie 1:3.

No i to by było tyle – wszystko powinno już działać. Na razie, zaraz się rozkręcimy. Przedstawione zostało tutaj wiele nowych pojęć i pewnie jesteś trochę skołowany. Mam nadzieję, że nie popełniłem nigdzie w opisie głupiego błędu, „literówki". Przestudiuj sobie ten przykład – jest on kluczem do reszty. Choć założenie jest proste, rozwiązanie problemu dostarcza prawie wszystkich potrzebnych do szczęścia słów kluczowych. Za chwilę zrobimy podobny, choć trochę bardziej złożony przykład, jednak to dobre miejsce aby dorzucić kilka cennych uwag / wskazówek:

- pamiętaj, że ograniczasz ruch wychodzący z interfejsu
- testy zacznij od prostych reguł, coraz bardziej je rozbudowując i sprawdzając
- od początku, w definicji klasy głównej użyj 95% jej teoretycznej przepustowości. Po co tworzyć zatory na modemie / routerze, skoro można ich uniknąć ? Zamiast 10mbit użyj więc 9500kbit. Tutaj nie robię tego z wygody, w praktyce stosuję.
- nie mieszaj jednostek, od początku używaj takich samych, z kalkulatorem w ręce
- pamiętaj, żeby zawsze rate <= ceil
- pamiętaj, żeby zawsze rate dziecka <= rate rodzica
- pamiętaj, żeby zawsze suma rate wszystkich dzieci <= rate rodzica

Na zakończenie tego punktu pracy, kilka informacji o kolejkowaniu w filtrach. Jak możesz zauważyć, filtry decydują o tym, które pakiety trafią do której klasy. A jak one są obsługiwane ? Ano na zasadzie FIFO (pierwszy, który do niej trafił, zostanie pierwszy wysłany). Jeśli zależy Tobie na tym, aby dodatkowo pakiety trafiające do filtra były obsługiwane na zasadzie „wszystkim po równo", powinieneś dodać do każdej klasy dodatkowy filtr: SFQ.

A chodzi w tym o to: HTB zapewnia podział pasma: nikt nie dostanie więcej, niż to zdefiniujesz. W naszym przypadku użytkownik kompA może się jednocześnie np. ftpować i telnetować. Jeśli ma dobre połączenie z FTPem, sam sobie może uniemożliwić telnetowanie. Po stronie HTB jest wszystko o.k.(szybkość będzie ograniczona), ale możemy i tak nie dopuścić do takiej sytuacji korzystając dodatkowo z filtra SFQ. Obsługuje on różne typy połączeń wrzucone do jednej docelowej klasy (tu 1:2) po kolei, co uniemożliwi w naszym przypadku przyblokowanie telnetu przez ftp. Czyli SFQ sprawiedliwie obsługuje wszystkie połączenia wrzucone do jednego wora (czyli klasy).

Naprawdę polecam SFQ, żebyś nie musiał co głupszym userom tłumaczyć „kazą zapchałeś sobie całe pasmo kretynie, dlatego nie możesz sobie oglądać stron z gołymi babami". Jeśli zastosujesz dodatkowo SFQ, to nie dość, że userzy będą ograniczeni (HTB) to dodatkowo idioci sami sobie nie zapchają przydzielonego pasma jedną usługą. I to jest dopiero zajebiste !

Oto, co należy dodać w naszym przypadku aby zamiast FIFO, stosowane było SFQ dla kompów kompA i kompB:

```
tc qdisc add dev eth0 parent 1:2 handle 2:0 sfq perturb 10
tc qdisc add dev eth0 parent 1:3 handle 3:0 sfq perturb 10
```

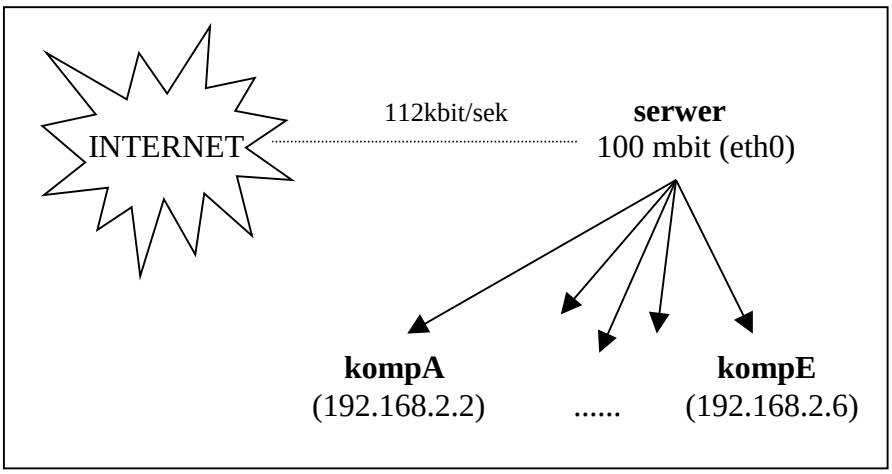
Krótkie omówienie: należy zastosować qdisc na określoną klasę, wskazuje ją parent. Handle może być czymkolwiek, byle nie kolidowało z już istniejącymi (u nas istnieje tylko 1:x). sfq wskazuje na to, że ma być użyte SFQ :) czyli sprawiedliwe dopuszczanie pakietów do kolejki. Natomiast perturb 10 jest wartością zalecaną przez autora SFQ i nie ma się w niego co wgłębiać.

I jeszcze jedno:
- jeśli zaczniesz czytać różne dokumenty na temat podziału pasma może Ci przyjść do głowy dodatkowe ograniczenie którejś kolejki na sztywno za pomocą TBF. Nie rób tego, po prostu ustaw rate = ceil przy definicji kolejki. Nie dość, że działa to szybciej i dokładniej, to jeszcze nie obciąża dodatkowo procesora. Poza tym TBF może mieć niezły rozrzut i możesz wszystko popsuć.

Myślę, że powinieneś sobie zrobić przerwę. Wypij kawę, zapal fajkę, idź na piwo...

Sytuacja B – drzewko rośnie

Sytuacja przedstawia się następująco: serwer połączony jest poprzez SDI do Netu (112kbit/sek) przez interfejs ppp0 oraz do sieci lokalnej (100mbit/sek) poprzez eth0. Dodatkowo pełni on (serwer) funkcję serwera plików dla LAN. Niech sieć lokalna składa się z 5 komputerów: kompA...kompE. Obowiązuje zasada „wszyscy są równi”.



Chcemy podzielić 100 kilo z SDI oraz 95 megabitów z serwera plików na całą sieć lokalną po równo. Oto wymagane regułki:

```
# standardowy początek, z odpowiednimi szybkościami
(1)   tc qdisc del root dev eth0
(2)   tc qdisc add dev eth0 root handle 1:0 htb

(3)   tc class add dev eth0 parent 1:0 classid 1:1 htb rate 99000kbit ceil 99000kbit

# podział całego pasma
(4)   tc class add dev eth0 parent 1:1 classid 1:2 htb rate 110kbit ceil 110kbit
(5)   tc class add dev eth0 parent 1:1 classid 1:3 htb rate 98000kbit ceil 98000kbit

(6)   tc class add dev eth0 parent 1:2 classid 1:4 htb rate 20kbit ceil 100kbit
(7)   tc class add dev eth0 parent 1:2 classid 1:5 htb rate 20kbit ceil 100kbit
(8)   tc class add dev eth0 parent 1:2 classid 1:6 htb rate 20kbit ceil 100kbit
(9)   tc class add dev eth0 parent 1:2 classid 1:7 htb rate 20kbit ceil 100kbit
(10)  tc class add dev eth0 parent 1:2 classid 1:8 htb rate 20kbit ceil 100kbit

# filtry
(11)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip src 192.168.2.1 flowid 1:3

(12)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 192.168.2.2 flowid 1:4
(13)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 192.168.2.3 flowid 1:5
(14)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 192.168.2.4 flowid 1:6
(15)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 192.168.2.5 flowid 1:7
(16)  tc filter add dev eth0 protocol ip parent 1:0 u32 match ip dst 192.168.2.6 flowid 1:8

# sprawiedliwy dostęp do kabla wielu jednoczesnych połączeń
(17)  tc qdisc add dev eth0 parent 1:3 handle 3:0 sfq perturb 10

(18)  tc qdisc add dev eth0 parent 1:4 handle 4:0 sfq perturb 10
(19)  tc qdisc add dev eth0 parent 1:5 handle 5:0 sfq perturb 10
(20)  tc qdisc add dev eth0 parent 1:6 handle 6:0 sfq perturb 10
(21)  tc qdisc add dev eth0 parent 1:7 handle 7:0 sfq perturb 10
(22)  tc qdisc add dev eth0 parent 1:8 handle 8:0 sfq perturb 10
```

Mam nadzieję, że się nie przestraszyłeś...

(1) i (2)

podobnie jak w przypadku sytuacji A.

(3)

To samo, co w sytuacji A. Utworzona zostaje klasa 1:1, która przejmuje całe pasmo interfejsu eth0. Przypominam, że 100mbit = 100000kbit.

Jeszcze jedno: nie wczuwam się w obliczenia typu „ile to jest 1024x1024” bo przypuszczam, że wiesz o co chodzi. Nie jest to praca na temat przeliczania jednostek a ja nie mam na to czasu. Zdecydowałem się na użycie 99 000 kbitów zamiast 100 000 kbitów żeby było to wszystko bardziej czytelne. I tak są to prędkości teoretyczne...

(4) i (5)

W końcu zaczyna się coś dziać. Przepustowość interfejsu eth0 została podzielona na dwie klasy. Pierwsza z nich (1:2) przeznaczona jest na SDI czyli połączenie z Internetem. Druga natomiast (1:3) przeznaczona jest na przerzucanie plików po sieci lokalnej i inne usługi serwera dla LAN. Nie przejmuj się szybkościami w przypadku LAN – i tak nie będą osiągnięte ;) Jeśli chcesz, zapodaj sobie 98888kbit.

(6) – (10)

Dzielimy przepustowość SDI na pięć komputerów. Ważna uwaga: każdy z nich wymaga indywidualnej kolejki. Przeznaczamy dla nich po 20kbit. Większy ceil oznacza, że jeśli inne komputery nie wykorzystują swojego pasma, zostanie ono podzielone między te, korzystające z Sieci.

Nie zajmujemy się podziałem pasma pozostałych 98mbitów, ponieważ nie chce mi się przepisywać linijek (6)-(10) lekko zmodyfikowanych. Jest także mało prawdopodobne, że nagle wszyscy jednocześnie rzucą się na serwer i zaczną z niego pobierać np. filmy. W przypadku tych 98mbitów wystarczy nałożyć na nie SFQ, aby każdy komp został sprawiedliwie obsłużony.

(11) – (16)

Czas na filtry: co gdzie trafi.

To najciekawsza część tego przykładu. Jeśli nadawcą pakietów jest serwer (192.168.2.1), dane trafiają do części przeznaczonej na ruch w sieci lokalnej, 98mbit (1:3).

Czego więc dotyczą pozostałe filtry ? Pakietów, które wychodzą przez eth0 serwera, ale ich nadawcą nie jest serwer. Mowa tu o pakietach z Internetu. Trafiają one do odpowiednich klas o ograniczonej już przepustowości.

W przypadku filtrów jest podobnie jak w przypadku ipchains / iptables – wygrywa pierwszy pasujący filtr. Czyli: jeśli coś pochodzi od serwera, idzie na 98mbit. Reszta musi pochodzić z Internetu i trafia do odpowiednich klas. Teoretycznie. Aby mieć całkowitą pewność, że tak będzie, powinieneś dopisać coś między protocol ip a parent 1:0 w filtrach (11) – (16). Przy (11) należy dopisać preference 1, przy pozostałych preference 2. Ja tego nie zrobiłem, żeby pomieścić wszystko w jednej linijce. Bez preference też powinno to tak działać, „wygrywa pierwsza pasująca reguła", ale lepiej mieć pewność.

Jeszcze raz wyjaśnię, po co to zamieszanie:
Zarówno pierwszy jak i pozostałe filtry pasują do sytuacji „serwer przesyła dane". Nam zależy jednak, aby dane bezpośrednio od niego, trafiły do 1:3 a nie przypadkiem do np.1:4 co spowodowałoby niesamowite spowolnienie szybkości przesyłanych danych i praktycznie „wyłączenie" Internetu. Jeśli jesteś pewien, że wygra zawsze pierwsza pasująca reguła – możesz pominąć preference. Jeśli jednak możesz tą pewność zapewnić dwoma dodatkowymi wyrazami – dlaczego tego nie robić ?

(17) – (22)

Nic niezwykłego. Oprócz podziału pasma, zapewniamy poprzez SFQ sprawiedliwe obsługiwanie danych płynących w odpowiednich klasach.

Może jeszcze podział użytkowników na równych i równiejszych. Komputery w naszym LAN sprawiedliwie podzielą się dodatkowym pasmem, które nie jest wykorzystywane przez ich sąsiadów. A czy można wskazać stację, która będzie miała pierwszeństwo gdy pojawi się nadwyżka ? Oczywista, służy do tego parametr prio a jego argumentem jest priorytet w dostępnie do dodatkowych danych. Obowiązuje stara zasada – im mniejszy priorytet, tym większe pierwszeństwo. Używa się go przy klasach, a więc bezpośrednio tam, gdzie trzeba.

W naszym przypadku linie (6) - (10) definują klasy ostatniego przeznaczenia. Przykładowo, gdy kompB (192.168.2.3) nie będzie korzystał z Netu, jego 20kbit zostanie rozłożone między pozostałe komputery. A gdyby naszym dobrym znajomym był właściciel kompD (192.168.2.5) ? Wystarczyłoby tak przerobić wspomniane linie:

(6) tc class add dev eth0 parent 1:2 classid 1:4 htb rate 20kbit ceil 100kbit **prio 2**
(7) tc class add dev eth0 parent 1:2 classid 1:5 htb rate 20kbit ceil 100kbit **prio 2**
(8) tc class add dev eth0 parent 1:2 classid 1:6 htb rate 20kbit ceil 100kbit **prio 2**
(9) tc class add dev eth0 parent 1:2 classid 1:7 htb rate 20kbit ceil 100kbit **prio 1**
(10) tc class add dev eth0 parent 1:2 classid 1:8 htb rate 20kbit ceil 100kbit **prio 2**

Taaa.... Gdy zwolni się trochę pasma, pierwszym komputerem który z niego skorzysta będzie kompD. Dopiero gdy on będzie miał dość (a z tego co otrzyma ekstra coś jeszcze zostanie), resztki trafią do pozostałych.

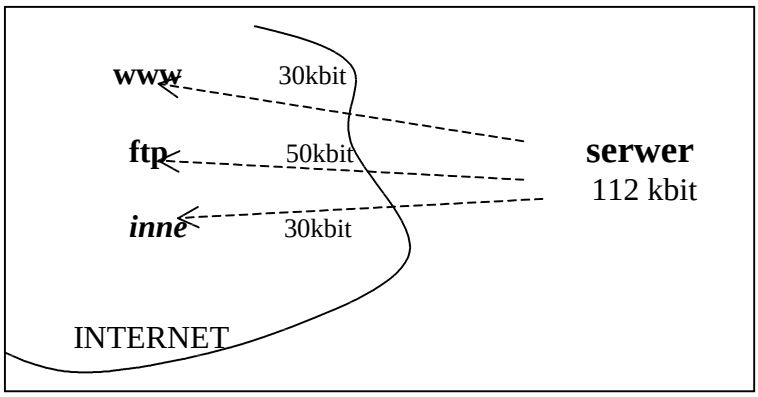
Sytuacja C – z drugiej beczki

Mamy już z grubsza rozeznanie jak udostępniać Internet. Czas pokazać w jaki sposób udostępnić Internetowi. W naszym chorym (w dużej mierze dzięki TPsa) kraju serwerki stawiane są nawet na SDI, więc poruszenie tego tematu uważam za „na miejscu”.

Jeśli masz odgórnie ograniczoną przepustowość wychodzącą – nie ma problemu ;). Jeśli tak nie jest, sam musisz zdecydować jaką część pasma poświęcisz na „wychodzące” dane. Coś na pewno musi wychodzić (zapytania DNS, potwierdzenia TCP), na pytanie „ile” odpowiedz sobie sam. W tym przypadku zakładam, że jest to maksymalna prędkość 112kbit/sek.

O ile w przypadku ograniczania Internetu możesz kierować się adresami IP i na ich podstawie przydzielać szybkość, o tyle w przypadku stawiania serwera Internetowego należy skoncentrować się na konkretnych usługach. Dla omawianego przykładu wybrałem WWW (min. 30kbit), FTP (min. 50kbit) oraz resztę (min. 30kbit). Zakładam również, że urządzeniem wyjściowym w świat jest tym razem ppp0.

Schemacik:



A wszystko wygląda tak:

```
# po staremu
(1)   tc qdisc del root dev ppp0
(2)   tc qdisc add dev ppp0 root handle 1:0 htb default 4

(3)   tc class add dev ppp0 parent 1:0 classid 1:1 htb rate 112kbit ceil 112kbit

# podział całego pasma: www, ftp, inne
(4)   tc class add dev ppp0 parent 1:1 classid 1:2 htb rate 30kbit ceil 112kbit
(5)   tc class add dev ppp0 parent 1:1 classid 1:3 htb rate 50kbit ceil 112kbit
(6)   tc class add dev ppp0 parent 1:1 classid 1:4 htb rate 30kbit ceil 60kbit

# filtry: www i ftp
(7)   tc filter add dev ppp0 protocol ip parent 1:0 u32 match  ip sport 80 0xffff  flowid 1:2
(8)   tc filter add dev ppp0 protocol ip parent 1:0 u32 match  ip sport 20 0xffff  flowid 1:3

# wszystkim po równo
(9)   tc qdisc add dev ppp0 parent 1:2 handle 2:0 sfq perturb 10
(10)  tc qdisc add dev ppp0 parent 1:3 handle 3:0 sfq perturb 10
(11)  tc qdisc add dev ppp0 parent 1:4 handle 4:0 sfq perturb 10
```

I to by było na tyle. Jedziemy:

(1) – (2)

oprócz zmiany urządzenia na ppp0 pojawiło się słowo default. Jego wartość (tutaj 4) oznacza, do której klasy wpadną wszystkie pakiety, które nie zostaną przechwycone przez żaden filtr. Ponieważ numerem głównej kolejki jest 1(:0), domyślną klasą docelową jest klasa 1:4 (1 z handle, 4 z default). Zaraz się wszystko wyjaśni, spokojnie.

(3)

nic nowego, nie licząc zmiany szybkości :)

(4) – (6)

utworzenie klas docelowych z odpowiednimi ograniczeniami. (4) dla www, (5) dla ftp, (6) dla reszty. Zauważ, że nawet gdy nikt nie ciągnie przez www ani ftp, maksymalną szybkością wychodzącą dla *innych* jest 60kbit. Zrobiłem tak ponieważ nie lubię Kazaa – jeśli ktoś będzie coś brał ode mnie, w idealnych dla niego warunkach będzie to 60kbit. Wychodząca Kazaa to m.in. *inne*.

(7) - (8)

to część która najwięcej wnosi w omawianym przypadku. Po pierwsze, pokazane zostało kolejkowanie według źródłowego portu (80 dla www, 20 dla ftp). Na tej podstawie odpowiednie usługi trafiają do odpowiednich klas.
Parametr 0xffff oznacza: „nie patrz na TOS nagłówka IP". Czyli wszystko co pochodzi z portu 80 trafia do 1:2 i wszystko co pochodzi z 20 trafia do ftp.

Na pewno zwróciłeś uwagę na to, że nigdzie nie ma definicji usługi „cała reszta", „inne", które to pakiety powinny trafić do 1:4. A teraz przypomnij sobie co napisałem na temat parametru default kilka akapitów wyżej. Wyjaśniła się sprawa ? Git.

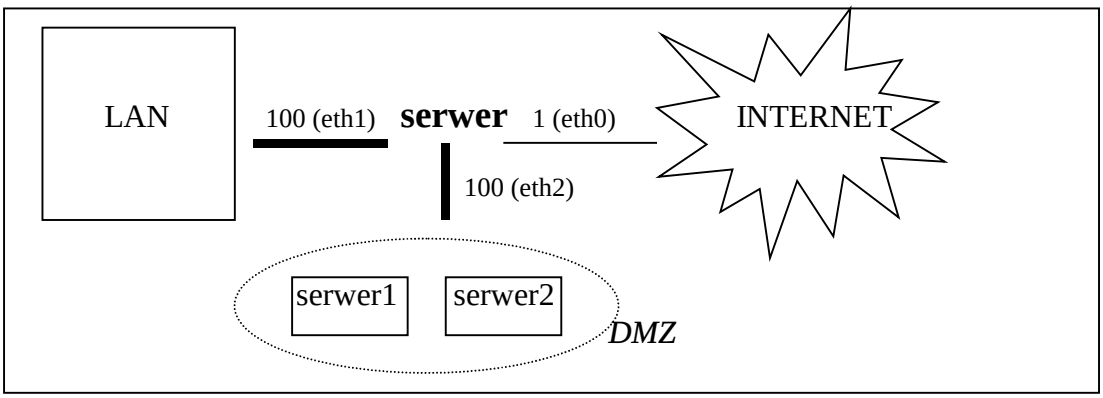
(9) – (11)

reszta jest piosenką :) Do usług dopuszczamy wszystkich klientów sprawiedliwie.

Hmmm....tyle na ten temat. Jako ostatni przykład podam bardziej złożoną konfigurację.

Sytuacja D – ćwiczenie na zrozumienie

Jest ona żywcem zerżnięta z docum.org ponieważ bardzo mi się podoba. Dodałem jedynie komentarze od siebie.



Serwer posiada 3 karty sieciowe 100mbit/sek. Transfer z LAN do DMZ powinien być możliwy z pełną prędkością.

Pamiętaj, że jeśli nie podano parametru ceil, przyjmuje on domyślnie wartość taką, jak rate (proponuję jednak określać to wprost).

Może jeszcze max. szybkość "wchodzącego" Internetu: 2mbit oraz "wychodzącego": 1mbit. No i adresy ip serwera1: 1.1.1.1 oraz serwera2: 1.1.1.2 (ktoś tu poszedł na łatwiznę) :)

Regułki dla LAN (eth1)

```
# wybrano 10:0 na identyfikator kolejki (co kto lubi)
tc qdisc del root dev eth1
tc qdisc add dev eth1 root handle 10:0 htb default 20

# główna klasa
tc class add dev eth1 parent 10:0 classid 10:1 htb rate 100mbit

# klasy docelowe
tc class add dev eth1 parent 10:1 classid 10:10 htb rate 98mbit ceil 100mbit
tc class add dev eth1 parent 10:1 classid 10:20 htb rate 1mbit ceil 2mbit

# filtrowanie
tc filter add dev eth1 parent 10:0 protocol ip u32 match ip src 1.1.1.1 flowid 10:10
tc filter add dev eth1 parent 10:0 protocol ip u32 match ip src 1.1.1.2 flowid 10:10
```

Oszczędzono sobie dołożenia SFQ. Domyślną klasą jest 10:20, do której wpada Internet - jego maksymalną szybkością jest 2mbit.

Regułki dla DMZ (eth2)

```
# wybrano 20:0 na identyfikator klasy dla eth2
tc qdisc del root dev eth2
tc qdisc add dev eth2 root handle 20:0 htb default 20

# główna klasa dla eth2
tc class add dev eth2 parent 20:0 classid 20:1 htb rate 100mbit

# podobnie jak dla eth1
tc class add dev eth2 parent 20:1 classid 20:10 htb rate 98mbit ceil 100mbit
tc class add dev eth2 parent 20:1 classid 20:20 htb rate 2mbit ceil 2mbit


# filtry: zakładając, że komputery w LAN mają adresy 192.168.2.x
tc filter add dev eth2 parent 20:0 protocol ip u32 match ip src 192.168.2.0/24
                                                                    flowid 20:10
```

Podobnie jak dla eth1, klasa 20:20 jest domyślna (jest to Internet). Podobnie również postąpiono nie dodając SFQ. To wcale nie jest taki złożony przykład...

Regułki dla Internetu (eth0)

```
# wybrano 30:0 na identyfikator klasy dla eth0
tc qdisc del root dev eth0
tc qdisc add dev eth0 root handle 30:0 htb default 22

# główna klasa dla eth0
tc class add dev eth0 parent 30:0 classid 30:1 htb rate 1mbit
```

```
# wychodzący Internet: min. 256kbit dla serwer1 i serwer2
tc class add dev eth0 parent 30:1 classid 30:10 htb rate 512kbit ceil 1mbit
tc class add dev eth0 parent 30:10 classid 30:11 htb rate 256kbit ceil 1mbit
tc class add dev eth0 parent 30:10 classid 30:12 htb rate 256kbit ceil 1mbit

# wychodzący Internet: min. 256kbit dla serwer1 i serwer2
tc class add dev eth0 parent 30:1 classid 30:20 htb rate 512kbit ceil 1mbit
tc class add dev eth0 parent 30:20 classid 30:21 htb rate 256kbit ceil 1mbit
tc class add dev eth0 parent 30:20 classid 30:22 htb rate 256kbit ceil 1mbit

# filtry:
tc filter add dev eth0 parent 30:0 protocol ip u32 match ip src 1.1.1.1 flowid 30:11
tc filter add dev eth0 parent 30:0 protocol ip u32 match ip src 1.1.1.2 flowid 30:12
tc filter add dev eth0 parent 30:0 protocol ip u32 match ip dport 80 0xffff flowid 30:21
```

Myślę, że jest to dobry przykład aby sobie pogłówkować. Ja nie mam już siły opisywać kolejnej podobnej rzeczy i udaje się do wyra. Dobranoc !

7. Czy to już koniec ?

Na razie tak. Jeśli posiadasz doświadczenie w konfiguracji HTB lub/i wskazówki, które powinny się znaleźć w tym dokumencie, napisz do mnie! Załącz skrypt, który wykorzystujesz do konfiguracji i napisz informacje na temat Twojej sieci (np. na temat szybkości i tego, co skrypt powinien robić).

Jeszcze parę rzeczy:

- do oglądania statystyk w czasie pracy HTB wykorzystaj: tc –s <qdisc | filter | class>
- jak tylko możesz, korzystaj z ttcp do sprawdzania szybkości
- aby ułatwić przerabianie skryptu i zmniejszyć prawdopodobieństwo wystąpienia błędów, możesz użyć zmiennych (np. USER=20kbit a potem .....$USER)
- ściągnij i przeczytaj dokumentację z linków, które podałem na początku pracy
- jeśli dowiesz się czegoś ciekawego, napisz do mnie, dorzucę to do tej pracy

Przyjemnej zabawy z HTB !

Jest to pierwsza wersja tego dokumentu i zawiera pewnie jakieś błędy / niedoróbki / potknięcia itp. Będę bardzo wdzięczny za wszelkie uwagi, komentarze, poprawki, wyrazy wdzięczności. Z góry przepraszam za powyższe i mam nadzieję, że będziecie mnie informować w których miejscach coś nie gra. Jako (częściowe) wytłumaczenie przedstawiam fakt, że większość tej pracy powstawała w późnych godzinach nocnych.

Kontakt ze mną: linio@terramail.pl

Henryk Liniowski
http://linio.terramail.pl
Poznań, 2002
