Wstęp

Konfiguracja opisywanej tu wersji NieShapera może przysporzyć pewnym problemów, stąd po wstępnym przyjrzeniu się dołączonym przykładowym pliką konfiguracyjnym (które same w sobie są całkiem niezłym wprowadzeniem), należy zapoznania się z niniejszą dokumentacją, w przeciwnym wypadku może nie udać się stworzyć optymalnie działającego podziału łącza. Dlatego bardzo rzadko odpisuję na listy pisane w stylu: "nie działa", "nie uruchamia się", "nie potrafię zainstalować", czy jakże dowcipne: "wyskakuje jakiś błąd", takie posty najczęściej lądują w /dev/null. Za to z przyjemnością czytam listy od osób które zadały sobie trud by zrozumieć program, i skonfigurować swój serwer do świadczenia swoim klientom wysokiej jakości usługi dostępu do internetu.

Punkty oznaczone (*) mogą nie być dostatecznie przetestowane lub nawet niedokończone.

Wymagania

- w minimalnej konfiguracji niezbędny jest system linuksowy z działającym firewallem iptables oraz pakietem iproute z filtrami u32 i/lub fw.
- NiceShaper nie jest w stanie przewidzieć wszystkich Twoich intencji, więc przyda się znajomość iptables i programu tc.
- opcjonalną częścią programu jest obsługa IMQ.
- wypadało by też wygospodarować kilka wolnych godzin i porządne wiaderko kawy:)

Instalacja

Ściagnięty oraz rozpakowany pakiet kompiluje się w standardowy sposób przez wykonanie
$ make
# make install
w efekcie w katalogu /usr/local/bin pojawi się gotowy program.
Jeśli nie posiadamy jeszcze w systemie plików konfiguracyjnych, można przekopiować pliki przykładowe
cp -R etc/niceshaper0.6 /etc
W przeciwnym wypadku należy upewnić się czy między starą a aktualnie instalowaną wersją, nie zaszły zmiany wymagające ingerencji w te pliki.

Konfiguracja

Składnia plików konfiguracyjnych.

- domyślne wymagane są 2 podstawowe pliki konfiguracji: */etc/niceshaper0.6/config.ns* oraz */etc/niceshaper0.6/class.ns*.
- include file */pelna/sciezka* użyte w tych plikach pozwala włączyć kolejne.
- konfiguracja składa się z "dyrektyw" i bezpośrednio przypisanych im pojedyńczych lub oddzielonych białymi znakami "wartości" np.:

```
  rate 128kb/s
  debug iptables iproute
```

- w ramach "dyrektywy" może pojawić się zbiór par "wartości" przyporządkowanych do konkretnie opisanych "parametrów" np.:

```
  stats file /var/www/stats/nsstats.txt unit kB/s mode 644
```

- w obydwu przypadkach parser pliku automatycznie rozbije podane linie konfiguracji, więc w ramach własnych preferencji powyższe przykłady można zapisać następująco:

```
  rate 128kb/s
  debug iptables
  debug iproute
  stats file /var/www/stats/nsstats.txt mode 644
  stats unit kB/s
```

- manewr ten można wykonać z każdą rozbudowaną dyrektywą.
- jednak nigdy nie wolno rozbijać ani tez zmieniać kolejności parametrów dyrektyw iface oraz class.
- akceptowane jednostki przepustowości to bit, bps lub b/s oznaczające bity na sekundę, B, Bps lub B/s oznaczające Bajty na sekundę, do których można dowolnie dodawać przedrostki 'k|m'.
- gdy jednostka zostaje pominięta użyta zostaje jednostka domyślna czyli b/s (bit na sekundę).
- znak '#' jest komentarzem i odnosi się do reszty linii za nim, zakomentowana część nie będzie rozpatrywana przez parser konfiguracji.
- znaczki '<#' komentarz '#>' jak powyższy są komentarzem, jednak w parze obejmują dowolnie duży wycinek konfiguracji.
- znak ';' jest ekwiwalentem przejścia do nowej linii konfiguracyjnej, pozwala zminimalizować długość pliku konfiguracyjnego klas, czasem pozytywnie a czasem negatywnie wpływa na czytelność.
- wszelkie zmiany w konfiguracji do wejścia w życie wymagają zrestartowania programu.
- poniższa konfiguracja i dostarczona z pakietem są tylko przykładem i nie są optymalne dla każdej sieci.

Główny plik konfiguracyjny, na przykładzie asymetrycznego łącza DSL 4Mbit/512Kbit.

- domyślnie głównym plikiem konfiguracyjnym jest /etc/niceshaper0.6/config.ns.
- podzielony na sekcje, gdzie wymagana jest sekcja globalna oraz minimum jedna sekcja na potrzeby kontroli okreslonego filtrami ruchu.
- dla każdej sekcji w systemie zostanie powołany proces potomny niceshapera, kontrolujący wyłącznie sobie przypisane klasy.

```
<global>
  run download upload
  mark-on-ifaces eth0
  stats unit kB/s
  stats file /var/www/stats/nsstats.txt
  stats owner root group root mode 644
  log syslog true
  log terminal true
  log file none
</>

<download>
  iface eth1 match dstip 192.168.0.0/24
  iface eth2 match dstip 192.168.1.0/24
  section speed 512kB/s
  section shape 450kB/s
  default low 10kB/s
  default ceil 100kB/s
  default htb prio 5
  default htb scheduler sfq
  default hold 30s
```

default hold 30s
iptables hook POSTROUTING
reload 3s
</>

Opcje globalne:

run - lista sekcji na potrzeby których należy uruchomić instancje (muszą być skonfigurowane). wygodne by tymczasowo wyłączyć jedną z sekcji bez potrzeby usuwania lub komentowania konfiguracji.
mark-on-ifaces - lista interfejsów wyjściowych, oddzielonych spacją na których w miejsce filtru u32 należy użyć filtru fw czyli markowania pakietów.
opcja niezbędna by kontrolować upload hostów z adresacją prywatną poddawanych maskowaniu na adres publiczny routera, lub korzystać z możliwości filtrowania które posiadają iptablesy lecz już filtr U32 nie.
stats - wyświetlanie statystyk pracy.
  unit - wyświetlane jednostki przepustowości.
  file {plik|none} - wskazuje pełną ścieżkę do pliku lub wyłącza przez none (wartość domyślna }.
  owner - ustawia systemowego właściciela pliku (domyslnie root).
  group - ustawia systemową grupę do której należy plik (domyslnie root).
  mode - ustawia uprawnienia do pliku w trybie numerycznym (domyslnie 644).
3 ostatnie opcje nie mają zastosowania jeśli automatyczny zrzut nie został włączony (file none).
  classes {all|active|working|false} - sterowanie wyświetlaniem klas w statystykach (domyślnie working).
    all - wyświetla wszystkie skonfigurowane klasy.
    active - wyświetla klasy które wykazały aktywność.
    working - wyświetla klasy które wykazały ostatnią aktywnośc w czasie krótszym od ustawienia parametru hold.
  sum {top|bottom|false} - miejsce wyswietlania sumy obciążenia sekcji.
log - sposoby logowania komunikatów.
  syslog {true|false} - domyślnie true
  terminal {true|false} - domyślnie true, po poprawnej inicjalizacji zostaje automatycznie wyłączone(nie dotyczy debugów)
  file {plik|none} - wskazuje pełną ścieżkę do pliku lub wyłącza przez none (wartość domyślna }.
lang {en|pl} - określa język wyświetlanych komunikatów.

Konfiguacja sekcji:

iface interfejs match test
- przyporządkowujemy określony ruch IP danej sekcji określając go za pomocą filtrów niceshapera. (opis niżej)
- informujemy na którym interfejsie należy założyć kolejkę HTB. Co ważne!! HTB dokonuje kolejkowania WYŁĄCZNIE na interfejsie którym pakiet opuszcza router i ten interfejs należy tu definiować.
- nie występuje już wzorem ns0.5 żadne rozróżnienie na interfejs zewnętrzny i wewnętrzny.(inet/local)
- nalezy zadbac by klasy niceshapera w pełni określały tu zdefiniowany ruch w przeciwnym razie może dojśc do niekontrolowanych przecieków.
- najczęściej określana tu będzie wyłącznie obsługiwana podsieć lokalna, stąd w uproszczonej składni :
iface eth1 dst network 192.168.0.0/24
iface eth0 src network 192.168.0.0/24
- iptables i iproute nie rozróżnia aliasów na interfejsach stąd w przypadku takich interfejsów pomijamy wszystko za dwukropkiem.
- vlany zapisywane w formie ethX.vid obsługiwane są identycznie jak to ma miejsce w linuksie.
eth1 - interfejs fizyczny eth1
eth1 - alias eth1:1
eth1.100 - vlan z vid = 100 na interfejsie fizycznym eth1
section - ogólne parametry pracy sekcji (zwykle parametry łącza lub przyporządkowanej części).
  speed - fizyczna wydajność pasma.
  shape - poziom na jakim chcemy utrzymać obciążenie.
- ważne by dobrać wartość o kilkanaście procent mniejszą od faktycznej wydajności naszego łącza. Jeśli zbyt wysoko ustawimy ten parametr, będzie cierpieć ruch interaktywny. - bardzo częstym błędem jest określanie tu tak wysokiej wartości że realne obciążenie nigdy nie sięga tej wartości, NiceShaper nie będzie spełnia swojej roli, a użytkownicy otrzymają maksymalne przydziały. Dla NiceShapera pietrwszym sygnałem do "obcinania", jest właśnie wykrycie przekroczenia tej wartości.
reload - okres taktowania, w sekundach.
wartość domyślna 4s jest bezpieczna i w miarę efektywna dla każdej maszyny klasy pentium I.
na maszynach z szybszym procesorem warto zwiększyć częstotliwość wykonywania nawet do 2s, co wyraźnie zwiększa interaktywność. Wartośc parametru musi się mieścić w przedziale 0.1s do 600s z krokiem nie mniejszym niż 0.1s.
mode - drobna opcja dla iptables ale w tej postaci wydaję się być bardziej czytelna dla nieznających w minimalnym stopniu tego firewall'a, a za zadanie ma ona określić czy głowne filtry danej sekcji mają zostać ulokowane:
  upload - w łańcuchu PREROUTING.
  download - w łańcuchu POSTROUTING.
Jest to jednak wyłącznie alias dla iptables hook.
iptables - parametry odnoszące się bezpośrednio do iptables w systemie.
  hook - opcja stojąca za uproszczonym mode, dużo lepiej oddaje swoją funkcjonalność.
  append-hook - reguła zostaje dopisana na końcu łańcucha (nawiasem mówiąc jest to domyślne zachowanie iptables hook).
  insert-hook - reguła zostaje wstawiona na początek wskazanego łańcucha.
debug - wyświetla przekazywane do systemu instrukcje.
  iptables - wyświetla instrukcje przekazywane iptables.
  iproute - wyświetla instrukcje przekazywane do programów z pakietu iproute (tc, ip).
default - definiuje parametry domyślne dla wszystkich klas w ramach sekcji, które mogą być następnie indywidualnie modyfikowane w pliku klas. Parametr nie jest wymagany jednak zwiększa czytelność.

Pojęcie klasy w NiceShaperze

- plik /etc/niceshaper0.6/class.ns zawiera definicję klas, które można uznać za odpowiednik klas w HTB.
- klasa zbudowana jest z nagłówka i dyrektyw konfiguracyjnych.
- pakiet zostaje zakwalifikowany do pierwszej dopasowanej za pomocą filtrów klasy.
- do dyspozycji poza standardową klasą mamy 3 dodatkowo predefiniowane typy:

do-not-shape - klasa tego typu nie zostaje wprowadzona do HTB, powołane zostają jedynie filtry kierujące ruch do kolejki root tak by zdefiniowany w tej klasie ruch nie był w żaden sposób ograniczany. Klasa ta istnieje jednak w iptables dzięki czemu możemy mierzyć wykorzystywane przez nią pasmo.
virtual - posiada wpisy tylko i wyłącznie w iptables. Klasa tego typu służy do mierzenia wykorzystywanego pasma. W przeciwieństwie do poprzedniego typu ruch zakwalifikowany do tej klasy nie partycypuje w utylizacji sekcji a dodatkowo nie zostaje usunięty z łańcucha w iptables i może być dalej zakwalifikowany do jednej z kolejnych fizycznych lub virtualnych klas
wrapper - klasa ta jest przydatna gdy chcemy ograniczyć coś co nie wpływa na wykorzystanie łącza, (np. transfer z lokalnego serwera plików, który kontrolujemy jedynie dlatego by nie przeciążać lokalnej sieci radiowej), więc nie potrzebujemy podziału dynamicznego ani wliczania tego ruchu do wykorzystania łącza.

budowa klasy

przykładowa najprostsza klasa ma postać:

```
class download eth1 Mariusz
  match dstip 192.168.0.77
```

lub w zagnieżdzonej postaci (w dalszej części będzie używany pierwszy czytelniejszy sposób zapisu):

```
class download eth1 Mariusz ; match dstip 192.168.0.77
```

definicje klasy rozpoczyna nagłówek klasy a kończy kolejny nagłówek lub koniec pliku:

```
class sekcja interfejs nazwa
```

sekcja - sekcja której przyporządkowujemy daną klase
interfejs - interfejs na którym realizowane jest kolejkowanie danej klasy przez HTB.
nazwa - nazwa klasy wyświetlana przez stats

Obowiązkowym parametrem ciała klasy jest filtr:

```
match test
```

- filtry definiują ruch którym dana klasa będzie zarządzać.
- każdą klasę może budować dowolna liczba filtrów (minimum jeden).
- naturalnie powyższe testy można ze sobą łączyć by uzyskać bardziej szczegółowe filtry.

parametry klasy:

low - minimalny przydział (domyślnie 0)
ceil - maksymalny przydział (domyślnie równe section shape).
rate - stały przydział pasma.
strict - wartość w procentach { 0% - 100% }, określa jak bardzo nie lubimy przeginających (domyślnie 70).
- wartości niskie będą skutkowały bardziej restrykcyjnym traktowaniem tychże delikwentów.
- wartości bliskie 100% przy przeciążeniu łącza, "współodpowiedzialnością ogółu".
hold - czas nieaktywności po którym klasa zostaje wyładowana z HTB (domyślnie 30s).
iptables - parametry odnoszące się bezpośrednio do iptables w systemie.
  target - {accept|return|drop} cel w iptables, pakietów zakwalifikowanych do filtrów klasy.
    accept - jest to domyslny cel, pakiet po zakwalifikowaniu przez filtry kończy swoja drogę w danym łańcuchu wbudowanym tabeli mangle.
    return - pakiety zakwalifikowane do klasy będą wracały z spowrotem do łańcucha PREROUTING/POSTROUTING np. w celu dalszej obróbki przez firewalla.
    drop - pakiet jest odrzucany, daje to prostą metodę np. na blokowanie klientów nie płacących z poziomu NiceShapera.
htb - parametry odnoszące się bezpośrednio do skonfigurowanych klas w HTB.
  prio - priorytet dla klasy w htb, przyjmuje wartości od 0 do 7, przy czym niższa wartość to wyższy priorytet. (domyślnie 5).
  scheduler - wybór schedulera dla klasy w htb, z pomiędzy obsługiwanych { none|sfq|esfq }(domyslnie sfq).
sfq - konfiguracja schedulera sfq jeśli zostanie użyty w ramach klasy.
  perturb - parametr perturb dla sfq (domyślnie 10)
esfq - konfiguracja schedulera esfq jeśli zostanie użyty w ramach klasy.
  perturb - parametr perturb dla esfq (domyślnie 10)
  hash - hash esfq {classic|src|dst} (domyślnie classic)

By nie powtarzać tych parametrów dla każdej klasy z osobna w ramach konfiguracji sekcji używa się dyrektywy default, a następnie jeśli zaistnieje potrzeba zróżnicowania ustawień klas nadpisuje wybrane z parametrów.

Testy podstawowe:

proto - protokół (tcp, udp lub icmp).
srcip - adres źródłowy.
dstip - adres docelowy.
srcport|sport - port źródłowy (należy wskazać protokół tcp lub udp).
dstport|dport - port docelowy (należy wskazać protokół tcp lub udp).
from localhost - oznacza wszystko co pochodzi z localhosta, szczegółowy opis wykorzystania w rozdziale "Jak traktować localhosta"
to localhost - oznacza wszystko co skierowane jest do localhosta, szczegółowy opis wykorzystania w rozdziale "Jak traktować localhosta"

srcip oraz dstip mogą wskazywać adres ip lub podsieć adresów o zasięgu zdefiniowanym w standardowy sposób przez maskę, zapisaną w formacie bitowym lub kropkowo-dziesiętnym.
Maska nie musi być ciągła (np. 255.255.128.255) jednak należy pamiętać że takiej sytuacji nie obsługuje filtr u32 i dla masek nieciągłych trzeba posłużyć się markowaniem pakietów.

przykładowe filtry:

pakiety skierowane do adresu 192.168.0.77:
match srcip 192.168.0.77
poczta pobierana z interii do podsieci 192.168.0.0/29:
match srcip 217.74.65.69 srcport 110 dstip 192.168.0.0/29 proto tcp

Testy wymagające włączonego markowania na interfejsie:

Iptables został zaopatrzony w olbrzymią liczbę filtrów, które nie są niestety możliwe do zrealizowania przy użyciu filtra U32. Dlatego też poniższe filtry wymagają mark'owania przez mark-on-ifaces. Każdy wychwycony i oznaczony przez iptables pakiet może już być bez problemu skolejkowany do odpowiedniej klasy HTB dzięki filtrowi fw który zostaje użyty w miejsce filtra u32 ten aspekt będzie szczególnie rozbudowywany w kolejnych wersjach testowych i zależny jest od możliwości iptables w systemie.

not-srcip - adres źródłowy inny niż podany.
not-dstip - adres docelowy inny niż podany.
not-srcport|not-sport - port źródłowy inny niż podany (należy wskazać protokół tcp lub udp).
not-dstport|not-dport - port docelowy inny niż podany (należy wskazać protokół tcp lub udp).
length - długość pakietu w bajtach, np. 500, :500(od 0 do 500), 500: (500 i większe), 128:500 (128 do 500).
state - stan pakietu:
  new - pakiet rozpoczyna nowe połączenie
  established - pakiet nalezy do nawiązanego połączenia
  related - pakiet rozpoczynajacy nowe połączenie jednak powiązany z istniejącą konwersacją (np. transfer danych po ftp)
  invalid - pakiet niemoże zostać rozpoznany

untracked - pakiet nie należący do śledzonego połączenia
tos - wartość pola TOS pakietu.
ttl - TTL pakietu równe podanej wartości
ttl-lower - TTL pakietu mniejsze od podanej wartości
ttl-greater - TTL pakietu większe od podanej wartości
mark - pakiety oznaczone podaną wartością

Obsługa markowania pakietów:

Standardowo jeśli w danej sekcji na interfejsie włączone zostanie markowanie pakietów, NiceShaper dla każdego filtra przypisze niepowtarzalny znacznik w iptables.
Jednak jeśli zaistnieje potrzeba jawnego zdefiniowania znaczników w ciele filtra można użyć odpowiednich opcji.
Trzeba tu pamiętać że każdy znacznik użyty w poniższych opcjach zostanie poddany ochronie i NiceShaper w żadnych wypadku nie będzie używał go we własnym zakresie, tak by żaden inny filtr nie otrzymał go automatycznie.

mark - wychwytuje pakiety oznaczone tym znacznikiem a podana wartość zostanie zachowana.
set-mark - podmienia automatycznie przypisaną do filtra wartość znacznika, ma także priorytet nad znacznikiem uzyskanym z użycia opcji mark.

set-mark jeśli użyty zostanie w ciele klasy (jako samodzielny parametr poza filtrem) przeniesie to oznaczenie na wszystkie filtry należące do danej klasy, dając jednak możliwości nadpisania w ramach konkretnego filtra.
przykład:

```
class download eth1 Mariusz
  set-mark 6
  match dstip 192.168.0.77
  match dstip 192.168.0.78
  match dstip 192.168.0.79 set-mark 11
```

Pierwsze dwa filtry zostaną oznaczone znaczkiem 6 a ostatni 11.
Gdyby pierwsza opcja set-mark nie została użyta obydwa filtry miały by znacznik nieokreślony, przyporządkowany automatycznie.

Wykorzystanie interfejsów IMQ

Obecnie interfejsy IMQ konfiguruje się analogicznie jak te fizyczne pomijając ich wirtualność.
NiceShaper automatycznie przekierowuje ruch na interfejs IMQ ,ale ze względu na elastycznośc którą zyskujemy przez możliwość dowolnego kombinowania wielu interfejsów fizycznych i IMQ zachowanie to jest konfigurowalne z poziomu sekcji oraz pojedyńczych klas.

imq autoredirect {true|false} Po wyłączeniu automatycznego przekierowania należy takie wykonać poza niceshaperem.
Dla ułatwienia w ramach konfiguracji sekcji posłużyć się można opcją default która automatycznie doda ten atrybut do wszystkich przyporządkowanych sobie klas:

default imq autoredirect false Co oczywiste zachowanie to znów można niezaleznie modyfikować w ramach poszczególnych klas.

. Przykładowy wycinek pliku config.ns:

```
iface eth1 match dstip 192.168.0.0/24
iface imq5 match dstip 10.10.5.0/24
```

Oraz pliku class.ns:

```
class download imq5 Jaroslaw
  match dstip 10.10.5.82
```

UWAGA!!
w wersjach do rc3 przekierowania należało jawnie zarządać, obecnie na dobrą sprawę użycie interfejsów imq zostaje wykryte i automatycznie oprogramowane, w skrócie od wersji 0.6rc4 opcja przekierowania domyślnie ustawiona jest na true.

Jak traktować localhosta

Ze względu na możliwość wykorzystania takich technologii jak NAT czy IMQ, zagadnienie to jest ciut bardziej zawiłe niż mogło by się wydawać na pierwszy rzut oka. Poza faktem forwardowania ruchu między siecią lokalną a internetem router samemu może pobierać lub wysyłać dane, jeśli komputery sieci lokalnej maskowane są do adresu routera powstaje problem odróżnienia ruchu klientów od generowanego i odbieranego przez samą maszynę routera. W realnym świecie router często jest także serwerem usług, tutaj nie zadowala nas wydzielenie kolejki dla ruchu routera lecz bardziej zaawansowany przydział różnego pasma dla każdej z nich. Na wstępie wspomnę o kilku zasadach:

Rozszerzenie filtrów o dyrektywy from/to localhost zmienia sposób konfigurowania klasy w iptables. Powoduje stworzenie dowiązania z łańcuchów OUTPUT/INPUT, a do samych reguł zostaje dodany także podany interfejs, jako interfejs wychodzący/wchodzący.
Należy pamiętać by klasy tego typu znajdowały się na samym początku chyba że chcemy osiągnąć inny efekt niż jestem w stanie przewidzieć w ramach krótkiej notki.

Tak więc by dobrze przyjrzeć się sytuacji posłużę się czterema możliwymi kombinacjami:

1. wymiana ruchu między routerem a siecią lokalną (niema problemu NATu, adresy źródłowe i docelowe widzimy w oczekiwany sposób):

a) Ruch z localhosta do sieci lokalnej (np. lokalny serwer ftp, samby, pop3).
Sytuacja jest bardzo przejrzysta, tworzymy klasę typu download gdyż to nasze hosty lokalne ściągają dane, można także użyć klasy typu upload, lecz wtedy należy rozszerzyć filtry o opcje from localhost. Dlaczego?
Dla klas typu upload tworzymy łańcuch dowiązany z tabeli PREROUTING, gdzie dane wysyłane przez nasz router nigdy się nie pojawią, from localhost tworzy dowiązanie z tabeli OUTPUT.
Ogólnie bezpiecznie jest zawsze sygnalizować że mamy do czynienia z localhostem, gdyż to właśnie tabela OUTPUT daje nam możliwość 100 procentowego określenia że ruch pochodzi z procesów lokalnych. Dowiązanie do sekcji typu download czy upload jest drugoplanowe, bo z reguły będziemy się starali takiego ruchu nie wliczać do ruchu generowanego przez wymianę z siecią internetową.

```
class download eth1 pop3_z_localhosta
  match from localhost srcip 192.168.0.1 srcport 110 dstip 192.168.0.0/16 proto tcp
  rate 100kB/s
```

do tego zalecane jest by klasa była jednym z poniższych predefiniowanych typów:

```
  do-not-shape # transfery są lokalne więc nie tniemy i nie wliczamy do sumarycznego wykorzystania łącza
  wrapper # transfery są lokalne ale boimy się przypchać radiówkę więc ograniczamy, lecz dalej nie wliczamy do wykorzystania łącza.
```

b) Ruch z sieci do localhosta (np. zapytania do serwera proxy, wysyłanie poczty za pomocą localnego serwera smtp, wrzucanie danych na lokalny serwer plików).
Sytuacja jest bardzo przejrzysta, tworzymy klasę typu upload gdyż to nasze hosty lokalne wysyłają dane, można także użyć klasy typu download, lecz wtedy należy rozszerzyć filtry o opcje to localhost. Dlaczego?
Dla klas typu download tworzymy łańcuch dowiązany z tabeli POSTROUTING, gdzie dane odbierane przez nasz router nigdy się nie pojawią, to localhost tworzy dowiązanie z tabeli INPUT.

Ogólnie bezpiecznie jest zawsze sygnalizować że mamy do czynienia z localhostem, gdyż to właśnie tabela INPUT daje nam możliwość 100 procentowego określenia że ruch zmierza do procesów lokalnych.
Dowiązanie do sekcji typu download czy upload jest drugoplanowe, bo z reguły będziemy się starali takiego ruchu nie wliczać do ruchu generowanego przez wymianę z siecią internetową.

Tu powstaje mały kłopot, filtry definiujemy standardowe lecz, jeśli nie wykorzystujemy interfejsów IMQ nie mamy kontroli nad transferem. Stąd klasa może wyglądać następująco:

```
class upload eth0 smtp_do_localhosta
  match to localhost dstip 192.168.0.1 dstport 25 srcip 192.168.0.0/16 proto tcp
  do-not-shape # gdyż nie jesteśmy w stanie wpłynąć na ten ruch a że nie wpływa on na wykorzystania łącza internetowego nie wliczamy go do niego.
```

Możemy wpłynąć na transfer gdy np. nie chcemy przeciążyć linku radiowego używając interfejsu IMQ:

```
class upload imq0 smtp_do_localhosta
  match to localhost dstip 192.168.0.1 dstport 25 srcip 192.168.0.0/16 proto tcp
  rate 100kB/s
  wrapper
  imq-redirect
```

Oczywiście interfejs IMQ musi być danej sekcji znany, i tu warto wspomnieć o elastyczności do której dąży NiceShaper 0.6, interfejs ten może należeć do sekcji która obsługuje ruch klientów, a może należeć do sekcji stworzonej specjalnie na potrzeby cięcia ruchu lokalnego.
Jednym słowem pełna swoboda.

2. Wymiana ruchu między localhostem a internetem.
a) Ruch z localhosta do internetu (np. zapytania generowane lokalnie słane w świat, poczta, www i inne usługi internetowe na naszym serwerze):

```
class upload eth0 localhost_do_internetu
  match from localhost srcip 80.53.211.226 srcport 80
```

Użycie klasy typu upload i filtrowania z from localhost nie wymaga tutaj szerszego komentarza. Daje nam pewność że właściwie odróżnimy ruch lokalny od hostów w sieci, klasa upload posiada łańcuch dowiązany z PREROUTING'u gdzie interesujące nas pakiety nigdy się nie znajdą więc za pomocą from localhost tworzymy dowiązanie z łańcucha OUTPUT. Dodatkowo wykorzystujemy pasmo wychodzące klientów więc naturalnie musimy ten ruch wliczyć do całkowitego wykorzystania pasma upload, by nie rozregulować dynamicznego podziału.
b) Ruch z internetu do localhosta (np. strony pobierane przez serwer proxy, czy aktualizacje systemu mogące zabierać pasmo klientą):

```
class download eth0 localhost_z_internetu
  match to localhost dstip 80.53.211.226 srcport 80
  match to localhost dstip 80.53.211.226 srcport 21
```

Użycie klasy typu download i filtrowania z to localhost nie wymaga tutaj szerszego komentarza. Daje nam pewność że właściwie odróżnimy ruch lokalny od hostów w sieci, klasa download posiada łańcuch dowiązany z POSTROUTING'u gdzie interesujące nas pakiety nigdy się nie znajdą więc za pomocą to localhost tworzymy dowiązanie z łańcucha INPUT. Dodatkowo wykorzystujemy pasmo przychodzące klientów więc naturalnie musimy ten ruch wliczyć do całkowitego wykorzystania pasma download, by nie rozregulować dynamicznego podziału.
Tu powstaje mały kłopot, filtry definiujemy standardowe lecz, jeśli nie wykorzystujemy interfejsów IMQ nie mamy kontroli nad transferem. Router uzyska bezwzględny priorytet na hostami klientów, sytuacja ta może być przydatna w sytuacji gdy najbardziej cenimy sobie ruch www, który udostępniamy klienta za pomocą serwera proxy, wtedy każdy inny ruch będzie dyskryminowany na korzyść serwera proxy.
Użycie interfejsów IMQ jest identyczne jak w przykładzie dotyczącym ruchu wchodzącego z sieci lokalnej.

To tyle tytułem wstępu do tematu localhosta, z chęcią wysłucham pomysłów na inne bardziej zawiłe sytuacje do których postaram się znaleść rozwiązania celem rozszerzenie powyższego opisu.

Plik userów

Aktualnie NiceShaper obsługuje pliki users z wersji 0.5 programu, składnia rozszerzona wprowadzona w pewnym momencie rozwoju wersji 0.6 została porzucona, ze względu na zbyt duży stosunek wprowadzanego zamieszania do korzysci płynących z jego istnienia.
Plik ten daje możliwość szybkiego zdefiniowania listy adresów ip których ruch chcemy kształtować. Plik users jest jednak nieelastyczny, ograniczna możliwości jakie oferuje NiceShaper w nowszych wersjach, pozwala na użycie 2 sztywnie zdefiniowanych funkcjonalnie sekcji.

Budowa pliku *users* (z NiceShaper'a 0.5)

W wybranym pliku muszą zostać umieszczone adresy IP hostów z sieci lokalnej, oraz oddzielona spacją, nazwa interfejsu je obsługującego. np.:

```
192.168.0.101 eth1
192.168.0.102 eth1
192.168.0.103 eth1
192.168.0.104 eth1 dl_ceil 50kbps
192.168.0.105 eth1 dl_rate 20kbps dl_prio 1
192.168.0.106 eth2 ul_low 3kbps ul_ceil 6kbps
192.168.0.107 eth2
```

dodatkowe parametry pozwalają nadpisać ustawienia globalne dla wybranych przypadków:

dodatkowe parametry pozwalają nadpisać ustawienia globalne dla wybranych przypadków.

dl_low, ul_low - minimalny przydział (dla downloadu i/lub uploadu).
dl_ceil, ul_ceil - maksymalny przydział (dla downloadu i/lub uploadu).
dl_rate, ul_rate - stały przydział (dla downloadu i/lub uploadu).
dl_prio, ul_prio - priorytet klasy w HTB, przyjmuje wartości między 0 a 7, czym niższa wartość tym wyższy priorytet. (dla downloadu i/lub uploadu). Domyślnie ma wartość 5.

Obsługa plików *users*

Tak więc plików tych używac można na 2 sposoby, poprzez uruchomienie programu w trybie convert i wygenerowanie pliku klas, lub przez włączenie w dowolnym miejscu pliku klas.
Pierwsza metoda daje możliwość wzglądu do wygenerowanego pliku klas i ewentualne wprowadzenie modyfikacji, bezpośrednie dołączenie nie daje tych możliwości lecz jest wygodniejsze, zawartość pliku zostanie automatycznie podmieniona na postać klas, jednak metoda ta wymaga większej pewności tego co robimy. Zalecane jest wcześniejsze testowe wygenerowanie klas by upewnić się że będą one zbudowane zgodnie z oczekiwaniami.
Nic nie stoi na przeszkodzie by w głownym pliku klas stworzyć klasy specjalnego przeznaczenia, np. klasy obsługujące hosta lokalnego, ruch priorytetowy itd. a dalej w dowolnym jego punkcie właczyć wygodny w edycji plik users, program automatycznie i niezauważalnie dokona transformacji na klasy.

Jako że zawartość pliku users nie zawiera wszystkich danych niezbędnych do wygenerowania klas, obsługiwane są parametry:

download-section - w NiceShaperze 0.5 istniały dwie wbudowane sekcje o nienaruszalnych nazwach, obecnie nazewnictwo i ilość sekcji jest dowolna stąd wymagane jest poinformowanie konwertera która sekcja pełni rolę sukcesora sekcji download.
upload-section - identycznie jak powyższe lecz dla sekcji upload.
iface-inet - interfejs wychodzący określony był w samym pliku konfiguracyjnym, jako że odpowiednia dyrektywa nie pojawiła się w nowej wersji NiceShapera należy zdefiniować ten parametr opcją.
resolve-hostname - parametr określający, to czy nazwa klasy przybiera postać adresu ip czy nazwy hosta, w przypadku niepoprawnej konfiguracji dns w systemie pobranie adresu hosta może być niemożliwe i znacznie wydłużyć czas uruchamiania, tutaj bezpieczną wartością jest false.

include usersfile /etc/niceshaper0.5/users download-section download upload-section upload iface-inet eth0 resolve-hostname false

Co ważne parametry te czytane są od początku do końca pliku klas z którego dokonuje się włączenia, mogą one być wielokrotnie zmieniane jednak samo włączenie pliku odbywa się w momencie napotkania parametru usersfile, najpóźniej w tej samej linii konfiguracyjnej znane muszą być wartości powyższych parametrów i każdy z nich zostanie użyty w ostatniej napotkanej wersji.

Drugim sposobem uzyskania klas, poprzez uruchomienie konwersji. posługujemy się bardzo podobnie, wydając z linii poleceń komende:

niceshaper convert --usersfile /etc/niceshaper0.5/users --download-section download --upload-section upload --iface-inet eth0 --resolve-hostname true

--outfile - pozwala zdefiniować plik w którym konwerter zapisze utworzone klasy, jeśli nie podano klasy zostaną zrzucone na ekran.

Znaczenie parametrów jest identyczne jak w przypadku włączania, jednak by uczynić zadość tradycyjnym standardom parametrów podawanych z linii poleceń, każdy poprzedzony jest dwoma znakami minusa.