

Monitorowanie ruchu za pomocą rrd i iptables.

Tekst pokazuje prosty sposób na monitorowanie obciążenia łącza do internetu, z podziałem na sieci docelowe lub usługi za pomocą rrd i iptables.

Ostatnio przez Internet przetoczyła się dyskusja spowodowana problemami z transmisją na styku sieci akademickich - Pol-34 a siecią TP S.A. Oczywiście, jak zwykle poszło o pieniądze i dodatkowo, żeby sprawę utrudnić, można było odnieść wrażenie, że poza pieniędzmi chodziło o ambicje dwóch, bardzo dużych jak na nasze polskie warunki, sieci.

W trakcie tej dyskusji przerzucano się statystykami błędów, obciążeń, wielkością styków z danym operatorem, itp. Dodatkowo pojawiło się wiele głosów omawiających kierunki ruchu, w którym idzie większość ruchu internetowego od większych i mniejszych operatorów, pomiędzy jakimi sieciami wymiana ruchu jest zbliżona, gdzie jest asymetryczna itd. Ponieważ sam byłem ciekaw jak to jest naprawdę i kto tak naprawdę korzysta z danych zgromadzonych na moim serwerze, postanowiłem się temu przyjrzeć.

Sytuacja była o tyle skomplikowana, że chodzi o jeden serwer z Linuksem na pokładzie, a nie o router z oprogramowaniem gotowym do generowania takich statystyk za pomocą NetFlow. Ostatecznie okazało się to prostsze niż się spodziewałem i już wiem od kogo mogę żądać peeringu;)

Celem było potwierdzenie lub zaprzeczenie tezie, jakoby gros ruchu z większości sieci szło w kierunku TP S.A., przy okazji dodatkowym zadaniem było ograniczenie ruchu ftp do określonego poziomu. Całość musiała się rozstrzygnąć przy pomocy narzędzi ogólnie dostępnych w oparciu o dane zgromadzone z niedużego serwera z Linuksem, z serwerem ftp z popularną zawartością i kilkoma mniejszymi rzeczami (www, smtp, pop3).

Zabawa w kopciuszkę czyli zliczanie pakietów.

Żeby móc zacząć robić statystyki musiałem zacząć od posortowania ruchu wg kategorii, które chciałem później analizować. Interesowały mnie tak naprawdę cztery kategorie ruchu:

- całość ruchu wychodzącego z serwera
- ruch wychodzący w kierunku TP S.A.
- ruch wychodzący w kierunku GTS, w tym wypadku głównego dostawcy łącza
- ruch wychodzący ftp generowany przez użytkownika, z którego prawami działał serwer ftp z dostępem anonimowym

Po przeczesaniu freshmeat.net i googlowaniu za odpowiednimi narzędziami ostatecznie okazało się, że poczciwe iptables jest doskonałym narzędziem do zabawy w Kopciuszkę.

Wykorzystałem łańcuch mangle, który pozwalał na znakowanie pakietów wg kategorii określonych powyżej. Mangle poza możliwością znakowania ruchu jest celem nieterminującym, czyli pakiety, które zostaną wysłane do tego właśnie celu, są dalej przetwarzane przez iptables. Do samego ograniczania ruchu ftp z serwera skorzystałem z pomocy tc.

Poniżej krok po kroku przedstawiam konfigurację iptables, która pozwoliła na pogrupowanie ruchu, oznakowanie ruchu i okresowe zbieranie statystyk.

Na początek dla porządku czyszczę cały łańcuch mangle:

```
iptables -t mangle -F
```

Żeby móc coś powiedzieć o ilości danych przesyłanych w określonym kierunku potrzebowałem informacji o sumie ruchu wychodzącego z serwera. Aby oznakować cały ruch utworzyłem własny łańcuch i tam skierowałem wszystkie pakiety wychodzące. W łańcuchu tym pakiety wpadające do niego otrzymują znacznik 0x11.

Znakowanie wszystkich pakietów

```
iptables -t mangle -X all_data
iptables -t mangle -N all_data

iptables -t mangle -A OUTPUT -o $ETH -j all_data
iptables -t mangle -A all_data -j MARK --set-mark 0x11
```

Oczywiście można to zrobić w jednym łańcuchu, ale ponieważ zliczane będą jeszcze inne grupy pakietów, zależało mi na jednakowym sposobie sumowania ruchu wychodzącego.

Podobnie znakowany będzie ruch wychodzący w kierunku operatora. Warto zwrócić uwagę, że znaczniki nadane we wcześniejszych liniach będą nadpisywane przez kolejne, jeśli pakiety trafią do różnych łańcuchów. W tym przypadku będzie tak zawsze, bo każdy pakiet wychodzący w kierunku GTS jest także pakietem wychodzącym w ogóle.

Znakowanie pakietów wychodzących w kierunku GTS

```
iptables -t mangle -X gts
iptables -t mangle -N gts

iptables -t mangle -A OUTPUT -o $ETH -d 195.94.192.0/19 -j gts
iptables -t mangle -A OUTPUT -o $ETH -d 217.8.160.0/19 -j gts
iptables -t mangle -A OUTPUT -o $ETH -d 217.153.0.0/16 -j gts

iptables -t mangle -A gts -j MARK --set-mark 0x21
```

Pojawia się przy okazji pytanie skąd brać aktualne zakresy adresów IP przypisane do danego operatora. Te dane udostępniają serwery whois. Jeśli chcę uzyskać spis sieci przyznaných TP S.A. wystarczy użyć polecenia:

Wyszukiwanie prefiksów rozgłaszanych przez operatora

```
vix:~# whois -h filtergen.level3.net RIPE::AS5617

vix:~# whois -h filtergen.level3.net RIPE::AS5617
Prefix list for policy RIPE::AS5617 = RIPE::AS5617

80.48.0.0/13
83.0.0.0/11
194.204.128.0/18
...
```

Podobnie jak powyżej oznaczałem pakiety skierowane do sieci GTS, oznaczam pakiety wychodzące do sieci TP S.A. Należy tylko zmienić zakresy sieci docelowych (opcja -d) oraz znacznik, którego użyję (opcja -set-mark).

Znakowanie pakietów wychodzących w kierunku TP S.A.

```
iptables -t mangle -X tpnet
iptables -t mangle -N tpnet

iptables -t mangle -A OUTPUT -o $ETH -d 80.48.0.0/13 -j tpnet
iptables -t mangle -A OUTPUT -o $ETH -d 83.0.0.0/11 -j tpnet
iptables -t mangle -A OUTPUT -o $ETH -d 194.204.128.0/18 -j tpnet
...
```

...

```
iptables -t mangle -A tpnet -j MARK --set-mark 0x31
```

Pozostaje mi policzyć pakiety wysyłane do użytkowników mojego serwera ftp. Tę regułę zostawiam na koniec po to, aby żadne inne nie nadpisały znacznika w przypadku ściągania danych z serwera ftp przez użytkowników GTS. Gdyby znakowanie według adresów GTS następowało po znakowaniu ruchu ftp, nie mógłbym później ograniczać ruchu z serwera ftp, bo część znaczników zostałaby nadpisana.

Znakowanie wychodzących pakietów ftp

```
iptables -t mangle -X ftp
iptables -t mangle -N ftp

iptables -t mangle -A OUTPUT -o $ETH \
-m owner --uid-owner anonftp -j ftp

iptables -t mangle -A ftp -j MARK --set-mark 0x41
```

Następnie sprawdzam czy ruch jest rzeczywiście zliczany:

Pokazywanie informacji o wychodzących danych ftp

```
vix:~# iptables -t mangle -L ftp -n -v -x
Chain ftp (1 references)
pkts    bytes    target prot opt in out  source      destination      MARK set 0x41
1002842 1356515158 MARK    all  --  *  *    0.0.0.0/0    0.0.0.0/0
```

Połowa zadania jest wykonana: pakiety zostały poznakowane i są już zliczane. Ruch ftp mogą następnie w dogodny dla siebie sposób ograniczyć za pomocą tc.

Ograniczenie ruchu wychodzącego ftp

```
tc qdisc del dev $ETH root
tc qdisc add dev $ETH root handle 1: htb default 10

tc class add dev $ETH parent 1: classid 1:1 htb rate 2mbit ceil 2mbit

tc class add dev $ETH parent 1:1 classid 1:10 htb rate 2mbit prio 1
tc class add dev $ETH parent 1:1 classid 1:20 htb rate 1024kbit prio 2

tc qdisc add dev $ETH parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev $ETH parent 1:20 handle 20: sfq perturb 10

tc filter add dev $ETH protocol ip parent 1: prio 2 handle 0x41 \
fw flowid 1:20
```

Wykresy.

Teraz przydałoby się pokazać zebrane dane w jakiejś rozsądnej postaci. Zdecydowałem się na rrd, które jest następcą zasłużonego i wysłużonego mrtg. rrd daje znacznie większe możliwości jeśli chodzi o manipulację danymi gromadzonymi w bazach .rrd jak i o prezentację wyników w formie wykresów.

Na początku trzeba stworzyć plik .rrd, w którym będą zbierane informacje:

Tworzenie pliku w którym będą zbierane statystyki RRD

```
rrdtool create stats.rrd -s 60 \
DS:all_data:COUNTER:600:0:U \
DS:gts:COUNTER:600:0:U \
DS:tpnet:COUNTER:600:0:U \
DS:ftp:COUNTER:600:0:U \
RRA:AVERAGE:0.5:1:10080 \
```

```
RRA:AVERAGE:0.5:2:5040 \
RRA:AVERAGE:0.5:4:2520 \
RRA:AVERAGE:0.5:6:1680 \
RRA:AVERAGE:0.5:180:56 \
RRA:AVERAGE:0.5:360:28
```

Polecenie to tworzy plik stats.rrd inicjując kolejno cztery źródła danych - DS. Każde ze źródeł będzie przechowywać informacje o określonym typie ruchu. Minimalna wartość została ustalona na 0, maksymalna jest niezdefiniowana U. Dane będą pobierane co 60 sekund, a przechowywane próbki określone są przez rekordy RRA.

```
RRA:AVERAGE:0.5:1:10080
```

Średnia z jednej próbki, czyli z 60 sekund; przechowywanych będzie 10080 próbek, co daje ostatecznie:

$10080 \text{ min} / 60 \text{ minut} / 24 \text{ godziny} = 7 \text{ dni.}$

```
RRA:AVERAGE:0.5:180:56
```

Średnia ze 180 próbek, czyli z trzech godzin; przechowywanych będzie 56 próbek. Ostatecznie daje to:

$3 \text{ godziny} * 56 \text{ próbek} = 174 \text{ godziny} = 7 \text{ dni.}$

Po szczegółowy opis parametrów rrd odsyłam do stron rrd, może warto tylko dodać, że jako typ danych został wybrany COUNTER, ponieważ pozwala on zliczać stale rosnące dane i dobrze radzi sobie z przewijającymi się licznikami.

Następnie będę wypełniać plik stats.rrd danymi sczytywanymi w odstępach jednonminutowych. Do tego wykorzystam prosty skrypcik uruchamiany poprzez cron w następujący sposób:

```
* * * * * /home/mazek/bin/update.sh
```

Sam skrypt ma za zadanie odczytać stan liczników dla podanych łańcuchów oraz wprowadzić te dane do bazy rrd.

Skrypt do aktualizowania danych w pliku rrd

```
#!/bin/bash
ITM="/sbin/iptables -t mangle"
# Pobieramy liczbę bajtów z podanego łańcucha.
ALL_DATA=`$ITM -L all_data -n -x -v | tail -n 1 | awk -F " " {'print $2;}`
GTS=`$ITM -L gts -n -x -v | tail -n 1 | awk -F " " {'print $2; }`
TPNET=`$ITM -L tpnet -n -x -v | tail -n 1 | awk -F " " {'print $2; }`
FTP=`$ITM -L ftp -n -x -v | tail -n 1 | awk -F " " {'print $2; }`
# Wprowadzamy dane pobrane do pliku .rrd
rrdtool update stats.rrd N:$ALL_DATA:$GTS:$TPNET:$FTP
```

I tu zaczyna się najlepsze, czyli narysowanie tego co zgromadziłem. rrd daje pod tym względem ogromne możliwości w porównaniu do swojego poprzednika - mrtg. Możliwości prezentacji danych przez rrd w różnoraki sposób są naprawdę imponujące i dla kogoś, kto

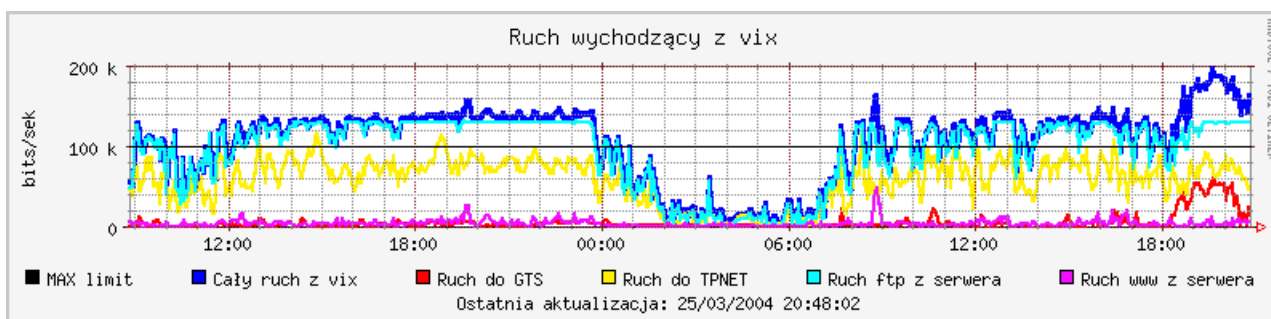
wcześniej korzystał z mrtg - wręcz przytłaczające;)

Na początek zgromadzę wszystkie dane na jednym wykresie:

Generowanie wykresu

```
COMMENT="Ostatnia aktualizacja: "` date +"%d/%m/%Y %T"``
rrdtool graph stats1.gif --title="Ruch wychodzący z vix" \
-w 700 --start -129600 \
  DEF:all_data=stats.rrd:all_data:AVERAGE \
  DEF:gts= stats.rrd:gts:AVERAGE \
  DEF:tpnet= stats.rrd:tpnet:AVERAGE \
  DEF:ftp= stats.rrd:ftp:AVERAGE \
  DEF:www= stats.rrd:www:AVERAGE \
  HRULE:100000#000000:"MAX limit" \
  LINE3:all_data#0000ff:"Cały ruch z vix"\
  LINE2:gts#ff0000:"Ruch do GTS"\
  LINE2:tpnet#fff000:"Ruch do TPNET"\
  LINE2:ftp#00ffff:"Ruch ftp z serwera"\
  LINE2:www#888282:"Ruch www z serwera"\
  COMMENT:"" \
  COMMENT:"$COMMENT" \
  --vertical-label "bits/sek"
```

To polecenie wykonuję za każdym razem, kiedy chcę obejrzeć najnowsze dane z pliku stats.rrd. Można je także wrzucić do crona i wykonywać w wybranych okresach czasu. Wykres wygenerowany za pomocą tego polecenia będzie prezentował dane z ostatnich trzydziestu sześciu godzin (129600 sekund). Parametr DEF określa źródło danych i sposób ich interpretowania. Mnie interesuje wartość średnia z przedziału czasowego, czyli AVERAGE. Ponieważ chcę pokazać całość na jednym wykresie, definiuję cztery źródła danych. Linie wykreślającą całość ruchu z serwera rysuję najgrubszą linią - LINE3, kolorem czerwonym - wartość rgb: ff0000, pozostałe będą kreślone linią średniej grubości - LINE2 w różnych kolorach. Na dole wygenerowanego obrazka dodana zostanie legenda łącząca kolory z określonym typem ruchu pokazywanego, oraz komentarz.



Rysunek 1: Wykres pokazujący wszystkie typy zliczanego ruchu

Rysunek 1 pokazuje wszystkie typy ruchu monitorowanego na jednym wykresie. Każdy z rodzaj ruchu pokazywany jest za pomocą linii. Użyteczność tego typu wykresów jest więc ograniczona.

Żeby nie mieszać rzeczy niepowiązanych ze sobą, czyli statystyk typu ruchu (ftp, www) i celu danych (TP S.A., GTS), rozbijam poprzedni wykres na dwa oraz próbuję sprawić żeby stał się bardziej czytelny.

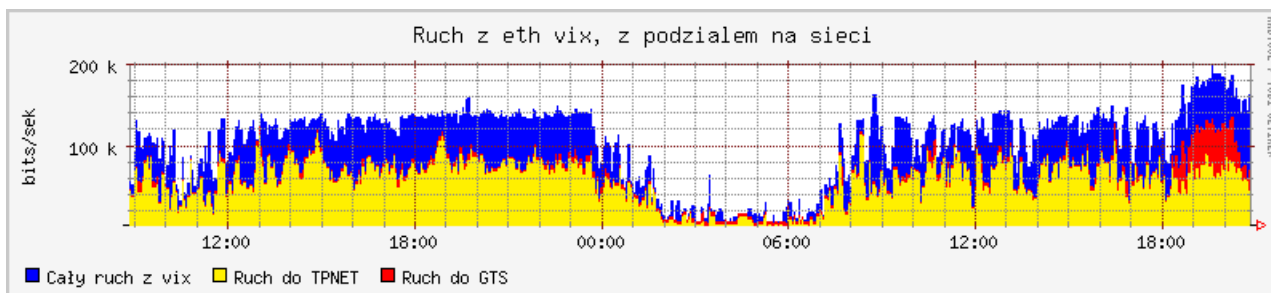
Generowanie wykresu

```
rrdtool graph stats2.gif --title="Ruch z eth vix, z podziałem na sieci" \
-w 700 --start -129600 \
  DEF:all_data= stats.rrd:all_data:AVERAGE \
  DEF:gts= stats.rrd:gts:AVERAGE \
```

```

DEF:tpnet= stats.rrd:tpnet:AVERAGE \
AREA:all_data#0000ff:"Cały ruch z vix"\
AREA:tpnet#fff000:"Ruch do TPNET"\
STACK:gts#ff0000:"Ruch do GTS"\
--vertical-label "bits/sek"

```



Rysunek 2: Ruch pogrupowany wg celu danych

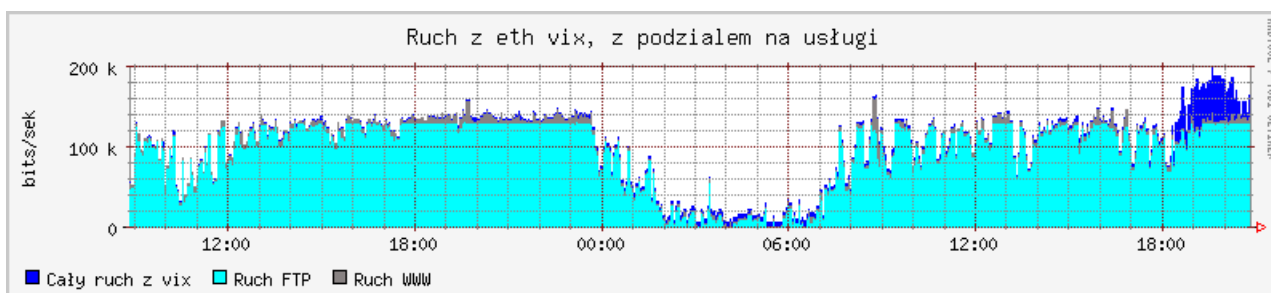
Rysunek 2 pokazuje zestakowane dane o ruchu wychodzącym wg. celu pakietów wychodzących.

Generowanie wykresu

```

rrdtool graph stats3.gif --title="Ruch z eth vix, z podziałem na usługi" \
-w 700 --start -129600 \
DEF:all_data= stats.rrd:all_data:AVERAGE \
DEF:ftp= stats.rrd:ftp:AVERAGE \
DEF:www= stats.rrd:www:AVERAGE \
AREA:all_data#0000ff:"Cały ruch z vix"\
AREA:ftp#00ffff:"Ruch FTP"\
STACK:www#888282:"Ruch WWW"\
--vertical-label "bits/sek"

```



Rysunek 3: Ruch pogrupowany wg typu danych.

Rysunek 3 pokazuje zestakowane dane o ruchu wychodzącym wg. usług. W tym przypadku badamy jedynie ftp i www.

W obydwu przypadkach utworzone zostały wykresy pokazujące dane z ostatnich sześciu godzin. Źródła danych (DEF) zostały ograniczone tylko do tych, które mnie interesują. Co najważniejsze, zmienił się także sposób prezentacji. Zamiast linii wykreślających poziom ruchu danego typu, pokazuję wypełnione obszary. W obydwu przypadkach całość ruchu jest pokazywana kolorem granatowym. Na Rys. 2. na obszarze granatowym pokazywane są dane o ruchu do sieci GTS i TP S.A. Ruch do sieci TP S.A. pokazany jest za pomocą koloru żółtego, do sieci GTS za pomocą koloru czerwonego. Jak widać, można spokojnie rozpoczynać negocjacje z TP S.A. w kwestii darmowej wymiany ruchu;)

Co ważne, ruch do różnych sieci pokazywany jest w formie zestakowanych a nie nakładających się obrazów dzięki użyciu słowa kluczowego STACK. Podobnie jest na rysunku 3., który pokazuje różne typy ruchu. Granatowy kolor oznacza całość ruchu wychodzącego, jasnoniebieski obszar to wychodzący ruch ftp a szary obszar to ruch www.

Jak widać, za pomocą prostych i powszechnie dostępnych narzędzi można uzyskać bardzo ciekawe informacje. Oczywiście rozwiązania te nie zastąpią zintegrowanych narzędzi do analizy ruchu, ale w dużym stopniu mogą ułatwić zarządzanie siecią i dostępnym pasmem.

Informacje:

[1] Strona domowa projektu RRD: <http://www.rrdtool.com/>

[2] Strona domowa projektu NetFilter: <http://www.netfilter.org/>

Autor:

Marcin Mazurek <M.Mazurek@netsync.pl>, <http://mazek.netysync.pl/>