

Подсчет и анализ сетевого трафика

Автор: Виталий Солдатов, инженер филиала [ООО "РН-Информ" в г. Красноярск](#)

Статья опубликована в рамках конкурса "Формула связи"

В данной статье рассматривается сбор данных с маршрутизаторов Cisco для тарификации его в автоматизированной системе расчетов, даются советы по исключению паразитного трафика при «помегабайтном» учете, так же проводится разбор и анализ трафика средствами flow-tools и rrdtools.

С маршрутизаторов Cisco снять данные можно несколькими способами, например с помощью netflow или snmp-протокола. Нельзя сказать, что каждый из них настолько универсален, что другим можно пренебречь. Более того, в составе пакета netflow включены данные об интерфейсах в виде чисел, взятых из базы данных управления snmp. Вот так в комплексе и предполагается использовать netflow и snmp.

Как правило, каждая автоматизированная система расчетов (АСР) содержит в себе коллектор netflow, который представляет собой программное обеспечение, принимающее udp-пакеты на определенном порту. В самой АСР имеется возможность указания источников netflow — маршрутизаторов, в противном случае пакеты от неизвестного источника будут отброшены. Если настройка АСР согласно документации вопросов может не вызвать, то к «включению» netflow на маршрутизаторе необходимо чуть более вдумчиво. Для начала необходимо сформулировать условия тарификации. Наиболее часто встречающиеся сейчас — дифференцированные по зонам, например, «интернет», «город», «бесплатная локальная сеть». Исходя из тарифов с зонами «вытекает» методика сбора. Так, например, если тарифицируется только интернет, а все остальные услуги входят в абонплату, то логичнее организовать сбор netflow только на маршрутизаторе, который имеет непосредственное присоединение к оператору связи, продавшему вам полосу в интернет. Можно, конечно, собирать netflow со всех маршрутизаторов и хранить тарифицируемые за 0 руб. данные на приобретенной специально по такому случаю X терабайтной корзине 3 года согласно ЗоС. Ведь если у вас такая информация собирается и хранится, то получается, что вы обязаны ее предоставлять. А на нет — спроса нет. Сделаю небольшое отступление «как сдавать абонента».

Дешевле всего сдать абонента по логам radius-сессий, если доступ в интернет коммутируемый (PPPoE, PPTP). В стартовом аккаунтинговом пакете хранится информация о логине пользователя, ip-адресе, выданном при подключении. Если время и адрес источника совпадает — получите, распишитесь. Дороже, если приходится собирать информацию о том, «а кто из ваших абонентов в период с 01.07.09 по 01.10.09 посещал блог [ekstremizm.livejournal.com](#) и ..(список на трех листах прилагается)?» Еще дороже получается, если подобные запросы приходят, а вы собираете flows со всей вашей сети.

В общем, задумываясь о тарификации по 1 коп за гигабайт какого-нибудь аниме-трафика следует вспомнить о шкафах с терабайтными корзинами, где лежит «сырая» статистика, а не фильмы и все еще раз взвесить. Возможно, что от жадности вам придет озарение хранить не всю «сырую» статистику или внезапно вы введете просто абонплату за пользование местными ресурсами.

Итак, на маршрутизаторе включается netflow следующим образом:

```
ip flow-export version 5
ip flow-export destination 10.0.0.1 9996
!ip flow-export source Loopback0
```

Здесь 10.0.0.1 9996 адрес и порт, на котором «слушает» коллектор АСР. Строка «ip flow-export source» как бы говорит от какого адреса будет слать маршрутизатор данные коллектору. Необходимо в том случае, если маршрутизатор имеет несколько подключений к сети и маршрут с адресом коллектора получает динамически. Либо вы предоставляете услугу в MPLS-сети на PE-маршрутизаторе, где интернет раздается абонентам через vrf INTERNET какой-нибудь и АСР не имеет доступа к бэкбону, а только к Lo0, находящемуся в этом же vrf INTERNET.

Очевидно, что необходимо включить сбор данных на вашем «граничном» маршрутизаторе, которым вы подключились к оператору связи. Здесь можно понаблюдать эволюцию Cisco. Ранее декларировалось так: включайте **ip route-cache flow** на «железных» интерфейсах (и надо заметить что другой команды включения подсчета трафика в старых IOS не было). Позже появилось: для подинтерфейсов получите команду **ip flow ingress**. Теперь (IOS 12.4T) имеем при включении **ip route-cache flow** на «железных» интерфейсах команду ip flow ingress на всех подинтерфейсах этого интерфейса, а самой команды **ip route-cache flow** больше не видеть :

```
interface GigabitEthernet0/0
description to switch
no ip address
ip flow ingress
```

```
duplex auto
speed auto
media-type rj45
!
interface GigabitEthernet0/0.101
description ACP
encapsulation dot1Q 101
ip address 10.0.0.254 255.255.255.0
ip flow ingress
!
interface GigabitEthernet0/0.102
encapsulation dot1Q 102
ip address 10.10.1.1 255.255.255.0
ip flow ingress
```

Это удобно лишь отчасти, так как в новые подинтерфейсы **ip flow ingress** придется добавлять вручную или повторять команду на физическом Gi0/0. Проверить сбор данных можно командами **show ip flow export** и **show ip cache flow**. В первом случае будет показано количество пакетов, переданных коллектору, а во втором — кэш, в котором происходит накопление данных прежде чем окончательно сформировать пакет для отправки коллектору. Этот кэш имеет собственные параметры и я редко встречал, чтобы их кто-то менял или вообще как-то использовал.

```
router(config)#ip flow-cache timeout active ?
<1-60> Timeout in minutes
```

```
router(config)#ip flow-cache timeout inactive ?
<10-600> Timeout in seconds
```

Что это значит? Активной длительной сессией, как правило, является tcp-сессия. Данные об этой сессии будут накапливаться на маршрутизаторе по-умолчанию в течение 30 минут и только потом пакет flow будет сформирован и отправлен коллектору. Чем это грозит? При выключении маршрутизатора данные по активным сессиям до 30 минут будут потеряны. Данные могут быть переданы коллектору в 00:01 следующих суток, а начался новый месяц. Могут быть переданы в 08:01, а у вас ночной трафик до 08:00 по льготным ценам. Если коллектор не анализирует метки времени в пакете (start,end timestamps), то у вас большие проблемы. Уменьшение времени кэширования позволяет существенно сгладить этот негативный момент за счет увеличения нагрузки на процессор, увеличения количества flows, увеличения занимаемого места на вашей терабайтой корзине. Из плюсов — наконец-то получился адекватный пятиминутный мониторинг трафика с ровными графиками, а не пиками раз в полчаса.

Запомнить: **ip flow-cache timeout active 5**

В заключение по кэшу хочу заявить, что все попытки испытать по метрологической теме netflow и кэша вызывают у меня улыбку. Как ЭТО сертифицировать?

Для создания зон/классов трафика необходимо включить snmp на маршрутизаторе, предварительно создав ACL для ограничения доступа.

```
access-list 15 permit 10.0.0.1
snmp-server community public RO 15
snmp-server ifindex persist
```

Без этого в flows в качестве индекса интерфейсов будут всегда нули. ifindex persist необходим в случаях, когда ACP не оперирует названиями интерфейсов, а только индексами. «UTM», например. Есть ACP, которые регулярно обращаются к маршрутизатору и делают сопоставления индексов именам интерфейсов и уже в виде имен начинают обработку по правилам. «Старт-IP», например. Если взять коллектор flow-tools, который просто коллектор, а не ACP, то таких фиш у него нет по понятным причинам. Коллектор должен только принимать пакеты и очень быстро их складывать на жесткий диск не загружая процессор своей работой.

Вот и подошли как-то сбоку к паразитному трафику. Очевидно, что если пакет был получен маршрутизатором в интерфейс SrcIf, имеющий какой-то индекс, отличный от нуля и передан в интерфейс DstIf, имеющий какой-то индекс, отличный от нуля, то пакет считается переданным. А если интерфейс с адресом DstIf был выключен/не было маршрута, то трафик дропнется в null и соответственно в DstIf будет ноль. Так будет, если вы предварительно прописали все ваши сети в null с большой административной дистанцией, например:

```
ip route 10.0.0.0 255.0.0.0 Null0 254
```

DstIf будет так же ноль, если трафик дропнулся на ACL. Например, абонент заплатил за

персональный firewall. Не учитывая правила формирования индексов для DstIf, при работе с ACL можно выставить абоненту весь не полученный им флуд.

Какие еще гадости могут быть при подсчете трафика? Вы выдали абоненту дополнительную сеть из 256 адресов и смаршрутизировали на его интерфейс. Абонент разбил ее на кучу мелких сетей и пока не использует целиком. В случае, если ваш абонент не сделал у себя тоже самое что и вы с вашим блоком адресов, а именно «ip route 10.10.10.0 255.255.255.0 Null0 254», то он получит 30кратное увеличение паразитного трафика за счет правильного, по-вашему мнению, сбора данных. **Ip flow ingress** будет обрабатывать на возвращаемый от абонента пакет пока не кончится TTL. Правильным будет объяснить абоненту, что маршрутизацией необходимо заниматься и еще можно повесить ACL такому абоненту.

```
ip access-list extended client.in
deny ip 10.10.10.0 0.0.0.255 10.10.10.0 0.0.0.255
permit ip any any
```

В АСР «Старт-IP» есть механизм, который позволяет формировать динамические правила для исключения подобной ситуации. Что-то вроде «если SrcIf и DstIf идентичны, то не тарифицировать». Понятно, что в «UTM» я такого не находил.

В итоге получились следующие признаки паразитного трафика:

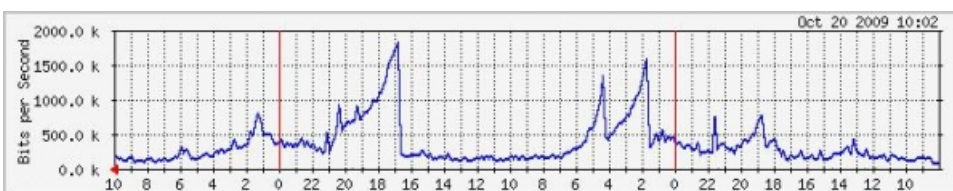
- пакет пришел на несуществующий маршрут и дропнулся в null. DstIf=0
- пакет дропнулся на ACL. DstIf=0
- пакет заиклился из-за ошибок маршрутизации. SrcIf=DstIf

Казалось бы, что список правил для первой нетарифицируемой зоны TRASH готов. Для нормальной АСР - да, но не для «UTM». В этой простой программе нельзя указать индекс «0». Для программы это тоже самое, что параметр с интерфейсом вообще не используется, т.е. не TRUE вовсе. Пришлось поломать немного голову и создать дополнительный подинтерфейс, у которого, конечно же, индекс отличен от нуля, куда все сети были смаршрутизированы.

```
interface GigabitEthernet0/0.254
encapsulation dot1Q 254
ip address 10.255.255.253 255.255.255.252
ip flow ingress
ip route 10.0.0.0 255.0.0.0 10.255.255.254 254
```

А на обратной стороне с адресом 10.255.255.254 эти же сети отправляются в null0. Кстати, посмотреть индексы на маршрутизаторе можно командой **show snmp mib ifmib ifindex**.

Решение дало интересный результат. Стало возможным увидеть количество паразитного трафика, проходящего через этот интерфейс. Предлагаю вашему вниманию «шум» на сеть из 512 адресов.

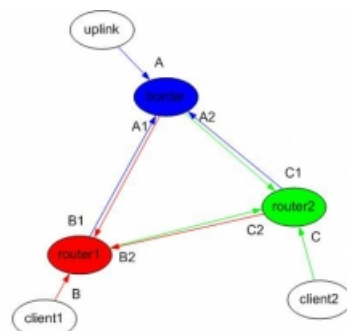


Что еще следует поместить в эту зону TRASH? Возможно, что некоторые из вас сталкивались с тем, что оператор складывал n раз трафик в биллинг и выставлял вам m*n денег. Рассказываю как это делается.

Существует куча из двух известных мне способов сбора трафика в neMPLS-сетях.

Первый из них — создание контура трафика. Сбор (почему-то русское слово использовать привычнее, чем нетфлоу аккаунтинг) включается на всех интерфейсах всех ваших маршрутизаторов за исключением тех, которыми эти ваши маршрутизаторы соединяются друг с другом. Соединения с маршрутизаторами других операторов в исключения не попадают. Исключаются только внутренние связи. Никакого дублирования нет в принципе, кроме случаев, когда на одной внутренней связи сбор все-таки включили. Такое может быть, если дали команду на физическом интерфейсе невзирая на подинтерфейс-аплинк. Все клиенты такого маршрутизатора повторно получают статистику через еще один **ip flow ingress** на пути прохождения пакета. Ловкость рук и никакого мошенничества с сертифицированной АСР, однако.

Во втором случае сбор трафика включается на всех интерфейсах всех маршрутизаторах. Внутренние связи исключаются путем внесения правил с интерфейсами в зону TRASH. Здесь должно иметь место слаженная работа оператора биллинга и сетевого инженера, который настраивает маршрутизаторы.



```
Router#border DIF=A1 Class=trash
Router#border DIF=A2 Class=trash
Router#router1 DIF=B1 Class=trash
Router#router1 DIF=B2 Class=trash
Router#router2 DIF=C1 Class=trash
Router#router2 DIF=C2 Class=trash
Router#* DIF=0 Class=trash
Router# DIF=S0 Class=trash
Router#* DIF=* Class=internal
```

Красным выделено правило для исключения «петель». Как правило, не реализуется через графический интерфейс АСР

По зоне TRASH вроде бы все и теперь описывается зона/класс «локальная сеть». В биллинг вносятся все собственные адреса. Порядок зон/классов должен строго соблюдаться. Вероятнее всего, АСР будет обрабатывать правила до первого совпадения. Первым идет TRASH, так как он может содержать исключающие внутренние связи, далее идут «локальная сеть». Если вы хотите выделить какой-то кусок из «локальной сети» и тарифицировать его отдельно, то логичнее поставить правило с куском выше «локальной сети».

- будете ли вы пропускать трафик по цене ~0 руб. для своих клиентов до выяснения ситуации лишь бы связность не нарушить?
- будете ли невзирая блокировать трафик бывшего пира на порту?
- победит жадность или разум при подключении к городским точкам обмена?

Заключительная зона/класс — это * (все что осталось). Тарифицируется как интернет трафик. На этом с ACP можно закончить и перейти к анализу трафика.

- **flow-tools** — коллектор netflow и набор программ для фильтрации, подготовки отчетов;
- **rrdtools** — программа по созданию баз данных, рисованию графиков.

```
ip flow-export destination 10.0.0.2 20000
```

Параметры запуска коллектора зависят от цели использования. Пусть первая цель будет обыденная — сбор и подсчет трафика для контроля работы АСР.

```
/usr/local/bin/flow-capture -w /data/flows 0.0.0.0/0.0.0.0/20000
```

При таком запуске в /data/flows будут автоматически создаваться каталоги 2009/2009-10/2009-10-20/ , где каждые 15 минут будут автоматически создаваться файлы вида ft-v05.2009-10-20.001500+0800 . Файлы лучше обрабатывать поочередно. Так меньше расходуется оперативная память и в итоге получается быстрее. Ниже в качестве примера приводится скрипт маленькой программы статистики (адреса изменены).

```
hostname# ls
filter.cfg report.cfg stats.sh tmp
```

```
hostname# cat filter.cfg
filter-primitive lan
type ip-address-prefix
permit 999.267.070.0/30
```

```
filter-primitive service
type ip-address-prefix
permit 999.267.070.28/30
```

```
filter-definition lan
match ip-destination-address lan
```

```
filter-definition service
match ip-destination-address service
```

```
hostname# cat report.cfg
include-filter /data/stat/filter.cfg
```

```
stat-report lan
type summary-counters
filter lan
output
options -header,-xheader,+totals
path /data/stat/tmp/lan
```

```
stat-report service
type summary-counters
filter service
output
options -header,-xheader,+totals
path /data/stat/tmp/service
```

```
stat-definition traf
report lan
report service
```

```
hostname# cat stats.sh
#!/usr/local/bin/bash
```

```
WORDIR='/data/stat'
PERIOD='/data/flows/2009/2009-09'
FILELIST=`find ${PERIOD} -name "ft*"`
HOSTLIST='lan service'
for h in ${HOSTLIST} ; do rm -f ${WORDIR}/tmp/${h}.sum ; done
for i in ${FILELIST} ; do
flow-report -s ${WORDIR}/report.cfg -S traf < ${i}
for h in ${HOSTLIST} ; do cat ${WORDIR}/tmp/${h} >> ${WORDIR}/tmp/${h}.sum ; done
done

echo "Traf stat for ${PERIOD}" > ${WORDIR}/tmp/final
for h in ${HOSTLIST} ; do
echo -n "${h} = " >> ${WORDIR}/tmp/final
grep -v ^# ${WORDIR}/tmp/${h}.sum | awk -F, '{s=s+$4}; END {print s}' >> ${WORDIR}/tmp/final
done
```

```
cat ${WORDIR}/tmp/final
```

```
hostname# ./stats.sh
```

Traf stat for /data/flows/2009/2009-09
lan = 220948540963
service = 243234438

Стоит отметить, что awk при работе с очень большими числами начинает их приводить к виду 1e+18. Решение есть.

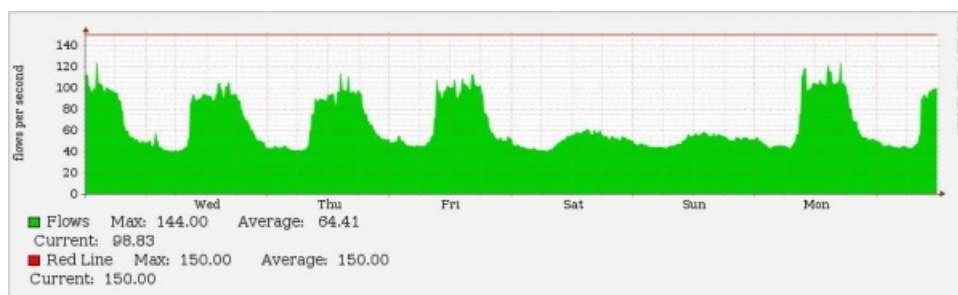
Создание разнообразных фильтров даст вам необходимую информацию. Например, объем передаваемых данных между вами и потенциальными пиринговыми партнерами.

Дальше про мониторинг все-таки. Для мониторинга нет необходимости хранить множество файлов и поэтому запускается коллектор со следующими параметрами:

```
/usr/local/bin/flow-capture -V 5 -e 100 -N 0 -n 287 -S 5 -w /usr/local/monitor/flow/router1 0/0/20000
```

Всего будет храниться 100 файлов, каталоги создаваться не будут, файлы создаются с интервалом в 5 минут.

Мониторинг вирусной активности. Вирусы характеризуются многочисленным сканированием достаточно большого количества адресов, а, как известно, маршрутизатор на каждое такое соединение генерирует flows и отправляет коллектору. В нормальном рабочем режиме этот поток данных достаточно предсказуем. Обратите внимание на недельный график:



Чтобы получить такой график необходимо обработать последний файл в коллекторе, при отсутствии базы данных rrd создать ее и занести туда данные, нарисовать график.

Скрипт ниже, запускаемый периодически, выполняет обработку файла и заносит данные в базу.

```
[root@mg /usr/local/monitor/bin]# cat flow-virus.sh  
#!/usr/local/bin/bash
```

```
RRDDIR='/usr/local/monitor/rrd'  
BINDIR='/usr/local/monitor/bin'
```

```
function virus {
```

```
  NODE=$1  
  REDLINE=$2
```

```
  FILE=`ls /usr/local/monitor/flow/${NODE}/ft* | tail -1`  
  FLOWDIR="/usr/local/monitor/flow/${NODE}"  
  TIMEST=`date -v0S +%s`
```

```
  FLOWS=`flow-cat ${FILE} | flow-stat | grep "Average flows / second (real)" | sed s~.*:.*\ ~ ~ | sed s~  
  [.*].*~~`
```

```
  [ -f ${RRDDIR}/${NODE}_virus.rrd ] || ${BINDIR}/create_v.sh ${NODE}_virus  
  [ -f ${RRDDIR}/${NODE}_virus.rrd ] && rrdtool update ${RRDDIR}/${NODE}_virus.rrd --template  
  redline:flows ${TIMEST}:${REDLINE}:${FLOWS}
```

```
  if [ ${REDLINE} -le ${FLOWS} ] ; then
```

```
    VIRUSDIR="/usr/local/www/data/monitor/virus/${NODE}"
```

```
    OUTFILE=`date +%y-%m-%d-%H-%M`
```

```
    [ -d ${VIRUSDIR} ] || mkdir -p ${VIRUSDIR}
```

```
    flow-cat ${FILE} | flow-stat -f9 -S1 | grep -v ^# | head -n 50 > ${VIRUSDIR}/${OUTFILE}.txt
```

```
    #head -n 5 ${VIRUSDIR}/${OUTFILE}.txt | awk '{print $1"\t"$2}' | mail -s "Abnormal network activity in  
    ${NODE}" mail@mail.it
```

```
  fi
```

```
}
```

virus router1 150

При превышении параметра, выставляемого вручную, происходит выборка по максимальному количеству пакетов от источников. Можно так же список отправить почтой. База создается один раз.

```
[root@mg /usr/local/monitor/bin]# cat create_v.sh
#!/bin/sh
```

```
RRD="/usr/local/monitor/rrd/$1.rrd"
CREATETIME=`date -v-5M +%s`
rrdtool create ${RRD} --start ${CREATETIME} --step 300 \
DS:flows:GAUGE:600:0:1000000000 \
DS:redline:GAUGE:600:0:1000000000 \
RRA:AVERAGE:0.5:1:600 \
RRA:AVERAGE:0.5:6:700 \
RRA:AVERAGE:0.5:24:775 \
RRA:AVERAGE:0.5:1440:3985 \
RRA:MIN:0.5:1:600 \
RRA:MIN:0.5:6:700 \
RRA:MIN:0.5:24:775 \
RRA:MIN:0.5:1440:3985 \
RRA:MAX:0.5:1:600 \
RRA:MAX:0.5:6:700 \
RRA:MAX:0.5:24:775 \
RRA:MAX:0.5:1440:3985 \
RRA:LAST:0.5:1:600 \
RRA:LAST:0.5:6:700 \
RRA:LAST:0.5:24:775 \
RRA:LAST:0.5:1440:3985
```

Ну и та часть исполняемого файла, которая отвечает за рисование.

```
function virus_t {

IMG=${IMGDIR}/${1}
[ -f ${RRDDIR}/${2} ] && RRD=${RRDDIR}/${2}

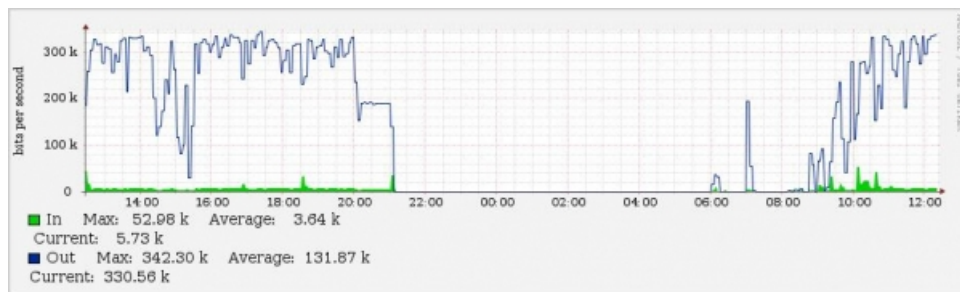
/usr/local/bin/rrdtool graph ${IMG} \
--imgformat=PNG \
--start=-${RRDSTART} \
--end=-300 \
--rigid \
--base=1000 \
--height=75 \
--width=400 \
--alt-autoscale-max \
--lower-limit=0 \
--vertical-label="flows per second" \
--slope-mode \
--font AXIS:8: \
--font LEGEND:10: \
--font UNIT:8: \
DEF:a=${RRD}:flows:AVERAGE \
DEF:b=${RRD}:flows:MAX \
DEF:c=${RRD}:redline:AVERAGE \
DEF:d=${RRD}:redline:MAX \
AREA:a#00CF00FF:"Flows" \
GPRINT:b:MAX:"Max\:%8.2lf %s" \
GPRINT:a:AVERAGE:"Average\:%8.2lf %s\n" \
GPRINT:a:LAST:" Current\:%8.2lf %s\n" \
LINE1:c#FF0000FF:"Red Line" \
GPRINT:d:MAX:"Max\:%8.2lf %s" \
GPRINT:c:AVERAGE:"Average\:%8.2lf %s\n" \
GPRINT:c:LAST:"Current\:%8.2lf %s"

}
```

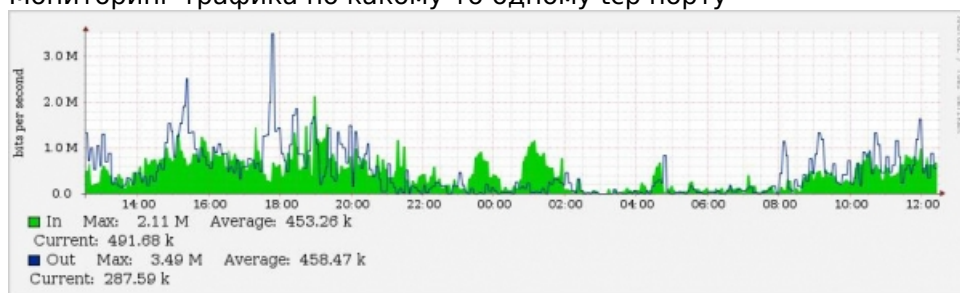
```
# virus activity
virus_t router1_virus_t_${RRDSTART}.png router1_virus.rrd
virus router1_virus_${RRDSTART}.png router1_virus.rrd
```

Чтобы окончательно не перегружать статью скриптами и примерами далее приведу графики, показывающие трафик различных приложений. Это удобно для настроек правил QoS, выяснения причин неполадок, да и вообще здорово, когда подобная информация перед глазами. Информация так же собирается в базы после обработки программами flow-report и т.п.

Репликация важного приложения по спутниковому каналу. Фильтр по адресам серверов.



Мониторинг трафика по какому-то одному tcp порту



Таким образом, обезличенный трафик, который привыкли видеть все в программах мониторинга можно разложить на составляющие. Я уже не говорю о том, что в случае чего можно проанализировав трафик принять адекватные меры для ограничения какого-нибудь потока данных или его полной блокировки. Например, обработка файла с сортировкой по количеству пакетов:

```
flow-cat ft-v05.2009-10-20.120500+0800 | flow-stat -f10 -S4 | head -n 20
```

А еще надо вовремя останавливаться. Спасибо за внимание!