ШейперДляБольшихСетей

Материал из ALT Linux Wiki

Содержание

Шейпер для больших сетей (IPv4/IPv6)

Особенности работы с большими сетями

В случае, если необходимо назначать индивидуальную скорость для сотен и тысяч адресов в сети, встроенный в etcnet инструментарий eqos становится мало пригодным. По той причине, что для отработки одного правила, в shell-скриптах производится десятки вызовов внешних программ и других скриптов. В результате останов и запуск сайтов с настройками для 3 тыс. классов может растянуться до получаса. В eqos из etcnet фильтры tc создаются линейного типа. В этом случае пакет будет обходить все правила фильтрации, пока не попадет на подходящий, либо не будет отправлен в фильтр по-умолчанию. При количестве правил фильтрации в несколько тысяч, и трафике в десятки тысяч пакетов в секунду, растет загрузка CPU системы и параллельно растут задержки прохождения транзитных пакетов. Это, в свою очередь, отрицательно влияет на комфортность работы с сетью. Аналогичные последствия и при злоупотреблении линейными правилами фильтрации в iptables/ip6tables.

Поэтому не используем eqos, сводим до минимума количество правил в iptables/ip6tables и в tc отказываемся от линейных фильтров.

В случае сети, в которой используется только IPv4, фильтры tc можно построить на основе хэш-таблиц

Предварительные требования к системе

В системе должен быть установлены пакеты

- iproute2-3.8.0-alt1 и новее;
- ipset-6.23-alt1 и новее;
- kernel-modules-ipset-*-6.24-alt1 и новее;
- iptables-1.4.21-alt1.M70P.1 и новее;

Создание дисциплин, классов и очередей

Так как управление трафиком для сотен и тысяч адресов в ручном режиме никто в здравом уме делать не будет, все приведенные здесь конфигурационные файлы есть результат выполнения сторонних программ и скриптов, если это особо не указано. Все параметры для tc и ipset загружаются в batch-режиме.

Итак, в качестве корневой дисциплины была выбрана hfsc, как более экономная к процессорному времени, и ее реализация в ядре linux не имеет global-locks, в отличии от htb, в результате чего нагрузка эффективно распределяется по всем процессорным ядрам в системе.

Пример batch-файла для tc:

```
/tmp/shape-tc.hfsc:
# bondEXT - интерфейс в сторону мира
# bondINT - интерфейс в сторону клиентов
#
# Чистим корневую дисциплину от всех предыдущих настроек
qdisc del dev bondEXT root
qdisc del dev bondINT root

# назначаем корневую дисциплину, и определяем класс по-умолчанию для трафика, не попавшего ни в один опр
qdisc add dev bondEXT root handle 1: est 1sec 8sec hfsc default ffff
qdisc add dev bondINT root handle 1: est 1sec 8sec hfsc default ffff

# создаем корневой класс и определяем его параметры
class add dev bondEXT parent 1: classid 1:1 est 1sec 8sec hfsc sc rate 2Gbit ul rate 2Gbit
class add dev bondINT parent 1: classid 1:1 est 1sec 8sec hfsc sc rate 2Gbit ul rate 2Gbit

# создаем клдасс по-умолчанию для трафика, не попавшего в определенный класс
# !!!! ВАЖНО!!!! этот класс, как и фильтр по-умолчанию должен быть создан.
filter add dev bondEXT parent 1: protocol ip prio 1000 u32 match u32 0 0 classid 1:ffff
filter add dev bondINT parent 1: protocol ip prio 1000 u32 match u32 0 0 classid 1:ffff
class add dev bondEXT parent 1: classid 1:ffff est 1sec 8sec hfsc sc umax 1500 dmax 150ms rate 1000kbit
class add dev bondINT parent 1: classid 1:ffff est 1sec 8sec hfsc sc umax 1500 dmax 150ms rate 1000kbit
qdisc add dev bondEXT parent 1:ffff handle ffff: pfifo limit 50000
qdisc add dev bondINT parent 1:ffff handle ffff: pfifo limit 50000
```

Пример правил для непосредственно управляемых адресов:

```
class add dev bondINT parent 1: classid 1:b est 1sec 8sec hfsc sc umax 1500b dmax 10ms rate 92160kbit ul
```

```
qdisc add dev bondINT parent 1:b handle b: pfifo limit 200
class add dev bondEXT parent 1: classid 1:b est 1sec 8sec hfsc sc umax 1500b dmax 10ms rate 92160kbit ul
qdisc add dev bondEXT parent 1:b handle b: pfifo limit 200
#
class add dev bondINT parent 1: classid 1:c est 1sec 8sec hfsc sc umax 1500b dmax 5ms rate 2048kbit ul r
qdisc add dev bondINT parent 1:c handle c: pfifo limit 200
class add dev bondEXT parent 1: classid 1:c est 1sec 8sec hfsc sc umax 1500b dmax 5ms rate 2048kbit ul r
qdisc add dev bondEXT parent 1:c handle c: pfifo limit 200
#
class add dev bondINT parent 1: classid 1:d est 1sec 8sec hfsc sc umax 1500b dmax 5ms rate 20480kbit ul
qdisc add dev bondINT parent 1:d handle d: pfifo limit 200
class add dev bondEXT parent 1: classid 1:d est 1sec 8sec hfsc sc umax 1500b dmax 5ms rate 20480kbit ul
qdisc add dev bondEXT parent 1:d handle d: pfifo limit 200

...

class add dev bondINT parent 1: classid 1:d44 est 1sec 8sec hfsc sc umax 1500b dmax 10ms rate 30720kbit
qdisc add dev bondINT parent 1:d44 handle d44: pfifo limit 200
class add dev bondEXT parent 1: classid 1:d44 est 1sec 8sec hfsc sc umax 1500b dmax 10ms rate 30720kbit
qdisc add dev bondEXT parent 1:d44 handle d44: pfifo limit 200
```

Классом 1:b назначена скорость 90Mbit, 1:c - 2Mbit, 1:d - 20Mbit и так далее вплоть до 1:d44 - 30Mbit. Обратите внимание, что в tc определения для классов, очередей и фильтров задаются в шестнадцатиричной системе в диапазоне 1-ffff.

Классификация трафика

Классифицировать трафик можно несколькими способами. Методы *u32 match ip dst|src* и *handle MARK fw flowid* нам не подходят, особенно второй метод, когда пакет анализируется два раза - один раз в iptables, когда выставляется MARK, и второй раз, когда помаркированный пакет вторично анализируется в фильтре tc. Более производителей метод с использованием хэш-таблиц, более подробно можно почитать тут - [1] (http://lartc.org/howto/). Но этот метод подходит только когда в сети используется исключительно IPv4. Отдельно для IPv6 хэштаблицы не реализованы, а обходные методы не обеспечивают должной гибкости и простоты понимания для человека.

Поэтому был использован третий способ.

В ipset, начиная с версии 6.22 была добавлена очень полезный инструмент - оперированием структурой skb на основании правил iptables/ip6tables. Теперь логику классификации можно вынести в правила iptables и ipset.

Для удобства, здесь используется функционал etcnet.

- Создаем сеты, в которые будем вносить адреса:

```
IPv4
```

```
/etc/net/ifaces/default/fw/ipset/nethash/shaper4:
    family inet skbinfo
IPv6
/etc/net/ifaces/default/fw/ipset/nethash/shaper6:
    family inet6 skbinfo
```

- Создаем правила iptables, которые будут производить классификацию трафика:

```
IPv4
/etc/net/ifaces/default/fw/iptables/mangle/POSTROUTING:
    -o bondEXT -j SET --map-set shaper4 src --map-prio
    -o bondINT -j SET --map-set shaper4 dst --map-prio
IPv6
/etc/net/ifaces/default/fw/ip6tables/mangle/POSTROUTING:
    -o bondEXT -j SET --map-set shaper6 src --map-prio
    -o bondINT -j SET --map-set shaper6 dst --map-prio
```

- С помощью стороннего скрипта генерируем batch-файл для ipset такого вида:

```
/tmp/shape-set.txt:
flush shaper4
flush shaper6
#
add shaper4 172.16.17.196/32 skbprio 1:b
#
add shaper4 172.16.17.220/32 skbprio 1:c
#
add shaper4 172.16.1.2/32 skbprio 1:d
add shaper4 172.16.1.5/32 skbprio 1:d
add shaper4 172.16.1.10/32 skbprio 1:d
add shaper4 172.16.1.12/32 skbprio 1:d
add shaper4 172.16.1.14/32 skbprio 1:d
add shaper4 172.16.1.31/32 skbprio 1:d
add shaper4 172.16.1.103/32 skbprio 1:d
add shaper4 172.16.1.128/32 skbprio 1:d
add shaper4 172.16.1.184/32 skbprio 1:d
add shaper4 172.16.2.115/32 skbprio 1:d
add shaper4 172.16.5.198/32 skbprio 1:d
add shaper4 172.16.6.38/32 skbprio 1:d
add shaper4 172.16.6.58/32 skbprio 1:d
add shaper4 172.16.6.69/32 skbprio 1:d
add shaper4 172.16.9.228/32 skbprio 1:d
add shaper4 172.16.10.208/32 skbprio 1:d
add shaper4 172.16.11.67/32 skbprio 1:d
add shaper4 172.16.11.68/32 skbprio 1:d
add shaper4 172.16.11.147/32 skbprio 1:d
add shaper4 172.16.15.213/32 skbprio 1:d
add shaper4 172.16.15.214/32 skbprio 1:d
add shaper4 172.16.17.75/32 skbprio 1:d

...

add shaper4 172.16.17.231/32 skbprio 1:d44
add shaper6 XXXX:4680:26:0:0:0:0:202/124 skbprio 1:d44
```

обратите внимание, что адреса IPv4 заносятся в сет shaper4, адреса для IPv6 -

в shaper6. В этих сетах и происходит присвоение трафику определенные классы. Соответствие идентификаторов классов в batch-файлах tc и ipset возлагается на внешний скрипт генерации.

Загрузка параметров в систему

Загрузка правил производится по крону, запуском такого скрипта:

```
/usr/local/sbin/shaper.sh:
#!/bin/sh

ВЫЗОВ_СКРИПТА_ГЕНЕРИРУЮЩЕГО_shape-tc.hfsc_И_shape-set.txt

if [ ! -f /tmp/shape-tc.old ]; then
    touch /tmp/shape-tc.old
fi

if [ ! -f /tmp/shape-set.old ]; then
    touch /tmp/shape-set.old
fi

# данные меняются не так часто, поэтому реальная я перезагрузка
# производится только когда что-то поменялось
T=`diff /tmp/shape-tc.old /tmp/shape-tc.hfsc`
if [ "$T" != "" ]
then
    /sbin/tc -force -batch /tmp/shape-tc.hfsc > /dev/null 2>&1
    /sbin/ipset -! restore < /tmp/shape-set.txt > /dev/null 2>&1
    mv -f /tmp/shape-tc.hfsc /tmp/shape-tc.old
    mv -f /tmp/shape-set.txt /tmp/shape-set.old
else
    T=`diff /tmp/shape-set.old /tmp/shape-set.txt`
    if [ "$T" != "" ]
    then
        /sbin/tc -force -batch /tmp/shape-tc.hfsc > /dev/null 2>&1
        /sbin/ipset -! restore < /tmp/shape-set.txt > /dev/null 2>&1
        mv -f /tmp/shape-tc.hfsc /tmp/shape-tc.old
        mv -f /tmp/shape-set.txt /tmp/shape-set.old
    fi
fi
```

Категория: HOWTO
