УДК 004.738 (040)

Феденко І.І.
Московченко Д.О.

МЕТОД РЕАЛИЗАЦИИ БАЛАНСИРОВАНИЯ КАНАЛОВ ИНТЕРНЕТ-ПРОВАЙДЕРОВ НА ОСНОВЕ ТЕХНОЛОГИИ *BGP*

Национальный авиационный университет

vanya_fedenko@ukr.net
den_moskva@ukr.net

Одной из задач беспрерывной работы любой организации является отказоустойчивость и работоспособность компьютерного оборудования и корпоративной сети. Данная методика обеспечивает равномерную загрузку каналов провайдеров ТТК и РТК (имена провайдеров) входящим трафиком для исключения перегрузки каналов. Конечно, указанный алгоритм нельзя считать универсальным, потому он подойдет только в подходящих условиях

Ключевые слова: трафик, интернет провайдеры, ТТК, РТК, AS, балансирование, сервер

Введение

Вопрос о планировании нагрузки следует решать еще на ранней стадии развития любого проекта. Проблемы с сервером (а они всегда происходят неожиданно, в самый неподходящий момент) чревато весьма серьезными последствиями – как моральными, так и материальными. Первоначально проблемы недостаточной производительности сервера в связи ростом нагрузок можно решать путем наращивания мощности сервера, или же оптимизацией используемых алгоритмов, программных кодов и так далее. Но рано или поздно наступает момент, когда и эти меры оказываются недостаточными. Приходится прибегать к кластеризации: несколько серверов объединяются в кластер; нагрузка между ними распределяется при помощи комплекса специальных методов, называемых балансировкой. Помимо решения проблемы высоких нагрузок кластеризация помогает также обеспечить резервирование серверов друг на друга. Эффективность кластеризации напрямую зависит от того, как распределяется (балансируется) нагрузка между элементами кластера. Балансировка нагрузки может осуществляться при помощи как аппаратных, так и программных инструментов. Об основных методах и алгоритмах и балансировки мы бы хотели рассказать в этой статье. Процедура балансировки осуществляется при помощи целого комплекса алгоритмов и методов, соответствующим следующим уровням модели *OSI*:

- сетевому;
- транспортному;
- прикладному.

Алгоритмы и методы балансировки

Существует много различных алгоритмов и методов балансировки нагрузки. Выбирая конкретный алгоритм, нужно исходить, во-первых, из специфики конкретного проекта, а во-вторых, из целей, которые мы планируем достичь.

В числе целей, для достижения которых используется балансировка, нужно выделить следующие:

- справедливость: нужно гарантировать, чтобы на обработку каждого запроса выделялись системные ресурсы и не допустить возникновения ситуаций, когда один запрос обрабатывается, а все остальные ждут своей очереди;

- эффективность: все серверы, которые обрабатывают запросы, должны быть заняты на 100%; желательно не допускать ситуации, когда один из серверов простаивает в ожидании запросов на обработку (сразу же оговоримся, что в реальной практике эта цель достигается далеко не всегда);

- сокращение времени выполнения запроса: нужно обеспечить минимальное время между началом обработки запроса (или его постановкой в очередь на обработку) и его завершения;

- сокращение времени отклика: нужно минимизировать время ответа на запрос пользователя.

Очень желательно также, чтобы алгоритм балансировки обладал следующими свойствами:

- предсказуемость: нужно понимать в полном объеме, в каких ситуациях и при каких нагрузках алгоритм будет эффективным для решения поставленных задач;

- равномерная загрузка ресурсов системы;

- масштабируемость: алгоритм должен сохранять работоспособность при увеличении нагрузки.

Анализ проблемы

Основная задача – нужно обеспечить рациональное использование двух каналов в сеть Интернет и позаботиться об автоматическом резервировании, чтобы работоспособность сети быстро восстанавливалась при проблемах на одном из каналов. Допустим оба канала подключены к нашему серверу в виде *Ethernet*, т.е. подключаются на обычные сетевые карты и настраиваются указанием *IP*-адреса самого интерфейса, адреса шлюза по умолчанию и маски подсети (т.е. как статические маршруты). Через третью сетевую карту подключается локальная сеть. Для определённости в дальнейших примерах будем считать, что внешними интерфейсами являются *rl*0 и *rl*1, а *ed*0 – внутренний. В зависимости от условий договоров с провайдерами могут возникнуть следующие подзадачи:

- переключение на более дорогой канал только на время проблем с дешёвым;
- направление трафика, чувствительного к задержкам, в более «быстрый» канал, в то время как весь не критичный к скорости доставки пакетов трафик (например, *FTP*-закачки) направлять в «медленный», но более дешевый канал;
- пропорциональная балансировка нагрузки между каналами, имеющими сопоставимые по качеству и стоимости характеристики.

Понятно, что определенными манипуляциями можно управлять лишь тем трафиком, который инициируется хостами локальной сети, получающими доступ в Интернет через данный сервер, либо самим сервером. То есть можно управлять тем, через какой из каналов будет загружаться запрошенная пользователем страница или выгружаться на удалённый *FTP*-сервер какой-то файл. А вот трафик, инициированный «снаружи» (например, почтовый или запросы на веб-сервер компании), необходимо обслуживать на том канале, куда он придёт.

Разработка алгоритма и методики балансировки каналов на основе технологии BGP

1. *BGP* позволяет управлять вероятностью т.е. указать предпочтительный входящий маршрут для определенной сети. Используется для этого искусственное «удлинение» маршрута — при анонсировании своего маршрута соседу можно несколько раз повторить номер своей *AS*. (*AS*- автономная система).

При этом, каждому соседу можно «удлинять» маршрут по-разному. Предпочтительным оказывается более короткий маршрут.

2. *BGP* позволяет делать описания отдельных частей *AS*. Т.е. в *RIPE* наша *AS* описана как 1.1.144.0/22, никто не мешает дополнительно в *BGP* описать (т.е. анонсировать) 1.1.144.0/24, 1.1.145.0/24, 1.1.146.0/24 и 1.1.147.0/24.

3. Алгоритм выбора маршрута при *BGP*-маршрутизации из нескольких имеющихся:

- выбирается маршрут с большей маской. Если не выбрано, то далее;
- выбирается более короткий маршрут (меньше промежуточных *AS*); Если не выбрано, то далее;
- выбирается маршрут, объявленный раньше (считается более надежный). Если не выбрано, то далее;

- однозначный псевдослучайный выбор.

К сожалению, «управление предпочтением» совсем не значит «управление вероятностью». На деле оборачивается тем, что почти весь трафик начинает идти по предпочитаемому маршруту. Т.е. используя удлинения маршрутов, плавно регулировать каналы не получится.

Идея

Большинство клиентов имеют «серые» *IP* адреса. Соответственно, на шлюзе транслируются сервером NAT. И это основной по объему трафик.

Всю нашу *AS* можно разбить на 4 части (1.1.144.0/24, 1.1.145.0/24, 1.1.146.0/24 и 1.1.147.0/24) и использовать их по-разному. Например, первые две для клиентов с «белыми» *IP* адресами, третья — предпочтение РТК и четвертая — предпочтение ТТК. Именно так сделано в примере.

На уровне правил файрвола *iptables* принимать решение какие адреса использовать для *NAT*.

Если адрес клиента 192.168.1-*N*.0/24, то для сервера *NAT* использовать 1.1.146.0/24. Иначе — 1.1.147.0/24.

Таким образом, изменяя *N*, можно балансировать входящий трафик двух каналов (рис.1)

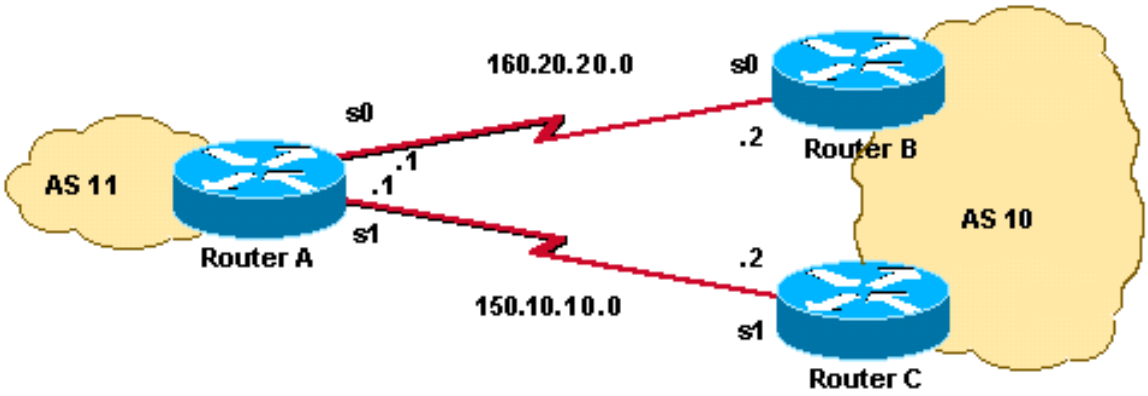

Рис. 1.Схема балансировки входящего трафика

Реализация

1. Для проверки принадлежности *IP* клиента к 192.168.1-*N*.0/24, в *iptables*, использовать модуль *ipset*, в правилах далее набор «*rtk*».

Цепочка «*NAT_AS*», которая транслирует:

:NAT_AS — [0:0]

-A NAT_AS -m state -s 192.168.0.0/16 --state ESTABLISHED,RELATED -j SNAT --to-source 91.235.146.0-91.235.147.255 --persistent

#PTK

-A NAT_AS -m state -m set --set rtk src --state NEW -j SNAT --to-source 1.1.146.0-1.1.146.255 --persistent

#TTK

-A NAT_AS -m state -m set! --set rtk src --state NEW -j SNAT --to-source 1.1.147.0-1.1.147.255 --persistent

-j SNAT используется с параметром *--persistent*. Этот параметр позволяет клиенту использовал постоянный «белый» *IP*. Без этого у клиента могут быть проблемы на многих сервисах в интернете.

В NAT */ POSTROUTING.*

 # eth1, eth3 — интерфейсы на которые подключены ТТК и РТК

-A POSTROUTING -s 192.168.0.0/16 -o eth1 -j NAT_AS

-A POSTROUTING -s 192.168.0.0/16 -o eth3 -j NAT_AS

2. Далее нужно подготовить набор правил *ipset «rtk»*, это файл, в котором могут храниться все параметры и использоваться в нескольких сервисах.

```
cat param_rtk_set:
export rtk_start=1
export rtk_min=1
export rtk_max=99
# PTK
export shp_rtk_max=547
# TTK
export shp_ttk_max=535
export scale=50
export f_set_end=/lib/init/rw/rtk_set_end
```

Непосредственно создание и обновление набора *rtk*. Нужно запускать регулярно.

```
cat create_rtk_set
#! /bin/sh
/usr/sbin/ipset -N rtk nethash -q
/usr/sbin/ipset -N temp_rtk nethash -q
/usr/sbin/ipset -F temp_rtk -q
. ./param_rtk_set
rtk_set_end=0
if [ -f $f_set_end ]; then
read rtk_set_end < $f_set_end
else
rtk_set_end=$(($rtk_min + $rtk_max))
rtk_set_end=$(($rtk_set_end / 2))
echo $rtk_set_end > $f_set_end
fi
net=$rtk_start
while [ $net -lt $rtk_set_end ]; do
/usr/sbin/ipset -A temp_rtk
192.168.${net}.0/24 -q
net=$(($net + 1))
done
/usr/sbin/ipset -W temp_rtk rtk
/usr/sbin/ipset -X temp_rtk -q
```

ОК, «управляющее воздействие» есть. Т.е. изменяя *N* (который хранится в /*lib*/*init*/*rw*/*rtk_set_end*), можно плавно изменять соотношения входящих трафиков ТТК и РТК.

Автоматизация

```
cat rtk-ttk:
ttk=$(/sbin/ifconfig eth1 | grep -Eo «RX
bytes:[0-9]*» | grep -Eo "[0-9]*")
if [ "$ttk" = "" ]; then
echo «No TTK ifconfig»
exit
fi
rtk=$(/sbin/ifconfig eth3 | grep -Eo «RX
bytes:[0-9]*» | grep -Eo "[0-9]*")
if [ "$rtk" = "" ]; then
echo «No RTK ifconfig»
exit
fi
work_dir="/lib/init/rw/"
read ttk_old < ${work_dir}shp_ttk_old
read rtk_old < ${work_dir}shp_rtk_old
echo $ttk > ${work_dir}shp_ttk_old
echo $rtk > ${work_dir}shp_rtk_old
if [ $ttk -le $ttk_old ]; then
echo «TTK RX smoll»
exit
fi
if [ $rtk -le $rtk_old ]; then
echo «TTK RX smoll»
exit
fi
ttk_cur=$(($ttk — $ttk_old + 1))
rtk_cur=$(($rtk — $rtk_old + 1))
. ./param_rtk_set
max_delta=5
p=$( echo «scale=10; $scale *
($shp_rtk_max / $shp_ttk_max) / ($rtk_cur /
$ttk_cur) + 100.5» | /usr/bin/bc )
p=${p%%.*}
p=$(( $p — 100 ))
n_for=1
while [ $scale -lt $p ]; do
#echo "$p add"
p=$(( $p — 1 ))
n_for=$(( $n_for + 1 ))
if [ $max_delta -lt $n_for ]; then
break
fi
./add_rtk
done
n_for=1
while [ $p -lt $scale ]; do
#echo "$p del"
p=$(( 1 + $p ))
n_for=$(( $n_for + 1 ))
if [ $max_delta -lt $n_for ]; then
break
fi
```

```
./del_rtk
done
./create_rtk_set
```

Настройка BGP

Для того, чтобы реализовать возможность управления, входящего трафика, нужно настроить сам инструмент *BGP*. Пример *bgp.conf* (настоящие *IP* и номера изменены под исходные данные)

```
hostname AS0000
password ****
enable password ****
log file /var/log/quagga/bgpd.log
router bgp 0000
no synchronization
bgp router-id
network 1.1.144.0/22
network 1.1.144.0/24
network 1.1.145.0/24
network 1.1.146.0/24
network 1.1.147.0/24
neighbor [IP шлюза РТК] remote-as [AS РТК (только номер, например 12345)]
neighbor [IP шлюза РТК] update-source [внешний IP РТК]
neighbor [IP шлюза РТК] route-map MY-OUT-RTK out
neighbor [IP шлюза РТК] route-map INTER_NET in
neighbor [IP шлюза ТТК] remote-as [AS ТТК (только номер, например 12345)]
neighbor [IP шлюза ТТК] update-source [наш внешний IP ТТК]
neighbor [IP шлюза ТТК] route-map MY-OUT-TTK out
neighbor [IP шлюза ТТК] route-map INTER_NET in
ip prefix-list upstream-out seq 10 permit 1.1.144.0/22
ip prefix-list up144 seq 10 permit 1.1.144.0/24
ip prefix-list up145 seq 10 permit 1.1.145.0/24
ip prefix-list up146 seq 10 permit 1.1.146.0/24
ip prefix-list up147 seq 10 permit 1.1.147.0/24
!==========================
!--- MY-OUT-TTK
route-map MY-OUT-TTK permit 10
match ip address prefix-list up144
!set as-path prepend 0000 0000
route-map MY-OUT-TTK permit 20
match ip address prefix-list up145
!set as-path prepend 0000 0000
route-map MY-OUT-TTK permit 30
match ip address prefix-list up146
set as-path prepend 0000
route-map MY-OUT-TTK permit 40
!match ip address prefix-list up147
!set as-path prepend 0000 0000
!route-map MY-OUT-TTK deny 200
!==========================
route-map MY-OUT-TTK permit 100
match ip address prefix-list upstream-out
!set as-path prepend 0000 0000 0000
route-map MY-OUT-TTK deny 200
!--- конец MY-OUT-TTK
!==========================
!--- MY-OUT-RTK
route-map MY-OUT-RTK permit 10
match ip address prefix-list up144
set as-path prepend 0000
route-map MY-OUT-RTK permit 20
match ip address prefix-list up145
set as-path prepend 0000
route-map MY-OUT-RTK permit 30
match ip address prefix-list up146
!set as-path prepend 0000 0000 0000
route-map MY-OUT-RTK permit 40
match ip address prefix-list up147
set as-path prepend 0000
!==========================
route-map MY-OUT-RTK permit 100
match ip address prefix-list upstream-out
set as-path prepend 0000 0000
route-map MY-OUT-RTK deny 200
!--- MY-OUT-RTK
!==========================
!---- Local nets
ip prefix-list local_ seq 15 permit 192.168.0.0/16
ip prefix-list local_ seq 18 permit 0.0.0.0/8
ip prefix-list local_ seq 19 permit 127.0.0.0/8
ip prefix-list local_ seq 20 permit 10.0.0.0/8
ip prefix-list local_ seq 21 permit 172.16.0.0/12
ip prefix-list local_ seq 22 permit 169.254.0.0/16
```

```
ip prefix-list local_ seq 23 permit 224.0.0.0/4
ip prefix-list local_ seq 24 permit 240.0.0.0/4
route-map INTER_NET deny 10
match ip address prefix-list local_
route-map INTER_NET permit 200
set local-preference 500
line vty
```

Здесь «0000» — номер *AS*, «1.1.» — начало используемых *IP*.

Выводы

В статье были рассмотрены алгоритм и метод настройки балансирования оборудования для заданной сети с определенным программным и аппаратным обеспечением. Алгоритм построен на главных факторах, которые могут повлиять на отказоустойчивость и работоспособность корпоративной сети. Пример методики построен и рассмотрен на основе технологии BGP. Она широко применяется в крупных сетях, где важно чтобы сеть работала исправно и беспрерывно. Когда появляется проблема с одним из провайдеров, то система должна автоматически переключится на второй канал. BGP является протоколом прикладного уровня и функционирует поверх протокола транспортного уровня TCP. После установки соединения передается информация обо всех маршрутах, предназначенных для экспорта. В дальнейшем передается только информация об изменениях в таблицах маршрутизации. При закрытии соединения удаляются все маршруты, информация о которых передана противоположной стороной. Данная методика полностью рабочая и протестирована в корпоративной сети одного из крупных ООО Украины.

Список литературы

1. A border gateway protocol 4 [Електронний ресурс]: https://tools.ietf.org/html/rfc4271

2. Zhukov I.A., Pechurin N.K., Kondratova L.P., Pechurin S.N. The algoritms coordinating traffic in computer network.:Збірник наукових праць: Випуск 4 (52). – К.: НАУ, 2015. – С. 31-36.

3. Практические примеры BGP [Електронний ресурс]: https://www.cisco.com/c/ru_ru/support/docs/ip/border-gateway-protocol-bgp/26634-bgp-toc.html

4. Manual:BGP Case Studies [Електронний ресурс] https://wiki.mikrotik.com/wiki/Manual:BGP_Case_Studies.

Статью представлено в редакцию 15.08.2017