Подключение к 2 ISP без собственной автономной системы BGP с использованием маршрутизатора cisco

Типичная задача, которая тем не менее, продолжает вызывать массу вопросов.

Попробую вкратце описать суть технологии и подводные камни.

Итак, пусть у нас есть один пограничный маршрутизатор cisco с одним внутренним портом (g0/0) и двумя внешними (f0/0, f0/1). Есть подключение к двум провайдерам, каждый из которых выдал свой пул адресов Pool(ISP1) и Pool(ISP2) (это некоторые сети, принадлежащие конкретному провайдеру). Пусть для простоты адреса интерфейсов f0/0 и f0/1 из этих же пулов. И адреса шлюзов из этих же пулов (Gate(ISP1) и Gate(ISP2) соответственно).
Так как у нас нет возможности поднять BGP, значит мы должны на каждого из провайдеров прописать маршрут по умолчанию. И вот тут возникает первый вопрос: какую задачу мы хотим решить? Резервирование или одновременная работа с двумя провайдерами?

Резервирование.

В этой топологии одновременно работает только один провайдер. То есть мы должны организовать проверку провайдера ISP1 и в случае если он живой – ходить через него, а если «мертв», то переключаться на запасного провайдера ISP2. Здесь есть подводный камень: NAT. Мы можем написать несколько правил трансляции, но надо как то указать, что при выходе через ISP1 мы используем Pool(ISP1), а при выходе через ISP2 – Pool(ISP2), иначе маршрутизатор всегда будет использовать трансляцию, которая первой написана в конфигурации. Понятно, что если идти через ISP2, а адреса источника будут из Pool(ISP1), то в лучшем случае мы получим несимметричную маршрутизацию, в худшем пакеты вообще никуда не дойдут, например потому, что провайдеры выполняют предписание использовать фильтрацию по RFC2827, что означает не принимать пакеты с адресами источника не из своей сети.
Итак, у нас 2 подзадачи: проверка провайдера (маршрута) на «живость» и трансляция адресов с учётом выходного интерфейса.

Проверка на «живость».
Маршрутизаторы cisco обладают замечательной технологией, называемой SLA. При помощи неё можно не только проверять пингом некий адрес, но также проверять живость определенных сервисов (ftp-connect, tcp-connect) или параметра канала связи (icmp-jitter, udp-jitter). Здесь

рассмотрим самый простой и распространенный способ – пинг определенного хоста. Для простоты будем пинговать адрес шлюза провайдера Gate(ISPX). Если надо пинговать другой адрес, то необходимо явно прописать маршрут до этого адреса через конкретного провайдера, которого мы проверяем.

```
! Задаем параметры «пинговалики»
ip sla {#}
  icmp-echo {ip} [source-interface {int}]
!
! Запускаем пинговалку
ip sla schedule {#} start now life forever
!
! Настраиваем «переключатель» (track), от которого будет зависеть маршрут
track {#} ip sla {#} reachability
!
! Настраиваем маршрут по умолчанию с трекингом
ip route 0.0.0.0 0.0.0.0 {next-hop} track {#}
```

Примечание: в старых IOS команда привязки track к sla выгдялела так

```
track {#} rtr {sla#} reachability
```

Если хост пингуется, то track будет в состоянии UP и маршрут будет в таблице маршрутизации. А если пинг пропадет, то через настроенный промежуток времени (по умолчанию 3*10 секунд) track поменяет состояние на DOWN и маршрут будет удален до тех пор, пока track вновь не изменит состояние.

Пример:

```
ip sla 1
  icmp-echo Gate(ISP1)
ip sla schedule 1 start now life forever
track 11 ip sla 1 reachability
ip route 0.0.0.0 0.0.0.0 Gate(ISP1) track 11
```

ISP2 можно не проверять, чтобы не создавать лишний служебный трафик в канал, т.к. он у нас запасной и может быть дорогим (спутниковый канал, к примеру, или коммутируемый канал, оплачиваемый по времени работы). Маршрут на второго провайдера мы напишем с большей административной дистанцией и тем самым заставим его работать только при пропадании основного.

Задание правил трансляции адресов с учетом исходящего интерфейса.

Тут на самом деле тоже 2 задачи: динамическая трансляция и статическая трансляция адресов. Первая нам нужна для выхода наружу, а вторая – для анонса сервисов. И в том и в другом случае нам понадобится конструкция, называющаяся route-map (создать надо будет по route-map на каждого провайдера)

```
! Создаем route-map
route-map ISPX permit {#}
  ! Указываем критерий попадания в этот абзац route-map
```

```
  match interface {исходящий интерфейс}
```

Тут есть тонкость: при указании слова interface в подсказке пишется

```
  interface         Match first hop interface of route
```

Т.е. вообще говоря, не понятно, что это за параметр. Плюс в зависимости от того, что написано на самом интерфейсе, этот критерий может означать как входящий интерфейс, так и исходящий! А зависит это от того, что написано в команде ip nat на интерфейсе:

`ip nat inside` – критерий будет означать входящий интерфейс
`ip nat outside` – критерий будет означать исходящий интерфейс

Далее, нам понадобится пул адресов от каждого провайдера

```
ip nat pool PoolX {start-ip(ISPX)}  {end-ip(ISPX)}
```

И можно уже писать правила NAT на каждого провайдера

```
 ip nat inside source route-map ISPX poolX overload
```

`overload` – ключевое слово, означающее использовать PAT (Port Address Translation, трансляцию с учётом порта источника)
Если к надо добавить статические трансляции, то делаем почти так же (пусть серверу мы зарезервировали адрес Srv(ISPX) от каждого провайдера, а локальный адрес у сервера – Srv(LAN).)

```
  ip nat inside source static Srv(LAN) Srv(ISPX) route-map ISPX
```

При этом конечно надо озаботиться, чтобы оба адреса (Srv(ISP1) и Srv(ISP2)) на ДНС серверах были прописаны и указывали на одно и то же имя.

Если мы терминируем на этом же маршрутизаторе туннели IPSec, то достаточно указать один и тот же crypto map на обоих внешних интерфейсах, а на соседних маршрутизаторах указать в соответствующих абзацах crypto map оба адреса, как адреса соседей, указав на них один и тот же ключ или сертификат.

```
! на нашем маршрутизаторе
int f0/0
  crypto map MYMAP
int f0/1
  crypto map MYMAP

! на соседнем маршрутизаторе
crypto map TOCENTER 10
  set peer Address(ISP1) default
  set peer Address(ISP2)
  set transform-set {TSET}
  match address {ACL}
crypto isakmp key {КЛЮЧ} address Address(ISP1)
crypto isakmp key {КЛЮЧ} address Address(ISP2)
```

Итого, у нас получилось:

```
!
! интерфейсы
int g0/0
  ip address [LAN]
  ip nat inside
!
int f0/0
  ip address Address(ISP1)
  ip nat outside
  crypto map MYMAP
!
int f0/1
  ip address Address(ISP2)
  ip nat outside
  crypto map MYMAP
!
! Маршрутизация
ip sla 1
  icmp-echo Gate(ISP1)
ip sla schedule 1 start now life forever
track 11 ip sla 1 reachability
ip route 0.0.0.0 0.0.0.0 Gate(ISP1) track 11
ip route 0.0.0.0 0.0.0.0 Gate(ISP2) 50
!
! Пулы для NAT
ip nat pool POOL1 {start-ip(ISP1)}  {end-ip(ISP1)}
ip nat pool POOL2 {start-ip(ISP2)}  {end-ip(ISP2)}
!
! route-map для NATa
route-map ISP1 permit 10
  match interface f0/0
!
route-map ISP2 permit 10
  match interface f0/1
!
! Правила NATa
ip nat inside source route-map ISP1 POOL1 overload
ip nat inside source route-map ISP2 POOL2 overload
ip nat inside source static Srv(LAN) Srv(ISP1) route-map ISP1
ip nat inside source static Srv(LAN) Srv(ISP2) route-map ISP2
```

Одновременное использование двух провайдеров

Если в первом случае все понятно и однозначно, то в случае с одновременным использованием двух провайдеров возникают проблемы.
Для начала: нам надо обоих провайдеров проверять на «живость» и переключать все потоки на одного в случае, если кто то «упал». Это делается полностью аналогично проверке ISP1 в главе про

Резервирование. С тем лишь отличием, что оба маршрута по умолчанию имеют одинаковую административную дистанцию

```
  ip route 0.0.0.0 0.0.0.0 Gate(ISP1) track 11
  ip route 0.0.0.0 0.0.0.0 Gate(ISP2) track 22
```

Правила NATa не претерпевают изменений. Те же route-map в динамических и статических трансляциях. Но как определить, какой пакет отправлять через какого провайдера? Можно принудительно раскидывать входящие с g0/0 пакеты ещё одним route-map.

```
! Заготавливаем списки доступа с «интересным» трафиком. Все знают, что такое
! «интересный» трафик? ☺
ip access-list extended ACLISP1
  permit {трафик на ISP1}
!
ip access-list extended ACLISP2
  permit {трафик на ISP2}
!
route-map STRELKA 10
  match ip address ACLISP1
  set ip next-hop {GateISP1}
route-map STRELKA 20
  match ip address ACLISP2
  set ip next-hop {GateISP2}
!
int g0/0
  ip policy route-map STRELKA
```

Правда в этом случае пакет, попавший в ACLISP1, пойдет на первого провайдера всегда, независимо от того, жив провайдер или нет. Чтобы этого избежать есть возможность в этом route-map применить проверку по track

```
  set ip next-hop verify-reachability {GateISPX} {sequence#} track {track#}
```

`sequence#` - это число от 1 до 65535. Если таких возможных next-hop будет много, то они будут упорядочены по этому числу.

Напомню, что в случае если пакет явно не попал в route-map, он будет использовать обычную таблицу маршрутизации.

Ну а теперь давайте попробуем разобраться с двумя очень сложными вопросами: каким образом будут ходить пакеты, если вы не используете явно разделение трафика при помощи route-map на внутреннем интерфейсе. И как сделать так, чтобы пакет, пришедший снаружи на адрес сервера Srv(ISP1) ушел обратно через тот же интерфейс, через который пришёл. Это действительно сложные вопросы. И красивого решения для них нет, поэтому я в своей практике стараюсь избегать таких топологий.
Однако, жизнь может заставить, поэтому разберемся.

Пусть снаружи приходит пакет на интерфейс f0/0 на адрес Srv(ISP1). Благодаря статической трансляции адрес назначения будет изменен на Srv(LAN) и пакет пойдет дальше на сервер. На

маршрутизаторе в кеше NAT трансляций появится запись о соответствии Srv(LAN) и Srv(ISP1). Сервер ответит, ответ дойдет обратно до маршрутизатора и…возникнет вопрос: по какому маршруту отправлять пакет в Интернет? В кеше трансляций есть явная запись, какой адрес ставить вместо Srv(LAN) – Srv(ISP1). Но нет ни намека, через какой интерфейс при этом посылать пакет. Для исправления этой неоднозначности надо по какому то критерию разделять приходящий с разных интерфейсов трафик. Этого можно добиться, но способ, по моему мнению, не очень элегантный: надо использовать подмену реальных адресов клиентов на разные пулы внутренних адресов. Надо только подобрать размер этого пула соответственно нагрузке на сервер – по одному адресу на каждого обращающегося, т.к. для внешнего NATa (outside NAT) на маршрутизаторах нельзя сделать PAT (Port Address Translation), только трансляция адрес в адрес. Тогда всегда точно известно, с какого интерфейса поступил запрос. В качестве критерия для трансляции адреса сервера можно в существующие route-map добавить такие списки доступа

```
ip access-list extended FORISPX
  permit ip host Srv(LAN) OUTPOOLX
```

Таким образом получим:

```
! задаем пулы для каждого из интерфейсов
ip nat pool OUTPOOL1 {start-ip-1} {end-ip-1}
ip nat pool OUTPOOL2 {start-ip-2} {end-ip-2}
!
! задаем критерий для outside NAT
ip access-list extended OUTNAT1
  permit ip any host Srv(ISP1)
ip access-list extended OUTNAT2
  permit ip any host Srv(ISP2)
!
! транслируем адреса источника клиентов
ip nat outside source list OUTNAT1 pool OUTPOOL1
ip nat outside source list OUTNAT2 pool OUTPOOL2
!
! дополняем route-map
route-map ISP1 permit 10
  match interface f0/0
  match ip address FORISP1
!
route-map ISP2 permit 10
  match interface f0/1
  match ip address FORISP2
```

Это решение, пусть не красивое, но все же полностью решает задачу, не затрагивая сервер (его часто и нельзя затрагивать: например, если это не приложение, а VPN сервер). Потеря здесь явная одна: сервер никогда не знает, с кем реально он общается, а значит нельзя собрать адекватную статистику и т.д.

Если же использовать возможности серверов, то на ум приходит несколько решений, не требующих внешнего NATa.
Первое – сделать на самом деле 2 сервера-партнера, с разными адресами и связанными друг с другом для репликации ещё одним линком. Каждый сервер транслируется в свой адрес, но в случае падения одного из каналов переключается на партнерский адрес.

Не самое простое и дешевое решение.

Второе: задать на одном и том же сервере 2 адреса. Если они из одной подсети, то проблемы с маршрутизацией не будет. Каждый из локальных адресов сервера строго транслируется только одного из провайдеров, т.к. физически сервер один и никакой выгоды по нагрузке мы все равно не получим. Для явного указания выходного интерфейса применяем route-map STRELKA. Тут может возникнуть проблема с самим сервером: часто бывает так, что при ответе сервер использует только первичный адрес интерфейса, независимо от того, на какой адрес пришел запрос.
Характерный пример: VPN сервер. Если в качестве VPN сервера выступает маршрутизатор cisco, то он всегда отвечает с первичного адреса интерфейса.
Если адреса из разных подсетей, то шлюз по умолчанию все равно один. А значит маршрутизироваться все будет только через один интерфейс и задача полностью решена не будет.

Сергей Фёдоров, CCIE Security #22974