Процесс принятия решения в BGP (коротко о главном)

Итак после длительного и мучительного ожидания, переписок с RIPE и некоторых расходов Вы таки получили свою AS и сеть к ней, и хотите влиться в дружный коллектив интернет-провайдеров. Или еще не получили, но уже хотите разобраться в том как работает BGP. Или хотите узнать что-то новое.

В данный статье я постараюсь осветить некоторые "общеизвестные" моменты настройки работы протокола BGP, которые многим начинающим незнакомы или освещены недостаточным образом. Главный упор будет делаться на практические примеры.

Итак что есть BGP ? BGP - это модифицированный дистанционно-векторный протокол маршрутизации, созданный с прицелом решать не столько технические, сколько политические задачи по управлению глобальной сетью и взаимодействию внутри нее. Из чего же, так сказать, состоит BGP? Главными элементами этого протокола являются:

- Автономная система (AS),
- eBGP/IGP(в том числе iBGP),
- соседи ("нейборы"),
- атрибуты и работа с ними.

Перечислив этот список попытаемся разобраться как ОНО работает. Начнем с основы - автономной системы. Автономная система представляет собой набор логически связанных маршрутизаторов, управление которыми осуществляется на основе единой и согласованной административной и технической политики. Глобальная сеть является совокупностью автономных систем. То есть, иначе говоря

В качестве транспорта для протокола BGP используется протокол TCP (порт 179). Маршрутизаторы, на которых включена поддержка протокола BGP общепринято называть спикерами. 2 спикера, которые обмениваются между собой маршрутной информацией по протоколу BGP называются соседями или "нейборами", или же "пирами". При работе с протоколом BGP, взаимодействующих соседей надо указывать явным образом. Соседи устанавливаем друг с другом tcp-соединения и начинают "договариваться" об установке соединения для обмена информацией по протоколу BGP. Такого рода соединение называется "BGP сессия".

Процесс создания BGP сессии достаточно сложен и проходит множество стадий, на которых соседи обмениваются различной служебной информацией и сверяют ее. Если на этапе обмена служебной информацией не произошло никаких ошибок, то соединение переходит в состояние, когда происходит обмен маршрутной информацией. Такое состояние BGP сессии называется установленное (Established). Процесс обмена маршрутами состоит в обмене специальным набором данных, которые состоят из префиксов сетей и их атрибутов. Ну, если префикс сети - это префикс сети и есть, и с ним все ясно, то что есть атрибуты ?

Атрибуты - это специальные данные, соответствующие префиксу и несущие ряд служебной информации о данном префиксе. К примеру, атрибут NEXT_HOP указывает, на какой ip-адрес надо направить пакеты, чтобы достичь той или иной сети.

Атрибуты бывают обязательные и необязательными, а также транзитивными или не транзитивными. Обязательный атрибут маршрута - это такой атрибут, без которого невозможна передача маршрута. К примеру, это, уже указанный выше, NEXT_HOP. Транзитивными же называются те атрибуты, передача которых производится не только между маршрутизаторами внутри своей AS, но и в другие AS. Не транзитивные же атрибуты за пределы своей AS "не выходят", то есть в другие AS не передаются.

К обязательным атрибутам относятся:

ORIGIN - Обязательный транзитивный атрибут. Указывает источник обновления маршрута, то есть откуда пришла информация о маршруте. Может принимать 3 значения:

- IGP - информация о доступности маршрута была генерированной внутри той AS, откуда пришел префикс.
- EGP - информация о доступности сети была получена через протокол EGP.
- INCOMPLETE - информация о доступности сети была получена по другому, то есть неизвестно как.

AS_PATH - Обязательный транзитивный атрибут. Определяет путь, через который проходит маршрут, прежде, чем попасть к нам. К примеру, если есть префикс x.y.z.w/N в некой AS (к примеру, 1), и между нами и AS1 расположены AS с номерами 3,5 и 7, то AS_PATH этого префикса будет иметь вид: 1 3 5 7. А после прохождения нашей AS, в AS_PATH добавится номер нашей AS (к примеру - 10) и примет вид 1 3 5 7 10.

NEXT_HOP - Обязательный транзитивный атрибут. Мы уже вскользь упоминали его. Суть данного атрибута - указать ip, который является шлюзом для достижения той или иной сети, причем в зависимости от характера работы протокола BGP, характер его поведения видоизменяется (подробнее об этом - чуть позже).

Кроме этих 3-х обязательных атрибутов есть еще и множество необязательных, рассматривать которые сразу (в отрыве от применения) нет смысла.

Однако после того, как был получен префиксы со всеми своими атрибутами, наш маршрутизатор должен принять решение относительно того, какой путь к искомой сети лучше, и занести лучший в таблицу маршрутизации. Решение о том, какой путь лучше, принимается на основании сравнения атрибутов по следующему алгоритму:

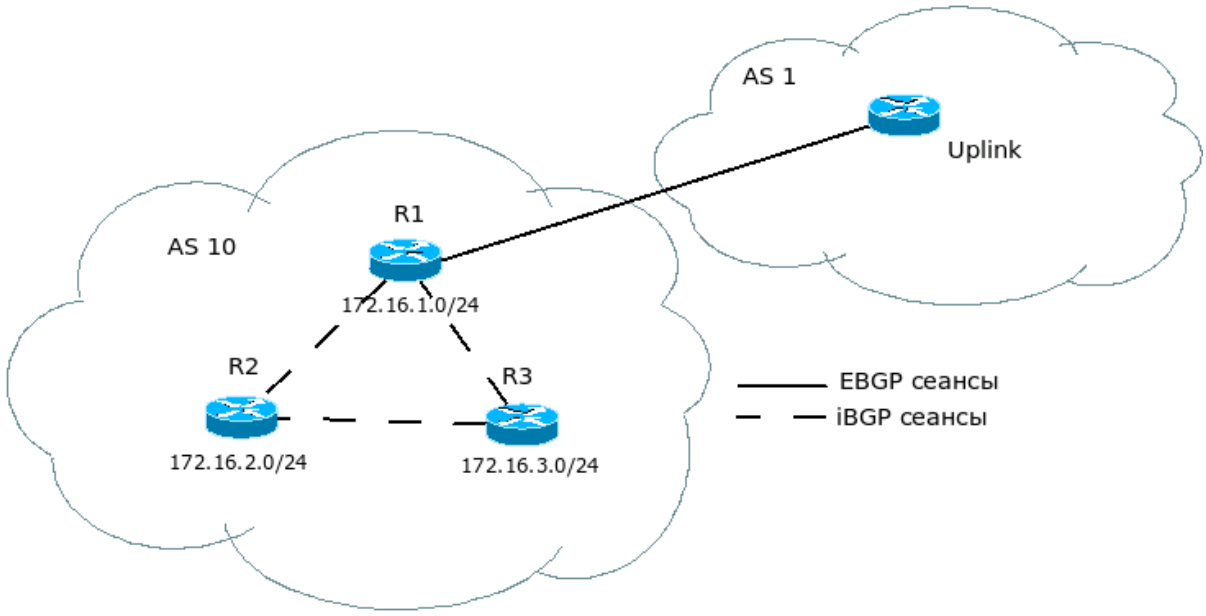
1. Если NEXT_HOP маршрута не доступен, маршрут игнорируется.
2. Сравнивается вес (атрибут WEIGHT) маршрутов. Маршрут с большим весом является предпочтительным (атрибут WEIGHT - это локальный атрибут, впервые появился на маршрутизаторах Cisco).
3. При равном весе сравниваются атрибуты локального предпочтения LOCAL_PREFERENCE. Опять же, чем больше LOCAL_PREFERENCE, тем "лучше" маршрут.
4. При равном LOCAL_PREFERENCE сравниваются AS_PATH. Чем AS_PATH короче - тем лучше.
5. Если AS_PATH равны, то сравнивается ORIGIN. При этом IGP лучше EGP, а EGP лучше INCOMPLETE.
6. При равных ORIGIN сравниваются атрибуты MED. Чем меньше MED - тем лучше маршрут.
7. Если MED равны, то eBGP маршруты лучше iBGP.
8. Если все еще наблюдается равенство, то выбирается маршрут с путем, кратчайшим по отношению к NEXT_HOP.
9. И уж как последняя надежда - сравнение атрибутов ROUTER_ID. ROUTER_ID - это уникальный ID маршрутизатора (чаще всего берется ip с интерфейса обратной петли). Чем меньше - тем лучше.
10. Но даже если по ROUTER_ID не удалось принять решение, то выбирается тот маршрут, BGP-сессия с которым на текущий момент длится дольше.

Маршрут выбрали и даже занесли в таблицу маршрутизации. Но перед тем, как привести пример начальной настройки необходимо осветить несколько нюансов:

1. AS_PATH предназначен для избежания петель маршрутизации. Делается это очень просто - если в AS_PATH принятого маршрута есть наша AS - маршрут игнорируется.
2. Внутри AS, для обеспечения связности между всеми маршрутизаторами, входящими в эту AS и работающих по протоколу BGP необходимо иметь iBGP сессии, то есть каждый маршрутизатор внутри AS должен быть логически связан с другими.

Для понимания процесса приведем несколько примеров настройки протокола BGP. Условия: есть AS , в ней 3 маршрутизатора. Один из них R1 - соединен с с аплинком, а два других (R2 и R3) с R1 и между собой.

Простейшая настройка BGP



R1:

```
! включаем поддержку bgp. NN - это номер Вашей AS, для примера - 10
router bgp NN
! отключаем синхронизацию с IGP
no synchronization
! указываем какие сети анонсировать
! формат  network сеть  mask маска опции
! стоит отметить, что сеть анонсируется только при наличии ее в
! таблице маршрутизации
network 172.16.1.0 mask 255.255.255.0
! описываем соседей. Формат  neighbor адрес_bgp_соседа опции
! Аплинк
! опция remote-as указывает на AS соседа.
neighbor 192.168.1.1 remote-as 1
! Опция next-hop-self  необходима при правильной установки атрибута NEXT_HOP
! при внутренних сетей соседями из других AS
neighbor 192.168.1.1 next-hop-self
! сессии с маршрутизаторами из своей AS
neighbor 192.168.2.2 remote-as 10
neighbor 192.168.3.2 remote-as 10
```

R2:

```
router bgp 10
no synchronization
network 172.16.2.0 mask 255.255.255.0
neighbor 192.168.2.1 remote-as 10
neighbor 192.168.4.1 remote-as 10
```

R3:

```
router bgp 10
no synchronization
network 172.16.3.0 mask 255.255.255.0
neighbor 192.168.3.1 remote-as 10
neighbor 192.168.4.2 remote-as 10
```

Стоит отметить, что каждый раз при изменении настроек соседей необходимо произвести сброс BGP-сессии. Делается это командой:

clear ip bgp ip_соседа

При этом процедура создания bgp сессии проходит заново. Если хотите сбросить все - вместо ip подставьте * .

Если изменения настроек соседей производятся "на живом человеке", то при изменении атрибутов маршрутов, коммунити, route-map, advertise-map, prefix-list, distribute-list и т.п., можно использовать "мягкий сброс" сессии, как входящей, так и исходящей:

```
- clear ip bgp <ip соседа> soft out для случаев изменения наших анонсов
- clear ip bgp <ip соседа> soft in для случаев изменения нашей политики обработки маршрутов от соседа
```

Route-map, prefix-list

Как Вам несомненно известно, в протоколе BGP множество атрибутов, причем несущих самую разную информацию. Ими надо как-то управлять, причем управлять осмысленно. Одним из главных инструментов, служащих данной цели, являются так называемые "карты маршрутов" (route-map). С помощью них можно с высокой точностью управлять как входящими, так и исходящими атрибутами и анонсами протокола BGP.

Формат применение карты маршрутов к соседу:

```
neighbor x.x.x.x route-map имя_карты направление
```

имя_карты - это собственно имя карты маршрутов, под которым она фигурирует в конфигурации. Чаще всего дают названия, несущие некий смысловой характер. К примеру названием AS1234-in очень удобно назвать карту маршрутов, которая работает с входящими анонсами от AS1234.

направление - эта директива указывает на направление, в отношении которого примется карта маршрутов. Соответственно, in - карта маршрутов применяется к анонсами приходящим от данного соседа, out - карта маршрутов применяется к анонсами уходящим к данному соседу.

Непосредственно формат "карты маршрутов":

route-map название действие номер1

```
match критерий1
...
match критерийN
set действие1
...
set действиеN
```

route-map название действие номер2

```
match критерий1
...
match критерийN
set действие1
...
set действиеN
```

название - это опять же имя карты. действие - это действие, которое производится при совпадении анонса с критерием (или критериями) match. Может принимать значение permit (разрешить прохождение анонса) или deny (запретить прохождение анонса). В том случае, если анонс совпадает с критериями текущего номера карты маршрутов, его дальнейшая обработка не производится. Действие по умолчанию - deny. номер - это номер правила внутри карты, который регулирует порядок прохождения. То есть, иначе говоря, применение правил идет от правила с меньшей цифрой по возрастанию. Теперь давайте разберемся с критериями и действиями. В качестве критерия может выступать выражение, которое указывает какой атрибут анализировать и каким образом он должен "выглядеть".

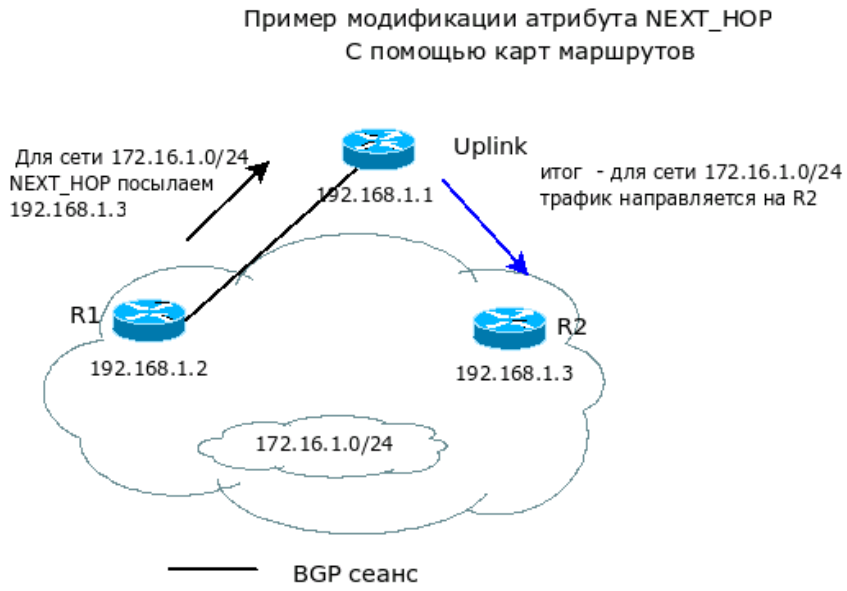
К примеру, Вам необходимо "отловить" анонсы сети 192.168.0.0/23. Реализуется это следующим образом:

! сам критерий match ip address 1 ! ACL coот. данному критерию access-list 1 permit 192.168.0.0 0.0.1.255

Что касается действий, то эти директивы направлены на работу с атрибутами. В зависимости от характера атрибута они могут его удалять, добавлять или модифицировать. В последующих разделах мы будем более плотно работать с критериями и действиями.

А пока для иллюстрации карт маршрутов рассмотрим решение простенькой задачки: Есть у Вас сеть 172.16.1.0/24. Есть маршрутизаторы R1 и R2, и Вам необходимо чтобы трафик, связанный с этой сетью проходил через R2 (не "прожует" R1 трафика к этой сети). Пикантность ситуации добавляет тот факт, что R2 не понимает BGP (ну например, не роутер это, а L3-коммутатор без

BGP). Ну с исходящим трафиком все просто: R2 - шлюз по умолчанию для сети 172.16.1.0/24, а для R2 шлюз по умолчанию некий вышестоящий роутер. А что-же делать с входящим трафиком? Можно ли неким образом рассказать BGP-соседям, что сеть то живет за R2 ? Можно. Для этого необходимо модифицировать атрибут NEXT_HOP таким образом, чтобы он указывал на R2. Для этого на R1 делается:



router bgp 10 no synchronization network 172.16.1.0 mask 255.255.255.0 neighbor 192.168.1.1 remote-as 1 neighbor 192.168.1.1 next-hop-self neighbor 192.168.1.1 route-map NET-TO-R2 out

access-list 10 permit 172.16.1.0 0.0.0.255

route-map NET-TO-R2 permit 10 match ip address 10 ! ключевое действие - объясняем bgp соседу, что трафик к сети 172.16.1.0 должен ! пройти через R2 set ip next-hop 192.168.1.3

! пропускаем другие анонсы без модификации route-map NET-TO-R2 permit 20

«Отлично», - скажете Вы, вот мы можем неплохо фильтровать анонсы, но ведь неудобно же работать с префиксами сетей. И будете правы - неудобно. Вот только мало кто работает с префиксами сетей с помощью карт маршрутов. Для этого существует специальный инструмент - списки префиксов (на "мове" prefix-list). С помощью данного инструмента можно изгаляться надо префиксами как Вам на душу положит. Причем как с входящими, так и исходящими.

Формат применение префиксов к соседу аналогичен карте маршрутов:

neighbor x.x.x.x prefix-list имя_префикс_листа направление

Думается мне, тут расшифровки не требуется. Само же описание списка префиксов выглядит следующим образом:

ip prefix-list имя_префикс_листа seq номер_правила_внутри_листа действие сеть ge начало_диапазона le конец_диапазона

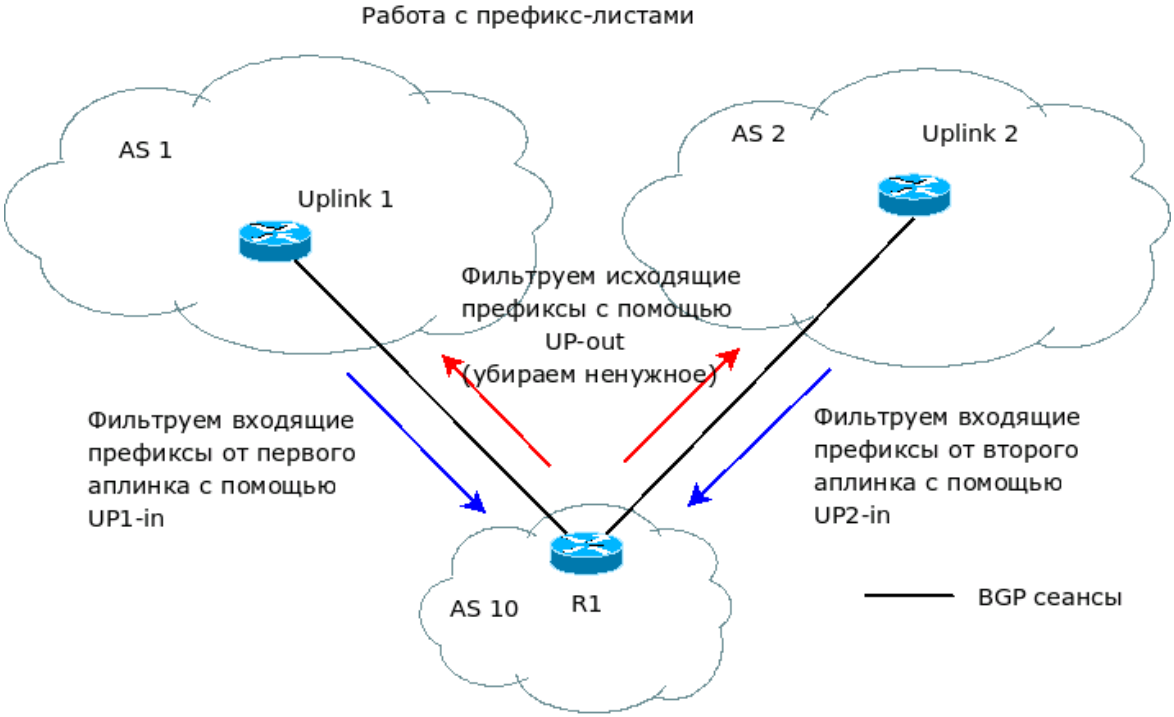
С именем все понятно, разберемся с номером. В списке может быть несколько правил, и номер определяет какое правило будет применено ранее. Как и в картах маршрутов, правила с меньшими номерами будут "пройдены" раньше. Если при прохождении найдено совпадение - следующие правила не рассматриваются. Действие – те же сакраментальные permit или deny. Соответственно permit разрешает прохождение префикса, а deny - запрещает. Как всегда по умолчанию - запрет.

В качестве параметра сеть выступает интересующий префикс. Параметры "ge начало_диапазона" и "le конец_диапазона" являются необязательными и указывают на, соответственно, начало и конец диапазона масок префиксов. То есть ge NN срабатывает на префиксах, маска которых более NN, а le NN срабатывает на префиксах у которых маска меньше NN.

Звучит слегка диковато и для понимания нужен пример. Есть 2 аплинка .От аплинков мы

должны получить префиксы, маска которых меньше или равна 23 (для экономии памяти маршрутизатора, причем сети аплинков принимаем без "обрезания") и не принимать маршруты по умолчанию (предположим они нам отдают full-view), а им анонсировать свои сети , но не мельче чем /23, и не анонсировать свой маршрут по умолчанию и свои серые сети.

- Сеть аплинка 1 x1.y1.z1.w1/8
- Сеть аплинка 2 x2.y2.z2.w2/10




```
router bgp 10
no synchronization
neighbor 1.2.3.1 remote-as 1
neighbor 1.2.3.1 next-hop-self
neighbor 1.2.3.1 prefix-list UP1-in in
neighbor 1.2.3.1 prefix-list UP-out out
neighbor 1.1.3.1 remote-as 2
neighbor 1.1.3.1 next-hop-self
neighbor 1.1.3.1 prefix-list UP2-in in
neighbor 1.1.3.1 prefix-list UP-out out

! запрещаем анонсирование маршрута по умолчанию
ip prefix-list UP-out seq 1 deny 0.0.0.0/0
! убираем серые сети
ip prefix-list UP-out seq 5 deny 10.0.0.0/8 le 32
ip prefix-list UP-out seq 10 deny 172.16.0.0/12 le 32
ip prefix-list UP-out seq 15 deny 192.168.0.0/16 le 32
! Разрешаем все префиксы /23 и менее (то есть /23,22 и др будет пропущен,
! в то-же время /24 пойдет лесом).
ip prefix-list UP-out seq 20 permit 0.0.0.0/0 le 23

! алинк1
! запрещаем прием маршрута по умолчанию
ip prefix-list UP1-in seq 1 deny 0.0.0.0/0
! пропускаем сеть первого аплинка без "обрезания"
ip prefix-list UP1-in seq 10 permit x1.y1.z1.w1/8 ge 9
! Разрешаем все префиксы /23 и менее
ip prefix-list UP1-in seq 20 permit 0.0.0.0/0 le 23

! алинк2
! запрещаем прием маршрута по умолчанию
ip prefix-list UP2-in seq 1 deny 0.0.0.0/0
! пропускаем сеть второго аплинка без "обрезания"
ip prefix-list UP2-in seq 10 permit x2.y2.z2.w2/10 ge 11
! Разрешаем все префиксы /23 и менее
ip prefix-list UP2-in seq 20 permit 0.0.0.0/0 le 23
```

LOCAL PREFERENCE - управляем исходящим трафиком

После того, как мы изучили некоторые практические основы работы протокола BGP, настроили пиринг и даже вроде бы все заработало, пришло время перевести дух и немного поразмышлять. А думать есть о чем - адреса то мы получили, с аплинками /IX местечкового масштаба пиринг подняли. И вроде все хорошо. Кроме одного - трафик "ходит" совсем не так, как нам хотелось бы. Естественно, что такая практика совершенно неприемлема. Приступим к процессу приведения путей движения трафика к нужному нам виду. И начнем мы с самого простого - исходящего трафика. Для управления исходящим трафиком предназначен атрибут LOCAL_PREFERENCE. LOCAL_PREFERENCE - это атрибут, представляющий собой число из диапазона от 1 до 32768, которое характеризует степень предпочтения анонса. Чем больше LOCAL_PREFERENCE - тем более предпочтительным является анонс. То есть если, к примеру, у Вас 2 анонса, лучшим будет выбран тот, у кого LOCAL_PREFERENCE больше. Атрибут LOCAL_PREFERENCE передается между iBGP пирами, то есть является нетранзитивным.

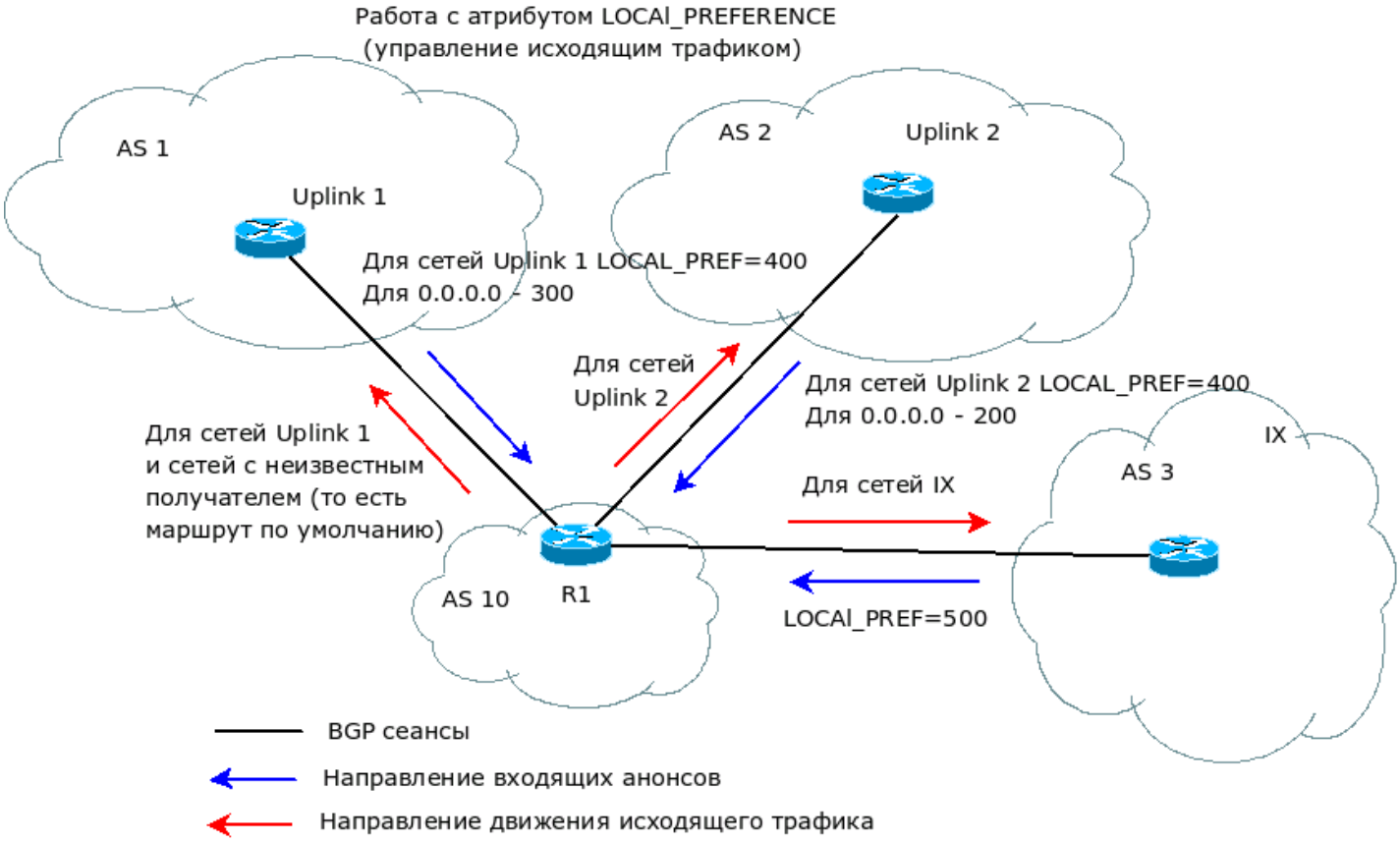
Для иллюстрации того, каким образом с ним работать, приведу пример: У нас есть 2 аплинка , которые анонсируют нам некие свои сети и маршрут по умолчанию, и мы включены в один паритет. Задача:

- трафик для сетей паритета должен уходить через паритет.
- трафик для сетей аплинков должен уходить через соответствующего аплинка.
- в качестве провайдера по умолчанию должен использоваться 1-й провайдер.

Для реализации данной задачи в отношении исходящего трафика необходимо выполнить следующие операции с анонсами:

- анонсы из IX - большой LOCAL_PREF, к примеру 500
- анонсы сетей из провайдеров - LOCAL_PREF 400
- анонс сети по умолчанию из аплинка 1 - LOCAL_PREF 300
- анонс сети по умолчанию из аплинка 2 - LOCAL_PREF 200

Реализация на практике:




```
router bgp 10
no synchronization
! описываем всех соседей и соот. привязываем фильтры
neighbor 1.1.1.1 remote-as 1
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.1 route-map UP1-IN in
neighbor 1.2.1.1 remote-as 2
neighbor 1.2.1.1 next-hop-self
```

```
neighbor 1.2.1.1 route-map UP2-IN in
neighbor 1.3.1.1 remote-as 3
neighbor 1.3.1.1 next-hop-self
neighbor 1.3.1.1 route-map IX-in in

access-list 1 permit 0.0.0.0 0.0.0.0

! устанавливаем для всех анонсов из IX LOCAl_PREF 500
route-map IX-in permit 10
set local-preference 500

! устанавливаем для маршрута по умолчанию из UP 1  LOCAl_PREF 300
route-map UP1-IN permit 10
match ip address 1
set local-preference 300

! для сетей из UP 1  LOCAl_PREF 400
route-map UP1-IN permit 20
set local-preference 400

! устанавливаем для маршрута по умолчанию из UP 2  LOCAl_PREF 200
route-map UP2-IN permit 10
match ip address 1
set local-preference 200

! для сетей из UP 2  LOCAl_PREF 400
route-map UP2-IN permit 20
set local-preference 400
```

AS PATH - управление входящим трафиком а также фильтрация

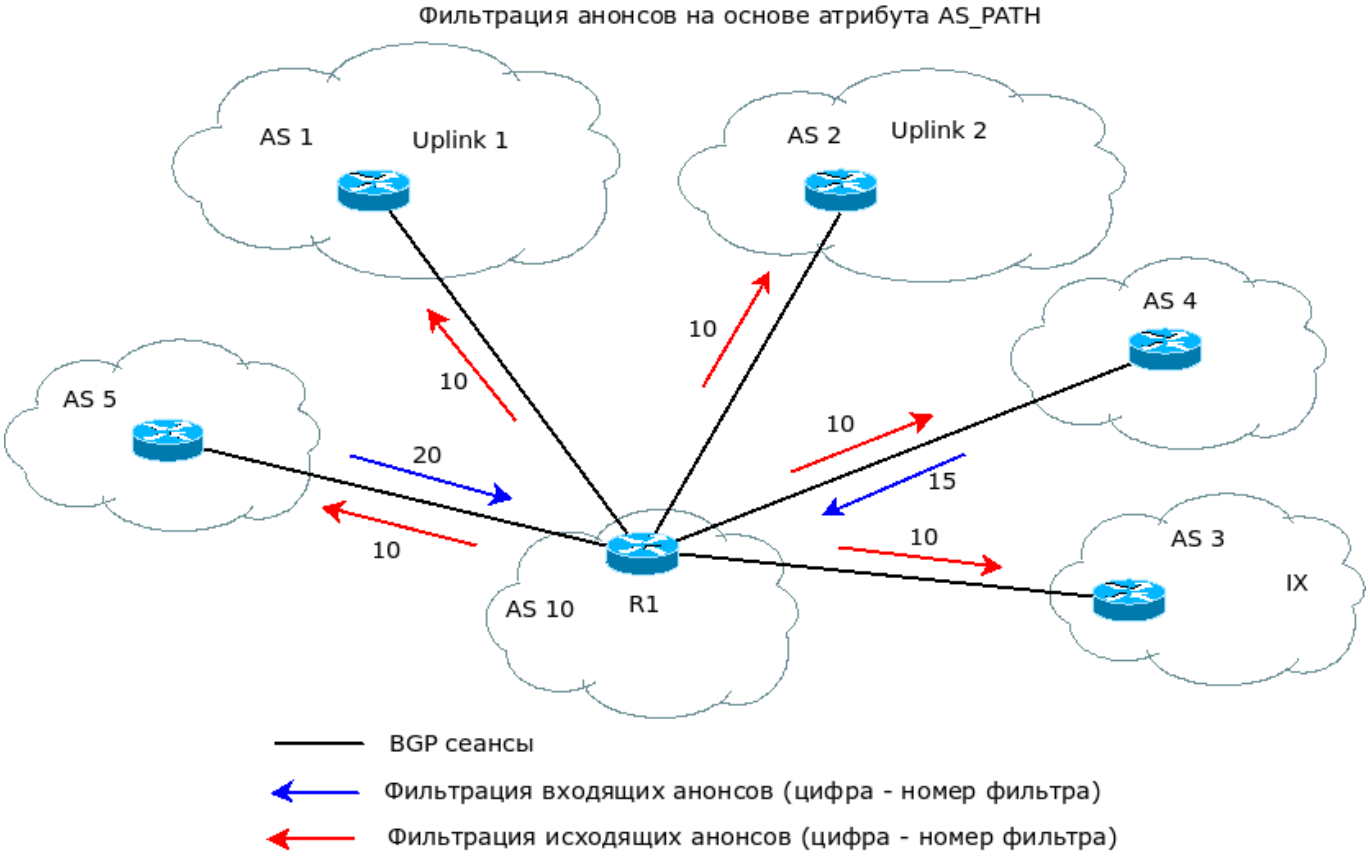
Разобравшись с исходящим трафиком, надо заняться входящим. Вопрос более чем серьезный, потому, как чаще всего, входящий трафик в несколько раз больше исходящего. Однако просто так, "в лоб", указать десяткам тысяч провайдеров Интернета что Вам хочется получать вход так, а не иначе - невозможно.

Придется идти в обход и вспоминать атрибуты BGP. Как Вы несомненно знаете, одним из ключевых атрибутов, который используется при работе с анонсами, является атрибут AS_PATH, представляющий собой список AS, которые необходимо пересечь для достижения искомой сети. То есть, иначе говоря, чем короче AS_PATH, тем более предпочтителен маршрут. А чем длиннее - тем соответсвенно и "хуже". А раз так, спросите Вы, то нельзя ли неким образом, удлинить тот маршрут, который для Вас менее предпочтителен? Можно, и даже более того, в ряде случаев даже нужно. То, как это сделать будет показано чуть ниже. Кроме того, атрибут AS_PATH позволяет строить простые и удобные фильтры для фильтрации анонсов, главное удобство которых в том, что можно не строить больших выражений и учитывать множество префиксов.

Для начала приведем пример фильтрации анонсов с помощью AS_PATH. К примеру, Ваша AS соединена с несколким аплинками (AS1 и AS2), включена в некий IX (AS3) и имеет несколько точек (AS5 и AS5) пиринга с соседями. Вам необходимо:

1. Не допустить что-бы Ваша AS стала транзитной для вышестоящих провайдеров (иначе говоря что-бы ни при каких условиях трафик между AS1 и AS2 не попытался пройти через Вашу AS)
2. Анонсировать в точку обмена трафика и своим соседям только свою AS.
3. Принимать от соседей только их их собственные сети.

Реализация:




```
router bgp 10
no synchronization
! описываем всех соседей и соот. привязываем фильтры
neighbor 1.1.1.1 remote-as 1
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.1 filter-list 10 out
neighbor 1.2.1.1 remote-as 2
neighbor 1.2.1.1 next-hop-self
neighbor 1.2.1.1 filter-list 10 out
neighbor 1.3.1.1 remote-as 3
neighbor 1.3.1.1 next-hop-self
neighbor 1.3.1.1 filter-list 10 out
neighbor 1.4.1.1 remote-as 4
neighbor 1.4.1.1 next-hop-self
neighbor 1.4.1.1 filter-list 10 out
neighbor 1.4.1.1 filter-list 15 in
neighbor 1.5.1.1 remote-as 5
neighbor 1.5.1.1 next-hop-self
neighbor 1.5.1.1 filter-list 10 out
neighbor 1.5.1.1 filter-list 20 in

! Самое главное - сами фильтры
ip as-path access-list 10 permit ^$
ip as-path access-list 15 permit ^4$
ip as-path access-list 20 permit ^5$
```

Принцип построения фильтров относительно прост и главный момент заключается в регулярном выражении, которое определяет какие as_path мы принимаем, а какие нет. Регулярное выражение предоставляет собой смесь служебных символов и цифр. Символы, используемые в регулярных выражениях:

. Все символы ***** Все выражения **+** 1 и более выражений **?** 0 и более выражений **^** Начало строки а также в качестве символа инвертирования внутри диапазона **$** Конец строки **_** Соответствует точке(.), фигурным и обычным скобкам ({} и ()), начало и конец строки а также пробел **[диапазон]** Определяет диапазон символов в выражении **-** разделяет конечные точки диапазона

Для того, чтобы не внося изменения в настройки оборудования посмотреть какие анонсы подпадают под созданное Вами регулярное выражение, можно воспользоваться командой: sh ip bgp regexp выражение

Фильтры атрибута AS_PATH на основе регулярных выражений можно также использовать и в

картах маршрутов. Та же самая задача, реализованная с применением карт маршрутов (пояснительного рисунка, мне думается, не требуется - принцип абсолютно аналогичен приведенному выше примеру):

```
router bgp 10
no synchronization
! описываем всех соседей и соот. привязываем фильтры
neighbor 1.1.1.1 remote-as 1
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.1 route-map PEER-out out
neighbor 1.2.1.1 remote-as 2
neighbor 1.2.1.1 next-hop-self
neighbor 1.2.1.1 route-map PEER-out out
neighbor 1.3.1.1 remote-as 3
neighbor 1.3.1.1 next-hop-self
neighbor 1.3.1.1 route-map PEER-out out
neighbor 1.4.1.1 remote-as 4
neighbor 1.4.1.1 next-hop-self
neighbor 1.4.1.1 route-map PEER-out out
neighbor 1.4.1.1 route-map PEER4-in in
neighbor 1.5.1.1 remote-as 5
neighbor 1.5.1.1 next-hop-self
neighbor 1.5.1.1 route-map PEER-out out
neighbor 1.5.1.1 route-map PEER5-in in

! Самое главное - сами фильтры
ip as-path access-list 10 permit ^$
ip as-path access-list 15 permit ^4$
ip as-path access-list 20 permit ^5$

route-map PEER-out permit 10
match as-path 10
route-map PEER4-in permit 10
match as-path 15
route-map PEER5-in permit 10
match as-path 20
```

После того, как освоены принципы фильтрации, перейдем к главному - управлению входящим трафиком на основе принципа искусственного удлинения AS_PATH.

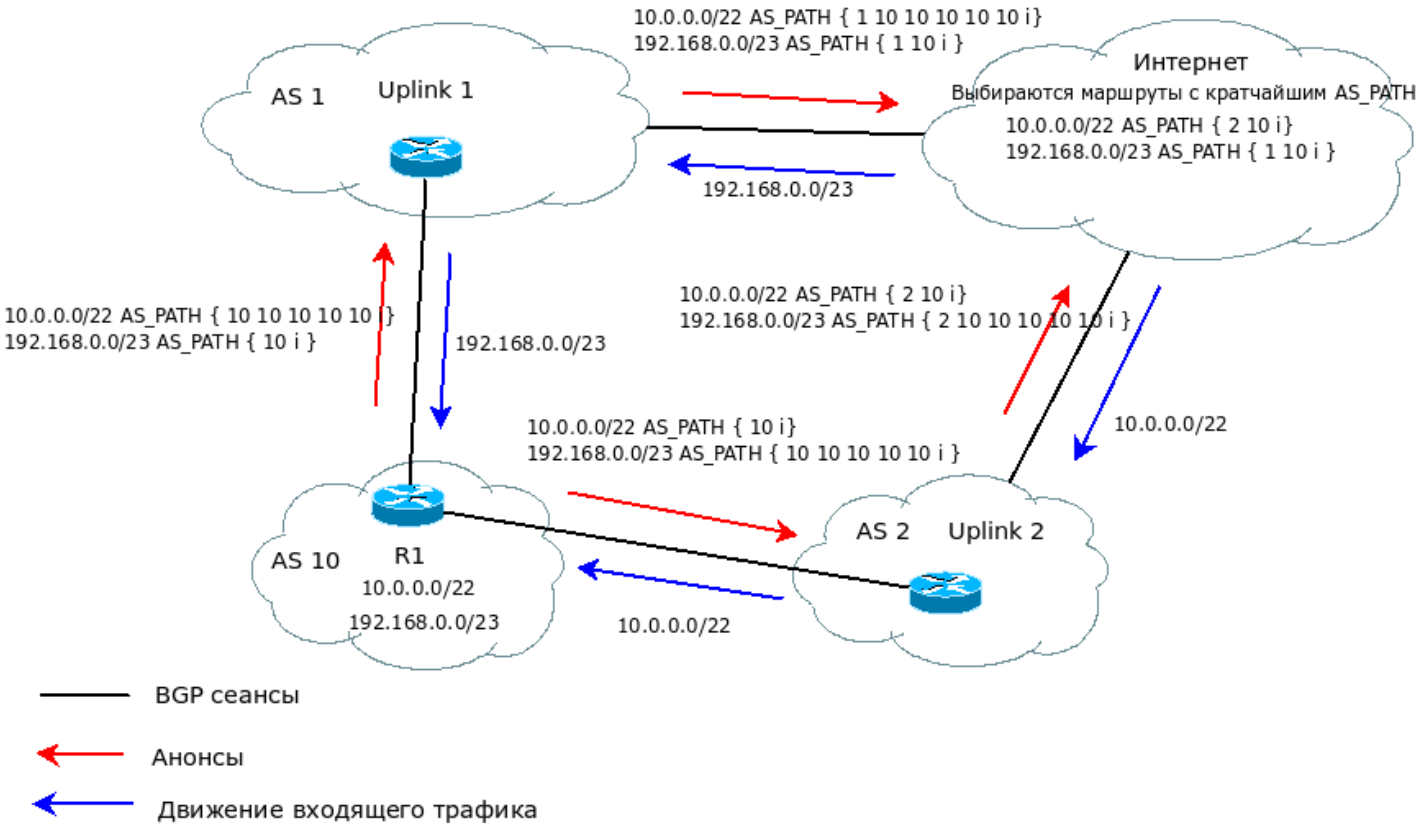
К примеру, Ваша AS (10), которая соединена с нескольким аплинками (AS1 и AS2). У аплинка 1 канал большой пропускной способности довольно надежен, но оплата по трафику, а у второго аплинка канал не всегда надежен, зато трафик не считается, а продается полосой.

У Вас есть 2 сети:

- 10.0.0.0/22 - обычные клиенты
- 192.168.0.0/23 - VIP клиенты.

Вы хотите, что-бы в стандартной ситуации клиенты получали входящий трафик через провайдера 2, а VIP клиенты через провайдера 1. Решается это следующим образом:

Управление движением входящего трафика с помощью атрибута AS_PATH




```
router bgp 10
no synchronization
network 10.0.0.0 mask 255.255.252.0
network 192.168.0.0 mask 255.255.254.0
neighbor 1.1.1.1 remote-as 1
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.1 route-map GW1-out out
neighbor 1.2.1.1 remote-as 2
neighbor 1.2.1.1 next-hop-self
neighbor 1.2.1.1 route-map GW2-out out

access-list 10 permit 10.0.0.0 0.0.3.255
access-list 20 permit 192.168.0.0 0.0.1.255

route-map GW1-out permit 10
match ip address 10
! для соот. сети удлиняем AS_PATH
set as-path prepend 10 10 10 10
route-map GW1-out permit 20

route-map GW2-out permit 10
match ip address 20
! для соот. сети удлиняем AS_PATH
set as-path prepend 10 10 10 10
route-map GW2-out permit 20
```

COMMUNITY

Итак BGP настроен как надо, и все довольны тем, как хорошо все работает. Но счастье вечным не бывает - добавляются новые паритеты и аплинки, появляются клиенты со своими AS, Вы расширяетесь и получаете новые сети. Вобщем, жизнь бьет ключом и, как это увы часто бывает, по голове. Потому что эти изменения требуют от Вас с каждым днем все большего напряжения памяти, так как Клиента 1 анонсировать в часть аплинков не надо, клиента 2 вообще надо анонсировать только с большим AS PREPEND, клиент 3 вообще сам хочет управлять тем, как его анонсировать (с препендами или без), часть собственных сетей надо анонсировать через один канал с "препендами" в другой без них. А учитывая "размножение" устройств с поддержкой BGP в сети, процесс усложнения управлением этого хозяйства подобен сходу снежной лавине. Что же касается понимания как ЭТО работает (читай сопровождения) - картина грустная.

И в один далеко не прекрасный день, Вы понимаете что "надо уже что-то решать" для того, чтобы максимально упростить и автоматизировать процесс управления анонсами.

Первое, что приходит на ум, поступить с анонсами так, как поступают с пакетами при работе с QoS - промаркировать, а затем работать только с маркерами (или метками) . Но можно ли таким же образом поступить с анонсами сетей в протоколе BGP ? Можно. Осуществляется сие действо с помощью применения специального атрибута, именуемого COMMUNITY. COMMUNITY - это необязательный транзитивный атрибут. Он имеет переменную длину и состоит из совокупности четырех байтовых значений из диапазона 0x00000000 до 0x0000FFFF (разве не по 0xFFFF0000???) (диапазон с 0xFFFF0000 по 0xFFFFFFFF зарезервирован для спец. целей).

По умолчанию BGP не посылает и не принимает данный атрибут. Для того, чтобы включить возможность посылки и приема данного атрибута следует указать в свойствах нейбора send-community. К примеру:

```
neighbor 1.2.3.4 send-community
```

После того, как мы для всего оборудования включили поддержку community, нам стоит извлечь из этого некую пользу, то есть сперва неким образом пометить анонсы, а затем ими управлять на основе именно этих меток. Но перед тем, как что-то метить или чем-то управлять, давайте разберемся в том, как "выглядит" на практике атрибут community.

На практике community принято использовать в виде: XX:YY где XX - это номер AS, а YY - это 2-х байтное число, которое вы можете назначать по своему усмотрению. К примеру, значение community равное 1234:1 традиционно означает, что в AS1234 имеется группа анонсов и индексом 1 или как говорят "сообщество" N в AS 1234 которому соответствует значения community 1234:1.

Кроме того, следует также упомянуть, что анонс может иметь несколько community, что сильно увеличивает гибкость управления. Вы же (в роли администратора AS и ее маршрутизаторов) можете, добавлять, удалять, модифицировать или назначать заново атрибут community. После данных необходимых пояснений переходим к процедуре установки меток на анонсы.

Рассмотрим пример установки метки 1234:3456 на анонсы клиента (AS 3456).

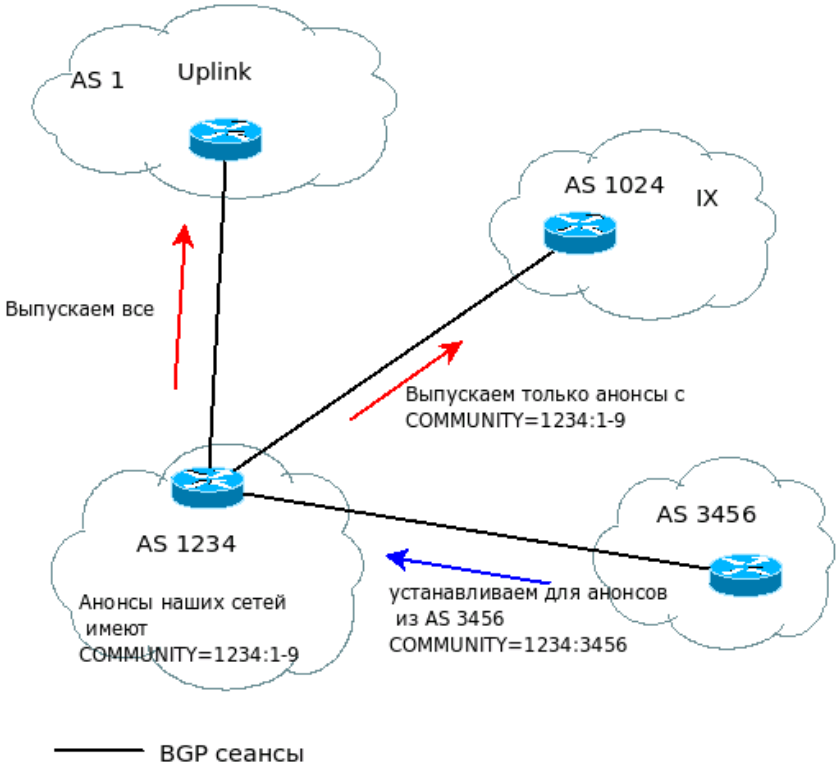
```
router bgp 1234
no synchronization
neighbor 2.3.4.1 remote-as 3456
neighbor 2.3.4.1 send-community
neighbor 2.3.4.1 next-hop-self
neighbor 2.3.4.1 route-map AS3456-In in

! Включаем поддержку нового формата  community
ip bgp-community new-format

route-map AS3456-In permit 10
set community 1234:3456
```

Таким образом, все анонсы клиента полученные от нейбора 2.3.4.1 получат метку. Но это пока малоинтересно, просто потому, что этой меткой надо воспользоваться, то есть на ее основе произвести некие действия. Для примера возьмем ситуацию, когда Вы входите в некий IX, анонсировать в который своих клиентов строго запрещено. Ваши анонсы имеют метку 1234:1-1234:9 Исходя из данного соображения мы строим исходящие фильтры:

Фильтрация анонсов на основе атрибута COMMUNITY
(Пример 1)




```
router bgp 1234
no synchronization
neighbor 2.1.4.1 remote-as 1024
neighbor 2.1.4.1 send-community
neighbor 2.1.4.1 next-hop-self
neighbor 2.1.4.1 route-map AS1024-Out out

ip bgp-community new-format

route-map AS1024-Out permit 10
match community MY_NET

ip community-list expanded  MY_NET permit 1234:[1-9]
```

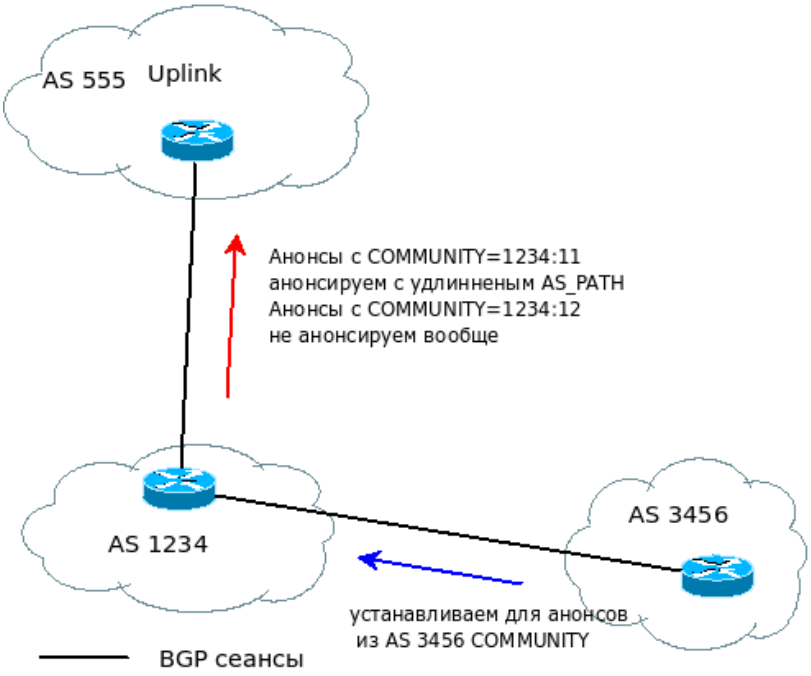
Ключевым в данном случае является ip community-list. Данный лист имеет следующий формат: ip community-list {standard | expanded} Имя листа {permit | deny} { номер community | регулярное выражение } Регулярное выражение используется в листе расширенного формата (expanded). Формат регулярного выражения соответствует традициям компании Cisco. Стандартным действием является deny. К примеру рассмотрим пример такого рода листов:

```
ip community-list expanded TEST deny _1234:10_
ip community-list expanded TEST permit _1234:.*_
```

Данный лист запрещает прохождение анонсов, в которых содержится community 1234:10, но разрешает прохождение анонсов с community, в которых содержится 1234:XX.

А на закуску я проиллюстрирую пример того, каким образом можно дать возможность клиенту автоматически управлять своими анонсами, то есть чтобы сам клиент решал анонсировать его нормально, с препендами, или вообще не анонсировать. Для этого Вам необходимо разработать "политику сообществ", то есть указать какие community для чего служат. К примеру:

Фильтрация анонсов на основе атрибута COMMUNITY
(Пример 2)




```
1234:10 - анонсы с таким community нормально  анонсируются всюду (кроме есст. "закрытых" IX)
1234:11 - анонсы с таким community анонсируются аплинком с препендом
1234:12 - анонсы с таким community не анонсируются аплинкам.
```

И в зависимости от того, какой community имеют анонсы клиента, Вы и соответствующим образом работаете с этими анонсами.

Практически это будет выглядеть следующим образом: Сессия с клиентом:

```
router bgp 1234
no synchronization
neighbor 2.3.4.1 remote-as 3456
neighbor 2.3.4.1 send-community
neighbor 2.3.4.1 next-hop-self
neighbor 2.3.4.1 route-map AS3456-In in

ip bgp-community new-format

ip community-list expanded NORMAL permit  _1234:10_
ip community-list expanded PREPEND permit  _1234:11_
ip community-list expanded INTERNAL permit  _1234:12_

route-map AS3456-In permit 10
match community NORMAL
! переназначение  community необходима для того, что-бы клиент не злоупотреблял
! доверием и не начал пытаться вложить неположенные community
set community 1234:10

route-map AS3456-In permit 15
match community PREPEND
set community 1234:11

route-map AS3456-In permit 20
match community INTERNAL
set community 1234:12

route-map AS3456-In permit 25
! помечаем все остальное стандартным community
set community 1234:10
```

Сессия с аплинком:

```
router bgp 1234
no synchronization
neighbor 2.1.1.1 remote-as 555
neighbor 2.1.1.1 next-hop-self
neighbor 2.1.1.1 send-community
neighbor 2.1.1.1 route-map AS555-Out out
```

```
ip bgp-community new-format

ip community-list expanded NORMAL permit  _1234:10_
ip community-list expanded PREPEND permit  _1234:11_
ip community-list expanded INTERNAL permit  _1234:12_

route-map AS555-Out permit 10
match community NORMAL
! удаляем из анонсов аплинку наш community
set comm-list NORMAL delete
! добавляем в анонс community, который  указывает аплинку на нормальное
! анонсирование  данных анонсов  (если аплинк разрешает Вам это делать)
set community 555:1 additive

route-map AS555-Out permit 15
match community PREPEND
set as-path prepend 1234 1234 1234 1234
! можно также воспользоваться директивой
!  set as-path prepend last-as 4
! которая добавит в значению AS_PATH 4 раза номер последней AS
set comm-list PREPEND delete
! добавляем в анонс community, который  указывает аплинку на нормальное
! анонсирование  данных анонсов  (если аплинк разрешает Вам это делать)
set community 555:2 additive

route-map AS555-Out deny 20
match community INTERNAL

route-map AS555-Out permit 30
```

При правильном проектировании community становятся незаменимым инструментом по управлению анонсами, который позволяет довольно точно и главное наглядно управлять анонсами BGP.

И в качестве завершающего штриха стоит упомянуть про специальные значения community :

- no-export - анонсы с таким значением community не анонсируются нейборам, чей номер AS отличается от номера AS роутера.
- local-as - анонсы с таким значением community анонсируются нейборам не только из той-же AS, в которой находится роутер, но и анонсируются соседям по конфедерации. За пределы конфедерации анонсы не уходят.
- no-advertise - анонсы с таким значением community не анонсируются вообще никуда.

BGP Conditional Advertisement

Клиенты бывают разные. Разные как в смысле характера, так и в смысле платежеспособности. Вот только если "жирный" клиент хочет услугу на своих условиях, «продажники», увы, сперва подписывают а потом думают. А разбираться приходится нам, технарям. Ну да в сторону лирику - перейдем к сути задачи.

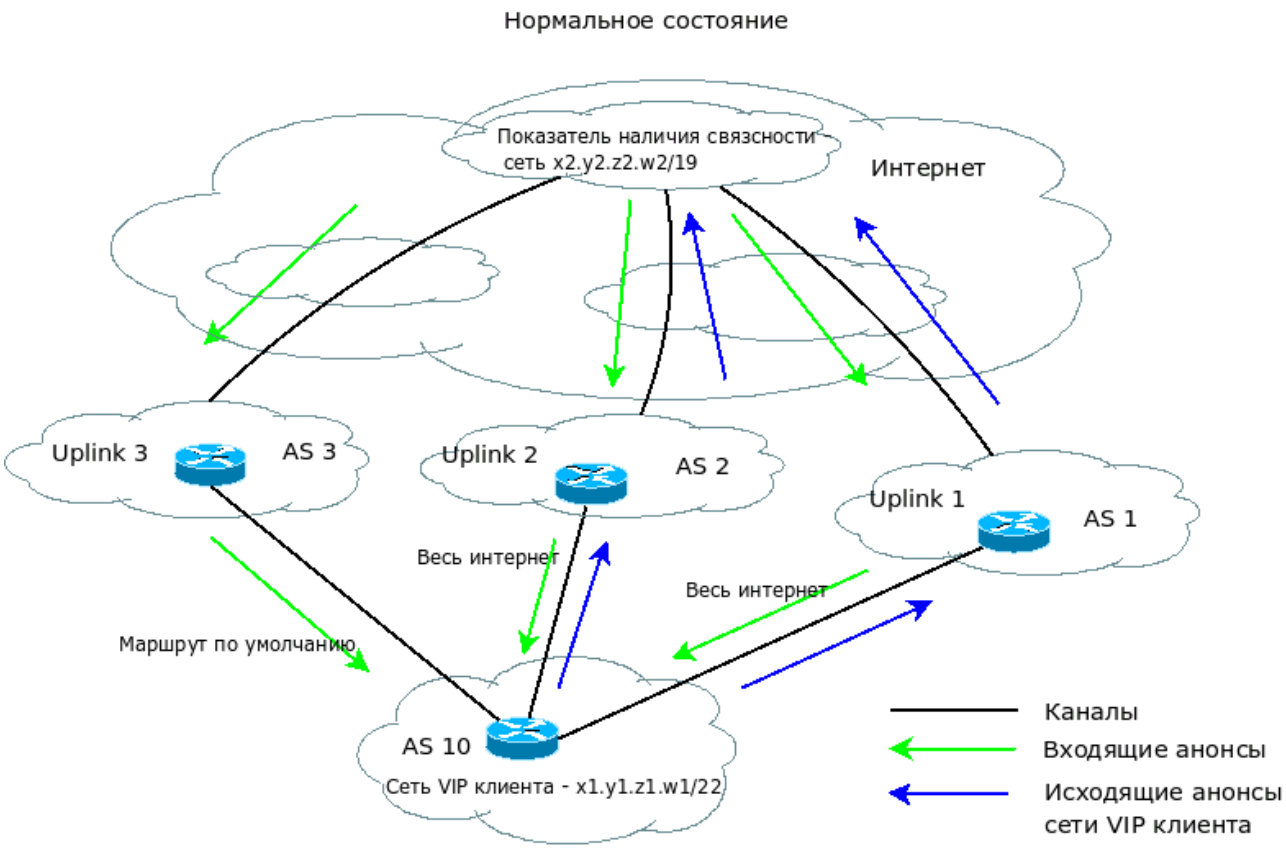
А задача заключается в том, что есть у Вас "жирный" клиент, который покупает большую полосу и платит он (по меркам Вашей конторы) вагон и малую телегу денег. Но хочет этот клиент, чтобы Вы обеспечивали железобетонную связь с сетью. До включения этого клиента у Вас было 2 аплинка и в принципе их надежность Вас устраивала. Однако иногда (редко, но все-же) наступал всеобщая катастрофа - падали оба. Ладно, нашли Вы третий аплинк. Независимый и с хорошим качеством, но цены на трафик заставят покраснеть любого. То есть, иначе говоря, канал от него должен использоваться только сетью VIP клиента, и только в случае если через первые 2 аплинка "умерли".

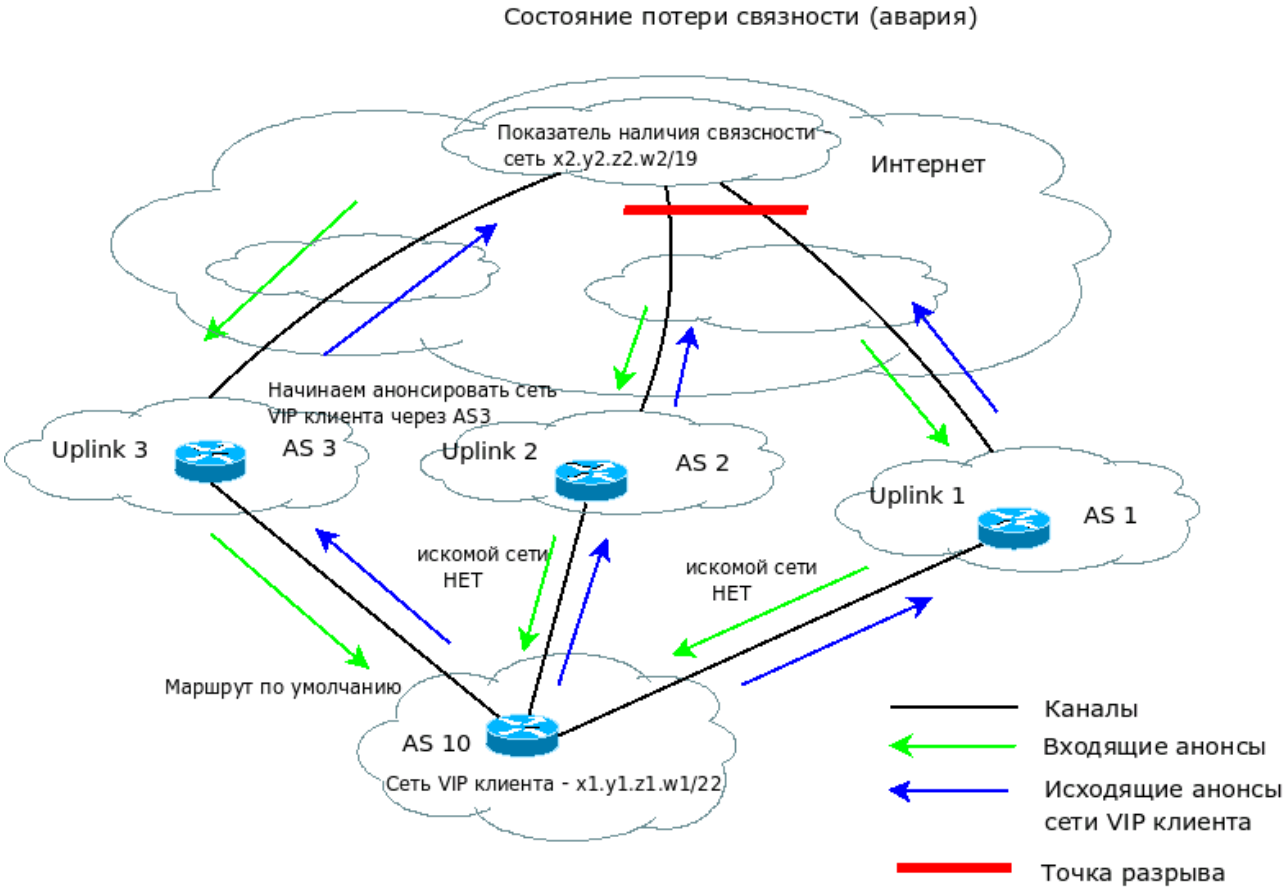
Первое, что приходит в голову - анонсировать в 3-й аплинк только сеть с VIP-клиентом (как минимум /24 , а желательно /23) и очень сильно удлинить AS_PATH для этого префикса. Это сделать конечно можно, но где гарантия что аплинк вообще будет смотреть (Вы для него-то клиент) на Ваш AS_PATH ? Поставит большой LOCAl_PREF для анонсов из Вашей сети и поминай как звали - поползет трафик по дорогому каналу как миленький.

С другой стороны, городить "костыли" типа скриптов, которые смотрят анонсы и, в случае необходимости, меняют настройки сессии с 3-м аплинком, не хочется. Тем более вставать ночью. Как-же сделать все в автоматическом режиме без "костылей" ?

Для решения этой задачи необходимо воспользоваться метода BGP Conditional Advertisement Feature. Суть данного метода состоит в том, что если в таблице маршрутизации нет некой сети,

то мы начинаем анонсировать в третьего аплинка сеть VIP-клиента. От 2-х первых аплинков мы получаем full-view, а от третьего - только маршрут по умолчанию.






```
router bgp 10
! /...кусь../
!
neighbor 3.2.3.1 remote-as 3
! ключевой момент. Если нет совпадений в карте, указанной в директиве non-exist-map
! то в "работу идет" карта из IF-NET
neighbor 3.2.3.1 advertise-map IF-NET non-exist-map IF-NO-NET
```

```
neighbor 3.2.3.1 prefix-list UP3-out out
! /...кусь../

!  сеть клиента
access-list 1 permit x1.y1.z1.w1 0.0.1.255
! некая показательная сеть /19 , при пропадании которой мы анонсируем клиента
! в 3-го аплинка
access-list 2 permit x2.y2.z2.w2 0.0.31.255
!
route-map IF-NO-NET permit 10
match ip address  2
!
route-map IF-NET permit 10
match ip address 1

! ограничиваем анонсы в 3-й аплинк.
ip prefix-list UP3-out seq 1 permit x1.y1.z1.w1/23
```

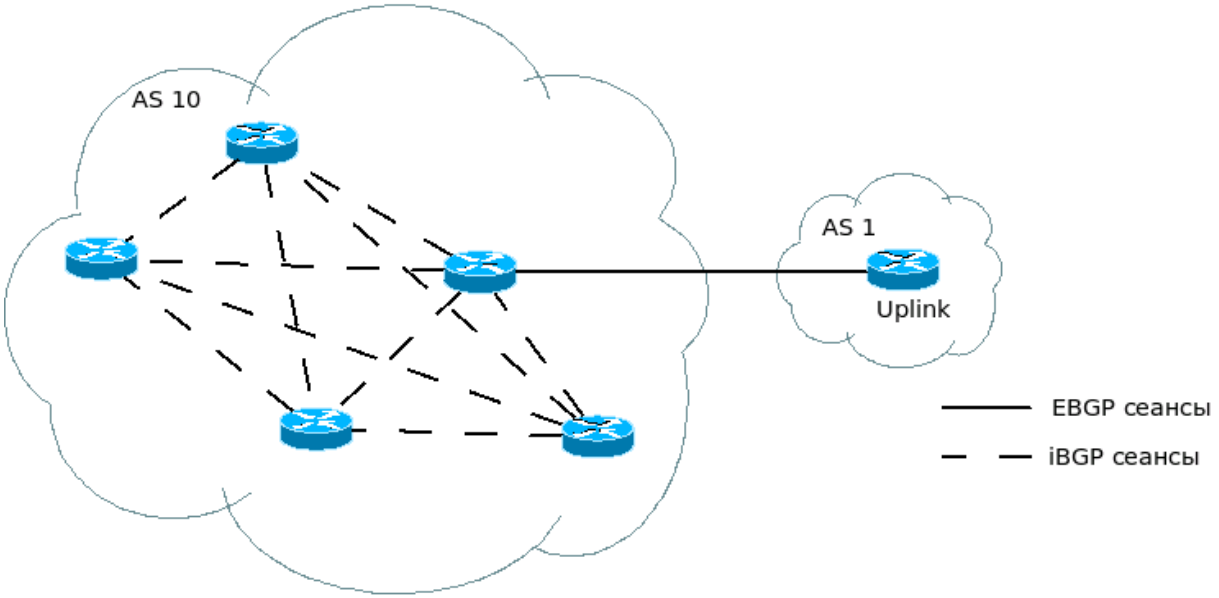
И мы получим, что при проблемах с 2-мя аплинками (например пьяный варяг рубанул общую оптику) сеть с VIP-клиентом будет получать сервис через 3-го аплинка.

Стоит добавить, что на случай аварии необходимо в сторону третьего аплинка разрешить прохождение трафика только от нашего роутера и сети VIP, этот шаг необходим с целью недопущения "засорения" дорогого канала лишним трафиком.

Конфедерации и отражатели маршрутов

Вернемся к вопросу, поднятому в прошлой главе "Счастье вечным не бывает". Растущая с каждым днем сеть вносит в Вашу жизнь некое подсознательное беспокойство - плохой сон, аппетит и др. Все чаще перед Вы говорите себе: "Так жить нельзя". В чем же причина ? А причина в том, что в обычном виде, поскольку внутри Вашей AS приходится на каждом маршрутизаторе поддерживать N-1 (N - число BGP маршрутизаторов в AS) iBGP-сессий, а общее число iBGP сессий составляет N(N-1) очень трудно сопровождать такого рода "конструкцию" равно как и диагностировать проблемы.

Полносвязная топология внутри AS без применения спец. средств



К примеру у Вас в AS есть 20 маршрутизаторов, надо добавить 21-й. Правда грустно заходить на каждый из 20-ти и добавлять? А если где-то промахнешься, забудешь добавить и т.д. - нарушится целость AS, со всеми вытекающими (такими как одна сеть не "видит" другую и др.)

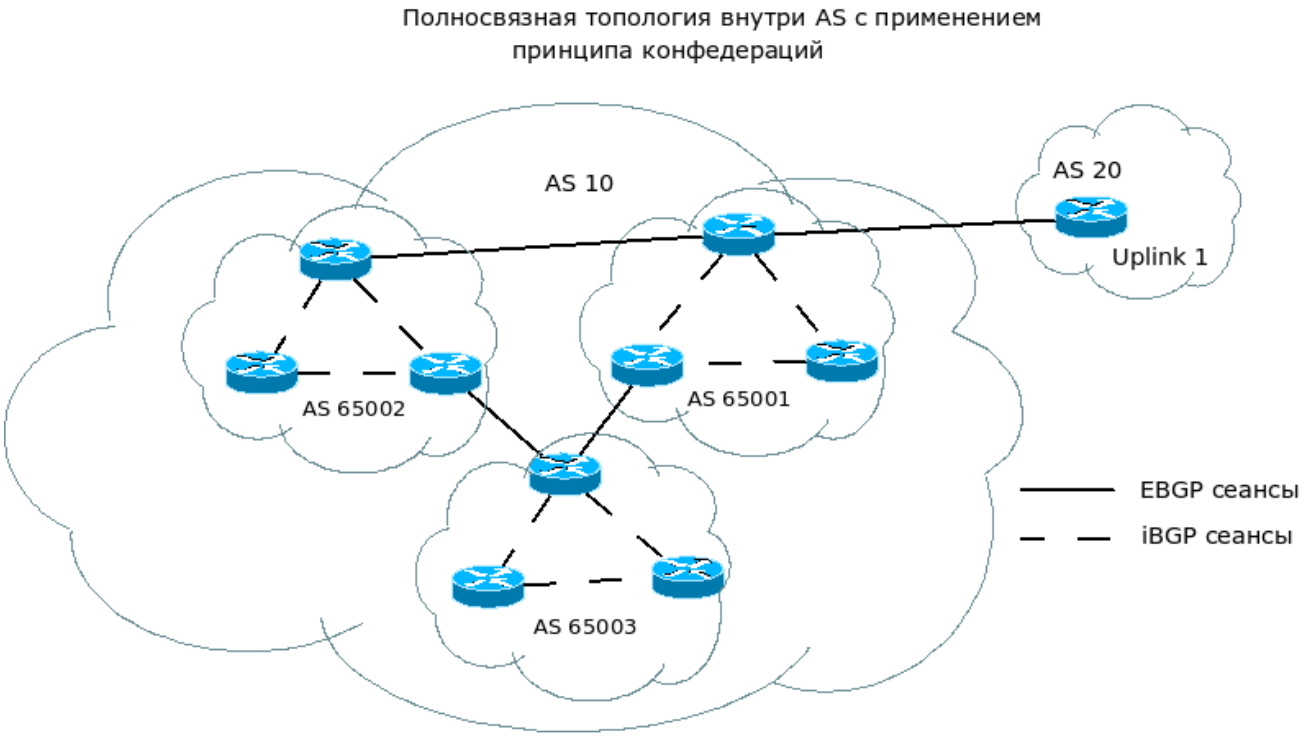
Для решения такого рода проблем существует 2 способа:

1. Конфедерации
2. Отражатели маршрутов.

Для начала рассмотрим концепцию BGP-конфедерации. В основу данной концепции положена идея разбиения большой AS на множество мелких "серых" AS. Взаимодействие между этими "мелкими" AS будет происходить по протоколу EBGP. То есть, к примеру, у нас 20

маршрутизаторов (2 из которых связаны с аплинками, паритетами, то есть с другими AS) в AS 10. Мы разбиваем AS 10 на 3 конфедерации - 65001,65002,65003 взаимодействие между которыми происходит по eBGP.

Примеры настроек маршрутизаторов: R1 - маршрутизатор в AS 65001, который кроме того поддерживает связь с аплинком (AS 20) с маршрутизаторами иных конфедераций и с маршрутизаторами своей конфедерации:




```
router bgp 65001
no synchronization
! наша AS - 10, ей должен соот. номер confederation identifier
bgp confederation identifier 10
bgp confederation peers 65001
! сессия с аплинком
neighbor 1.2.3.1 remote-as 20
neighbor 1.2.3.1 next-hop-self
! ключевой момент - удаляем серые AS из AS_PATH анонсов, отправляемых  в
! сторону соседа с реальной AS
neighbor 1.2.3.1 remove-private-as
! сессия с роутером из другой конфедерации.
neighbor 2.2.3.1 remote-as 65002
neighbor 2.2.3.1 next-hop-self
! сессия с роутером из нашей конфедерации.
neighbor 2.3.9.1 remote-as 65001
```

R2 - роутер внутри серой AS.

```
router bgp 65001
no synchronization
bgp confederation identifier 10
bgp confederation peers 65001
! сессии с роутерами нашей конфедерации
neighbor 2.2.9.2 remote-as 65001

........
neighbor 2.3.9.N remote-as 65001
! если нужно - сессия с роутером из другой конфедерации.
neighbor 2.2.3.2 remote-as 65003
neighbor 2.2.3.2 next-hop-self
```

Таким образом, основным отличием является применение директив bgp confederation identifierб bgp confederation peers и remove-private-as. Директива bgp confederation peers для сохранения нетранзитивных атрибутов (таких как LOCAL_PREF) для передачи через сеанс EBGP внутри сети с конфедерациями. Для роутеров внутри вашей AS, не связанных напрямую с реальными AS директива remove-private-as не применяется.

Как говорится дешево и сердито. Разбив сеть на конфедерации, мы значительно уменьшаем

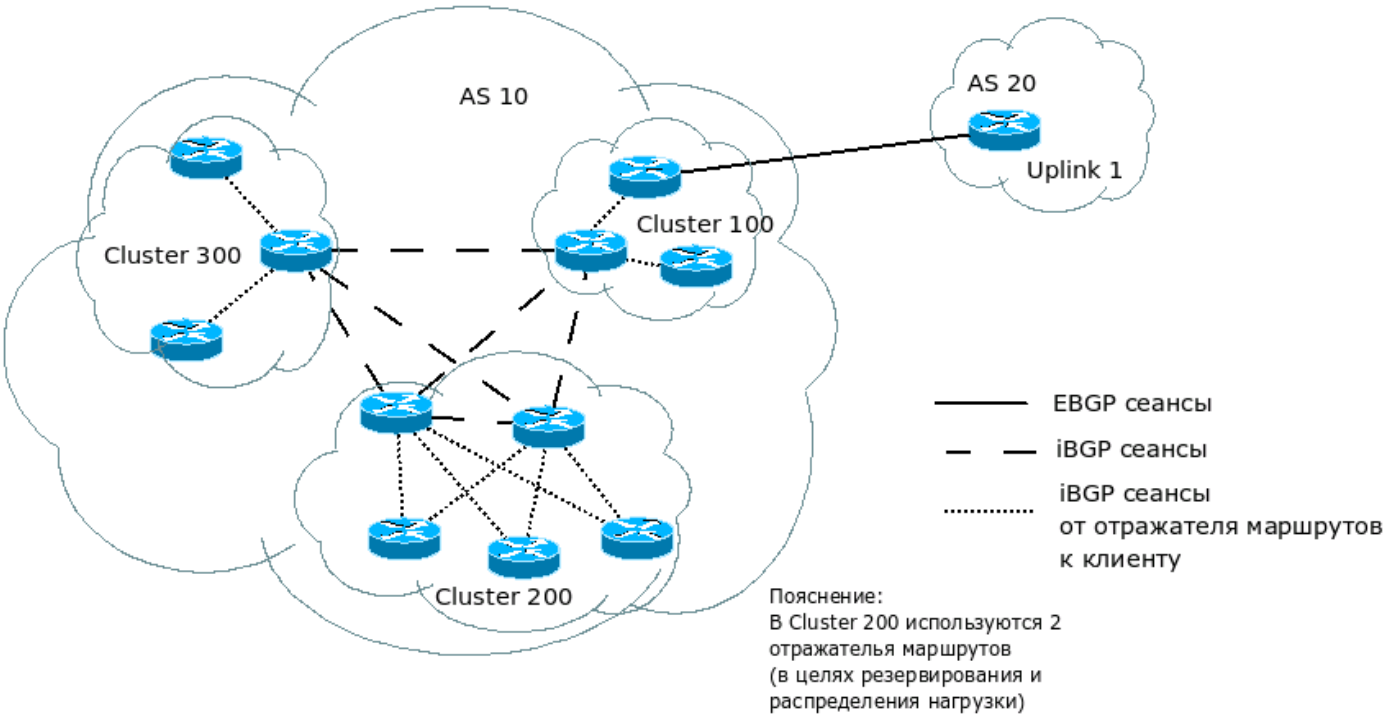
кол-во iBGP-сессий и получаем относительно (того, кто проектировал) понятную структуру. Особый смысл обретает понятие конфедерации в случае приобретения нами провайдера или резким расширение сети. В этом случае соответствующие участки просто включаются как отдельные "серые" AS и нам не приходится морочить голову с их с долгой интеграцией в нашу сеть.

Однако у конфедераций есть довольно существенный недостаток - если уже существует немалая сеть, то относительно трудно перейти к использованию конфедераций.

С другой стороны, кроме концепции конфедерации для решения проблем работы BGP в больших AS существует концепция "отражателей маршрутов". Суть концепции отражателей маршрутов заключается в том, что внутри AS, исходя из физической топологии, выявляются специальные маршрутизаторы (которые и называются отражателями маршрутов), которые получив анонс по протоколу iBGP, не просто заносят его к себе в таблицу маршрутизации а еще и передают их другим маршрутизаторам. Таким образом в сети, где работают отражатели маршрутов обычным маршрутизаторам нужно поддерживать iBGP-сессию только с отражателем маршрутов. Сами же отражатели соответственно поддерживают iBGP-сессии с другими отражателями и клиентами.

Для примера возьмем уже пригодившийся пример с AS. Исходя из топологии мы выбрали 3 отражателя маршрутов, настройка которых будет следующей:

Полносвязная топология внутри AS с применением принципа отражателей маршрутов




```
router bgp 10
no synchronization
/       кусь    /
! ключевой момент - сессия с клиентами
neighbor 2.2.3.1 remote-as 10
neighbor 2.2.3.1 route-reflector-client
.....................
neighbor 2.2.3.N remote-as 10
neighbor 2.2.3.N route-reflector-client
! сессии с другими отражателем маршрутов
! тут - все аналогично обычному iBGP
neighbor 2.5.3.1 remote-as 10
neighbor 2.5.4.1 remote-as 10
```

Для клиента не измениться ничего:

```
router bgp 10
no synchronization
! ключевой момент - сессия с клиентами
neighbor 2.2.3.2 remote-as 10
```

Все это конечно хорошо, скажите Вы мне, но как-же избежать петель маршрутизации при использовании отражателей ? Для избежания образования петель при использовании отражателей используются атрибуты ORIGINATOR_ID и CLUSTER_LIST. Атрибут ORIGINATOR_ID в префиксе - это ROUTER_ID того маршрутизатора, который этот самый маршрут генерировал. И если полученный префикс имеет ORIGINATOR_ID равный ROUTER_ID нашего маршрутизатора - префикс отвергается.

Теперь разберемся с CLUSTER_LIST. Но для начала опять же представим нашу большую AS. Если AS большая, то в ней так или иначе немало и отражателей маршрутов. И эти отражатели имеют своих клиентов. Так вот отражатели вместе с клиентами представляют собой кластер отражателей маршрутов. Каждый кластер идентифицирует уникально число - CLUSTER_ID. Так вот - CLUSTER_LIST - это список тех CLUSTER_ID, через которых прошел префикс. И если маршрутизатор обнаруживает внутри CLUSTER_LIST свой CLUSTER_ID - маршрут игнорируется. Для активации данной возможности, необходимо разбить на некие логические участки причем сделать это с учетом имеющихся отражателей. Как уже говорилось отражатели вместе с клиентами будут составлять кластер. Затем каждому кластеру назначить некий id (число) и в настройках отражателей указать его с помощью команды: bgp cluster-id XX

К примеру если у Вас получается 3 кластера, то отражатели первого:

```
bgp cluster-id 100
```

второго:

```
bgp cluster-id 200
```

третьего:

```
bgp cluster-id 300
```

Что же лучше отражатели или конфедерации ? К сожалению, на этот вопрос однозначного ответа нет. Надо "смотреть по месту", то есть учитывать конкретные условия и требования. Иногда эти 2 подхода даже можно комбинировать - к примеру очень большую AS разбиваем на конфедерации, а внутри них - использовать отражатели.

BGP в работе - разные полезные советы

Итак мы почти дошли до конца. Но почти - не значит совсем. На "десерт" мы рассмотрим некоторые необязательные, но очень важные моменты работы протокола BGP.

И начнем с агрегации. Давайте подставим, что у нас есть 2 сети - 192.168.0.0/24 и 192.168.1.0/24. А вспомним любимый нами CIDR и запишем эти сети одной сетью - 192.168.0.0/23. Иначе говоря, маленькие ручейки стекаются в большие реки, то есть теоретически чем более приближен маршрутизатор к глобальный магистралям, тем менее длинными префиксами он оперирует. На практике (по многим причинам) это правило выполняется не всегда. Однако, так или иначе разобраться в вопросе необходимо - как минимум потому, что серьезный оператор не будет заниматься анонсированием сотни префиксов /24 (опять же в общем случае) , если их можно объединить в 2-3 префикса.

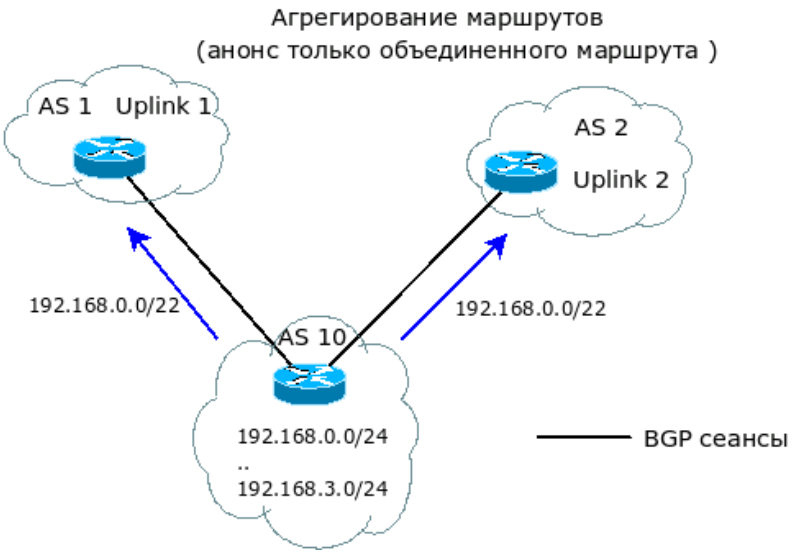
Итак главная задача агрегации - это сократить число префиксов и тем самым упростить диагностику проблем и повысить надежность. Да-да, именно надежность. Просто потому, что многие операторы в целях экономии аппаратных ресурсов (к слову, в примере Выше я также использовал данный прием) принимают только префиксы /23 и меньше ..... /24 они не принимают. То есть, иначе говоря, анонсируя только /24 Вы рискуете столкнуться с тем, что некоторая часть сети Вас видеть не будет. Кроме того, проблемы внутри сети при анонсировании отдельных префиксов могут быть причиной блокировки отдельных маршрутов Вашей сети как "проблемных" (о том что это такое - ниже).

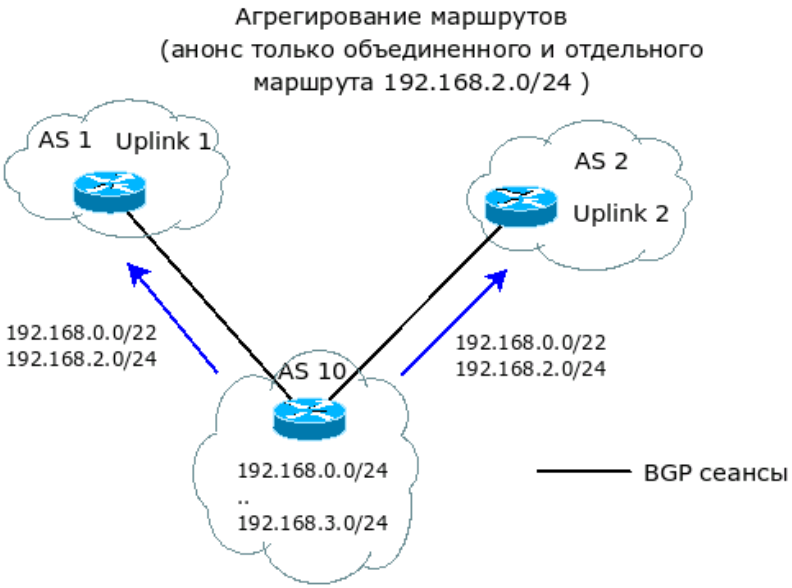
Агрегация может осуществляются несколькими способами способами:

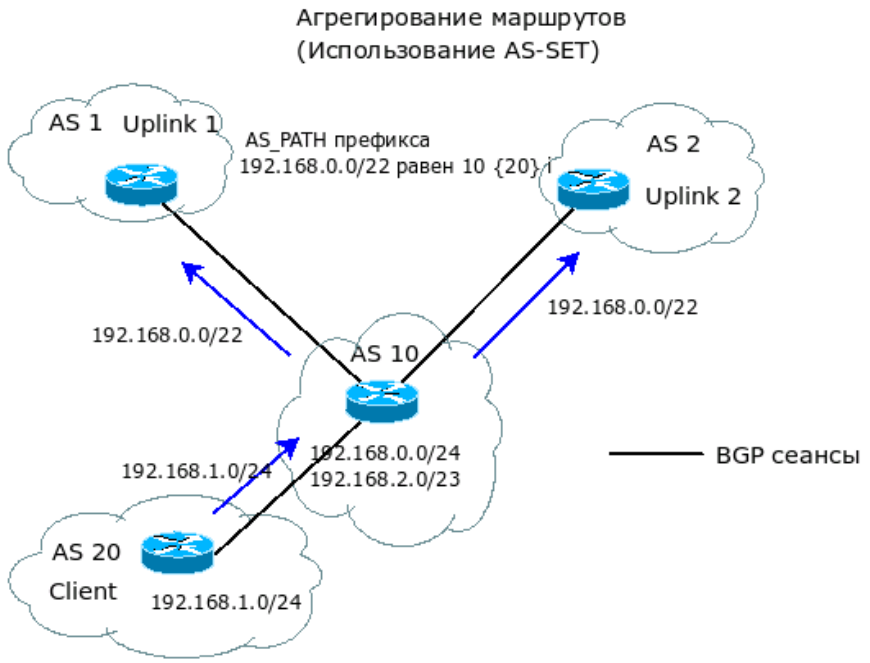
1)С помощью статических маршрутов. Как это не странно - но это так. Для понимания сути механизма, приведем пример. Вам выдана сеть 192.168.0.0/22. Естественно внутри сети Вы ее делите на необходимое к-во сетей. А анонсировать надо /22. В таком случае указанная сеть привязывается к интерфейсу Null0 и прописывается данную сеть в директиву network. Ну и само собой аплинкам с помощью префикс листов (я думаю Вы сами справитесь с построением соот. листа) анонсируем только эту сеть.

```
router bgp 10
no synchronization
! анонсируем объединенную сеть с помощью протокола bgp
network 192.168.0.0 mask 255.255.252.0
! сессия с аплинком1
neighbor 1.2.3.1 remote-as 20
neighbor 1.2.3.1 next-hop-self
! сессия с аплинком2
neighbor 1.1.3.1 remote-as 30
neighbor 1.1.3.1 next-hop-self
! а забываем, что bgp не анонсирует абы что, а анонсирует только те префиксы,
! которые есть в таблице маршрутизации. Мы соот. и привязываем объединенный
! префикс к интерфейсу Null0.
ip route 192.168.0.0 255.255.252.0 Null0
```

2)Использование специальной директивы (рис. 14, 15, 16).







Для управления объединением сетей используется директива aggregate-address. Синтаксис данной директивы имеет следующий вид.

```
aggregate-address сеть маска [as-set] [summary only] [suppress-map карта1] [attribute-map карта2 ]
```

Для понимания - приведем примеры: 1) просто

```
aggregate-address 192.168.0.0 255.255.252.0
```

Данная директива просто генерирует объединенный маршрут, но при этом не трогает более мелкие. То есть кроме /22 будут анонсироваться префиксы /23, /24 и др.

2) если мы добавим summary only - будет анонсироваться только объединенный маршрут, то есть /22.

Для понимания зачем необходимы as-set рассмотрим пример:

У Вас есть некая сеть - к примеру несколько префиксов /23 и /22. И есть клиент со своей AS, у которого тоже некие сети, но более мелкие - /24, к примеру. Если же некие Ваши префиксы и клиента объединить в один - получится красивый префикс /21. Но при просто объединении префиксов атрибуты теряются и, главное, сбрасывается AS_PATH. То есть префикс /21 будет выглядеть как сеть целиком из Вашей AS. Получается очень некрасиво: за такие фокусы можно и по шее получить ;) . Так вот, для того, что-бы в объединенном маршруте сохранялись такая Важная информация, как AS_PATH, то и необходимо добавлять в директиву aggregate-address опцию as-set. И при применении такого рода директивы, если часть маршрутов прошли через некую (или некие) AS, то этот список добавляется к AS_PATH в виде {список пройденных AS}. Вспоминая пример, AS_PATH для объединенного маршрута будет выглядеть так:

10 {20} i, где 20 - номер AS клиента.

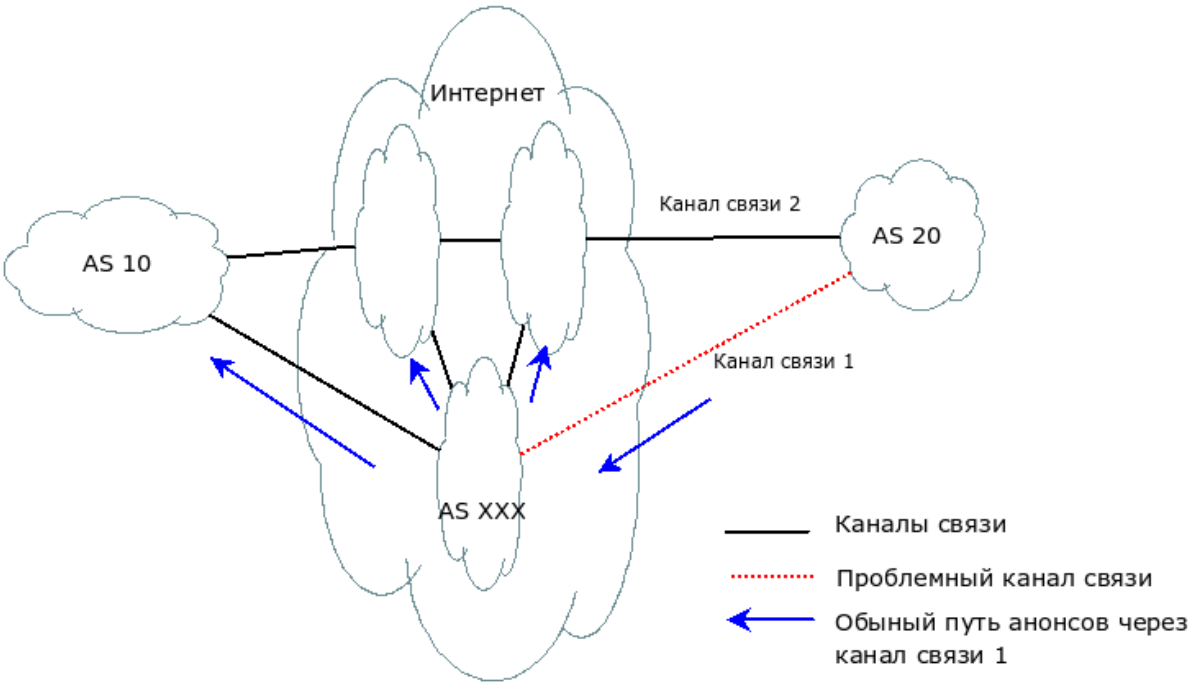
3) Опция suppress-map предназначена для того, что-бы четко определить: какие префиксы, кроме объединенного, подлежат анонсированию. Соответствующая карта строится по общепринятым правилам.

4) Опция attribute-map указываем на карту, которая изменяет атрибуты уже объединенного маршрута.

Таким образом, мы более или менее оптимизируем анонс своих префиксов. Сразу хочу оговориться, что бездумное объединение может плохо кончиться. Потому, прежде чем пользоваться этим инструментом, тщательно проверьте вносимые изменения не только на синтаксическую, но и логическую корректность.

Второй и главной темой для обсуждения нашего "десерта", стоит рассмотреть вопрос обеспечения стабильности протокола bgp.

Обеспечение стабильности протокола BGP



Пояснение:
Как только у анонсов, полученных через канал 1 количество штрафных балов (из-а проблем с каналом), превысит порог, AS XXX их заблокирует на определенное время. Установив более жесткие требования, Вы можете начать блокировать эти анонсы раньше.

К примеру есть некая сеть, которая имеет 2 канала выхода в глобальную сеть. Все бы хорошо, но один из этих каналов время от времени начинает "плющить как радиорелейку при восходе солнца" (знающие поймут). То упадет, то поднимется, и так далее. А поскольку путь через "плохой" канал наиболее короток, то пока по глобальной сети распространится информация о его непригодности, то все пакеты будут потеряны. Кроме того, каждая такая перестройка "кушает" ресурсы маршрутизаторов. Неплохо бы, скажете Вы, неким образом в автоматическом режиме отлавливать такого рода жуков и на время помещать их в некий "черный список": вдруг хозяин AS починит за это время канал. И еще лучше - при стабилизации маршрута исключать их из "черного списка". Можно ли так сделать ?

Можно, и делается это с помощью механизма "разгрузки маршрутов". Его суть сводится к простой схеме: каждому маршруту присваивается специальный индекс "ненадежности", по англицки – penalty, и, начиная с некоторого значения penalty, маршрут просто блокируется. Расчет penalty довольно прост: у нас есть некий промежуток времени, и если в его течении произойдет пропадание/возвращение маршрута, к значению penalty добавляется штрафной бал. Если за интервал времени проблем не было - штрафные балы маршруту снимаются. Как только сумма балов превысит разумные пределы, маршрут блокируется (то есть не принимает участие в принятии решений) на указанное время. Блокировка может быть снята раньше, если сумма штрафных балов снизится до значения, меньшего спец. определенной цифре.

Перейдем к примеру. Нам необходимо защитить свою сеть от "зловредных" маршрутов. Для этого служить директива bgp dampening. формат: bgp dampening [таймер1 порог_восстановления порог_отбрасывания время_блокировки] [route-map имя_карты]

Для начала рассмотрим первый случай - для всех сетей глобальный настройки, то есть: bgp dampening [таймер1 порог_восстановления порог_отбрасывания время_блокировки]

таймер1 - это у помянутый Выше промежуток времени. От 1-й до 45 минут. По умолчанию - 15 минут. порог_восстановления - это порог штрафных балов, при значении ниже которого маршрут вновь становиться активным. Может принимать значения от 1 до 20000. По умолчанию - 750. порог_отбрасывания - этот порог штрафных балов, при достижении которого маршрут блокируется. Может принимать значения от 1 до 20000. По умолчанию - 2000. время_блокировки - максимально время блокировки маршрута. От 1-й до 255 минут. По умолчанию - рассчитывается по формуле: таймер1х4.

К примеру, Вы хотите интервалом измерения сделать 10 минут, порог отбрасывания 5000, а порог восстановления - 2000.

```
bgp dampening 10 2000 5000 40
```

Теперь вспомним, что под одну гребенку всех грести все-же не стоит. Неплохо бы для разных сетей устанавливать разные пороги.

Для этого и служит второй вариант - bgp dampening route-map имя_карты В качестве примера - для одних сетей устанавливаем указанные Выше параметры, а для всех остальных более жесткие - 10 500 1000 40.

```
router bgp 10
no synchronization
bgp dampening route-map SET_DUMP

! для сетей, полученных от аплинков - первые значения
route-map SET_DUMP permit 1
match community UPLINK
set dampening 10 2000 5000 40

! для остальных сетей
route-map SET_DUMP permit 5
set dampening 10 500 1000  40
```

Теперь вернемся к первой главе и вспомним о том, как мы меняли параметры BGP-сессии и после этого сбрасывали ее "жестким" (то есть сбросом TCP сеанса) образом. Тогда это было приемлемо. Теперь же у нас большая сеть и мы не можем позволить себе роскошь при каждом чихе сбрасывать BGP сессию. Для того, чтобы менять параметры BGP-сессии, и при этом сохранять стабильность сети, необходимо воспользоваться механизмом мягкой перестройки. Для мягкой перестройки параметров, связанных с исходящими параметрами, необходимо воспользоваться командой:

clear ip bgp IP_соседа soft out

Для мягкой перестройки входящих параметров следует воспользоваться командой:

```
clear ip bgp IP_соседа soft in
```

И в качестве последнего аккорда рассмотрим понятие "труппы узлов" и, главное, зачем это нужно. В ряде случаев, многие BGP-сессии имеют похожие параметры и очень не хочется по 10 раз вводить одно и тоже (да и попробуй потом этом зоопарке разберись). Для того, чтобы не напрягаться как при настройке так и при разборе конфигурации, и применяют группы узлов.

Синтаксис:

```
neighbor ИМЯ_ГРУППЫ peer-group
neighbor ИМЯ_ГРУППЫ общая опция 1
...
neighbor ИМЯ_ГРУППЫ общая опция N

neighbor X.Y.Z.W peer-group ИМЯ_ГРУППЫ
```

К примеру, у Вас есть группа маршрутизаторов из Вашей AS, которые являются клиентами Вашего отражателя маршрутов, должны получать COMMUNTY и анонсы от них должны проходить через одну карту маршрутов.

Реализация:

```
neighbor PE peer-group
neighbor PE remote-as 10
neighbor PE route-reflector-client
neighbor PE send-community
neighbor PE route-map AS10-PE-In in
```

А описание самой bgp сессии с этой группой сведется к:

```
neighbor 1.2.3.1 peer-group PE
..........
neighbor 1.2.3.N peer-group PE
```