BGP (Border Gateway Protocol)

Описание

Еще глубже в динамическую маршрутизацию. Разбираемся с BGP (Border Gateway Protocol)

Оглавление

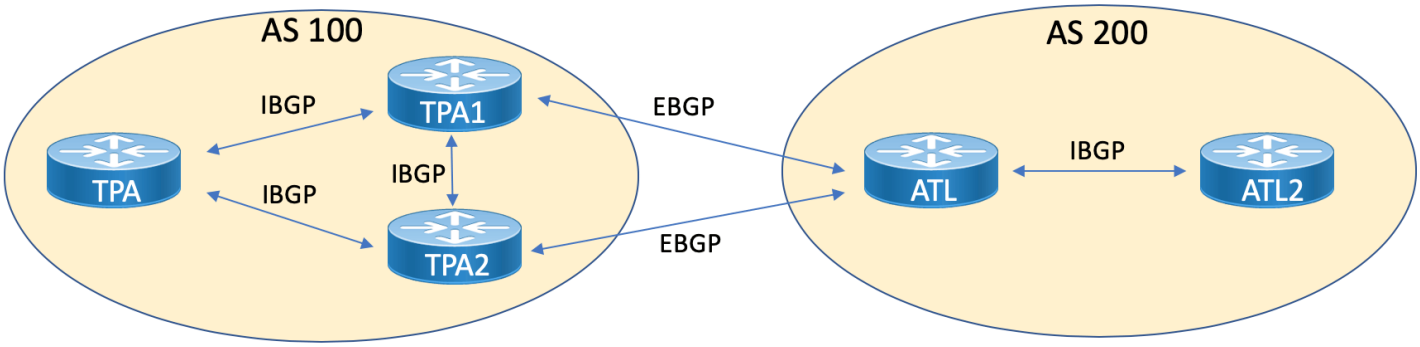
1. Основы протокола Border Gateway Protocol (BGP)

В этой статье вы познакомитесь с основами BGP и узнаете о его различных типах сообщений и состояниях.

ОБЗОР BGP

Давайте посмотрим правде в глаза - **Border Gateway Protocol** невероятно уникален, особенно когда мы сравниваем его с другими протоколами маршрутизации. Самое первое, что делает **BGP** таким уникальным, - это то, что он наш единственный внешний шлюзовой протокол (**EGP**), широко используемый сегодня. Мы знаем, что у нас есть **Interior Gateway Protocols (IGPs)**, и похожий на **OSPF**, работающий внутри автономной системы. Но BGP - это EGP, а это означает, что он (как правило) будет принимать префиксы, которые находятся внутри автономной системы, и отправлять их в другие автономные системы.

На рисунке 1 показан пример топологии BGP.



Именно поэтому протокол BGP является протоколом, который обеспечивает функционирование сети. Интернет-провайдеры (**ISP**) могут использовать BGP для перемещения префиксной информации между другими Интернет-провайдерами. Однако уникальные характеристики BGP на этом не заканчиваются. Одна из вещей, которая очень уникальна в протоколе, заключается в том, что он формирует пиринги (***равноправный информационный обмен**) точка-точка с другими спикерами BGP, и вы должны создавать эти пиринги вручную.

С протоколом пограничного шлюза (BGP) нет такой вещи, как автоматическое формирование соседства с целой кучей устройств на одном сегменте. Для каждого из устройств, с которыми BGP должен пиринговать, он делает это с помощью одного одноранговое отношения, которое мы предпочитаем называть пирингом BGP.

Еще одно очень уникальное свойство заключается в том, что BGP - это протокол прикладного уровня. По общему признанию, большинство сетевых инженеров поспорили бы, что это протокол сетевого уровня – и они проиграли бы этот спор!

Как компонент прикладного уровня, BGP делает что-то блестящее. Он использует протокол управления передачей (TCP) для своих операций. Если мы рассмотрим EIGRP в качестве примера, то создателям пришлось приложить большие усилия, чтобы встроить надежность в сам протокол. Например, спикер EIGRP будет передавать многоадресные передачи, и, если это не сработает, он вернется к одноадресным передачам, чтобы попытаться обеспечить надежность.

С помощью Border Gateway Protocol разработчики решили не включать в протокол все эти типы контроля надежности. Они просто полагаются на чудесную надежность коммуникаций TCP. В частности, BGP использует TCP- порт `179`.

Когда мы думаем о наших протоколах маршрутизации, мы знаем, что будет некоторое значение, которое будет служить метрическим значением для измерения расстояния. Например, в случае OSPF мы знаем, что метрикой является стоимость, а стоимость напрямую зависит от пропускной способности.

BGP не работает таким образом. Протокол BGP использует атрибуты, а не только одного показателя. Одним из главных атрибутов протокола BGP называется атрибута `AS_PATH`. Это список всех автономных систем (**AS**), которые префикс должен был передать на своем пути, скажем, в вашу автономную систему.

`AS_PATH` - это фактически запись всей информации о пути AS. Путь AS настолько важен для функции BGP, что протокол часто называют протоколом маршрутизации вектора пути. Обратите внимание, что это не протокол вектора расстояния (**Distance Vector**), а вектор пути (**Path Vector**). `AS_PATH` используется не только для определения наилучшего пути к месту назначения (т.е. более короткого пути AS), но и в качестве механизма предотвращения петель.

Когда автономная система видит свой собственный номер **AS** в `AS_PATH`, она очень обеспокоена тем, что в коммуникациях может быть петля. Что- то еще, что делает BGP невероятно уникальным, - это тот факт, что, когда мы формируем пиринги внутри автономной системы, они называются внутренними пирингами BGP, а правила, которым следуют, являются внутренними правилами BGP (**IBGP**).

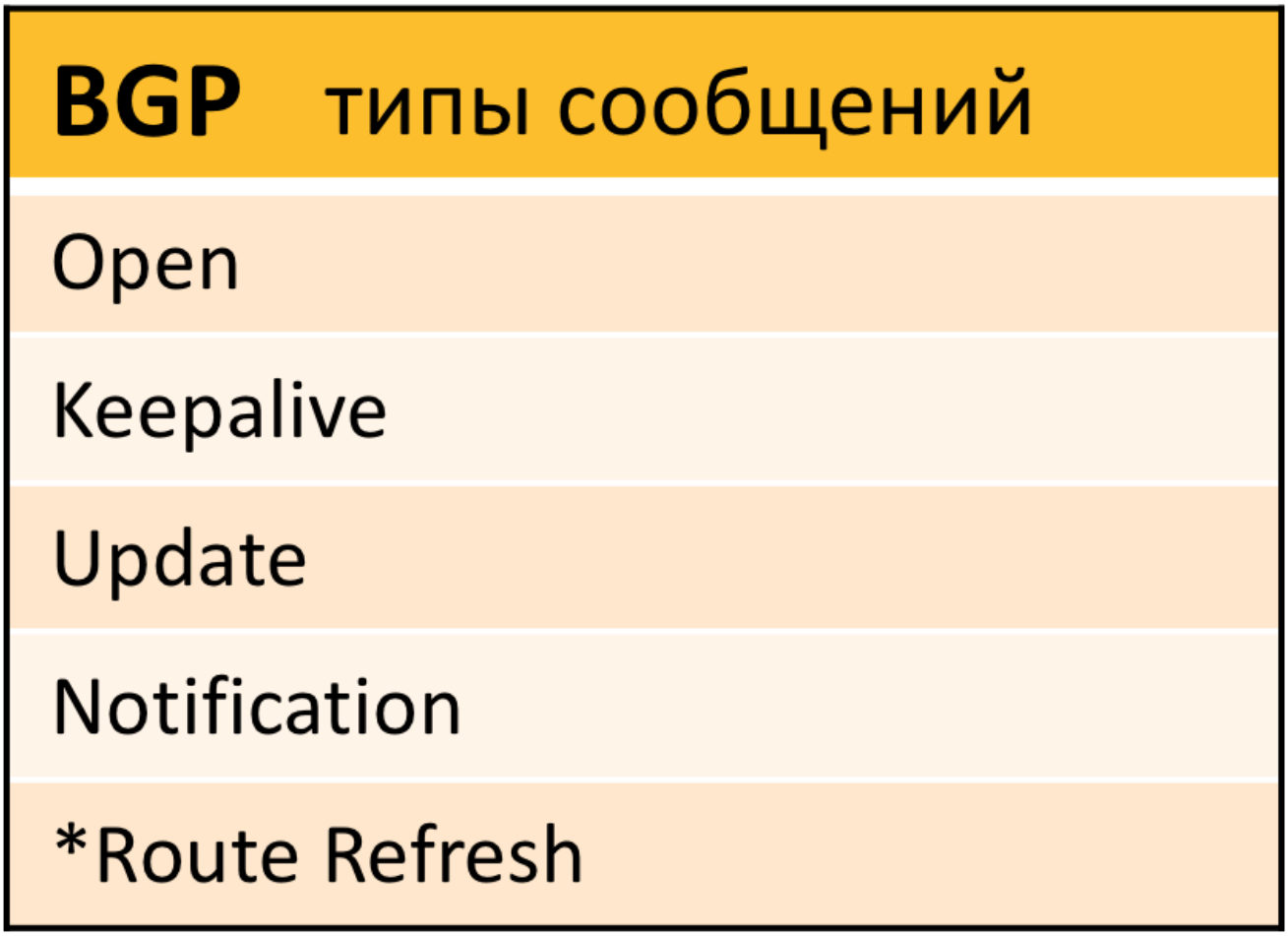
Когда мы формируем пиринг между автономными системами, это называется протоколом внешнего пограничного шлюза (**EBGP**). (Примечание: в некоторых литературных источниках EBGP пишется как **eBGP**.) Помните, что причина, по которой BGP различает пиринг IBGP и пиринг EBGP, заключается в том, что эксплуатационные характеристики должны изменяться в зависимости от того, как выполняется пиринг. Например, мы заявили, что существует путь AS, который записывает автономные системы, которые передаются. Очевидно, что при пиринге EBGP, когда префикс передается от одного AS к другому AS, отправляющий AS должен поместить свою автономную систему в путь. Но с IBGP, префикс остается в AS, поэтому протокол BGP не обновляет значение AS. Вы можете вернуться к рисунку 1, чтобы увидеть эти различные типы пиринга в действии.

Таким образом, правила меняются, когда мы говорим о IBGP против EBGP, чтобы быть последовательным и безошибочными. И уникальные свойства BGP просто не заканчиваются на этом.

ТИПЫ СООБЩЕНИЙ BGP, ФОРМАТЫ И СОСЕДНИЕ ТИПЫ СООБЩЕНИЙ СОСТОЯНИЯ СОСЕДСТВА BGP

Многие люди описывают протокол пограничного шлюза (BGP) как чрезвычайно сложный протокол, но я не согласна с этим. Видите ли, установка политик BGP и контроль распространения префиксов внутри BGP-это может быть довольно сложно. Но сам протокол, хотя и уникален, в основном прост в своей работе.

В этом части статьи мы рассмотрим типы сообщений BGP. На рисунке 2 показаны различные типы сообщений BGP.



Запомните первый шаг. Когда два спикера BGP хотят сформировать пиринг, они будут полагаться на протокол управления передачей (TCP). И, конечно, мы знаем, что будет **three-way handshake** (трехстороннее рукопожатие) с TCP, чтобы начать этот надежный сеанс связи.

Что же происходит дальше? Так это то, что эти устройства будут обмениваться открытыми сообщениями. Открытое сообщение содержит очень важную информацию, основным компонентом которой является номер автономной системы одноранговогo узла. Это будет определять, является ли это пиринг IBGP или пиринг EBGP.

Когда происходит обмен открытыми сообщениями, то спикеры BGP далее начинают обмениваться сообщениями **Keepalive**. Это, простой механизм, чтобы убедиться, что другой прибор жив, счастлив и здоров, и что пиринг в состоянии up. После этого спикеры BGP получают обновления для совместного использования, называемое сообщением **Update**.

Если в какой-то момент времени что-то пойдет не так, спикеры BGP могут использовать простое сообщение **Notification**. Данное сообщение прерывает пиринг в результате ошибки, которая может произойти с BGP.

Одним из очень интересных типов сообщений BGP является тип сообщения **Route Refresh** (обновления маршрута). Хотя этот тип сообщений не был включен в исходный стандарт BGP, большинство наших основных сетевых вендоров поддерживают Route Refresh. Route Refresh позволяют соседям обновлять, скажем, информацию о маршруте BGP или даже обновлять вещи после довольно серьезной реконфигурации политики, не разрушая пиринг и не влияя на пиринг каким- либо большим негативным образом.

Рисунок 3 показывает эти типы сообщений в действии благодаря захвату **Wireshark**'ом обмена сообщениями BGP в нашем примере топологии из рисунка 1.

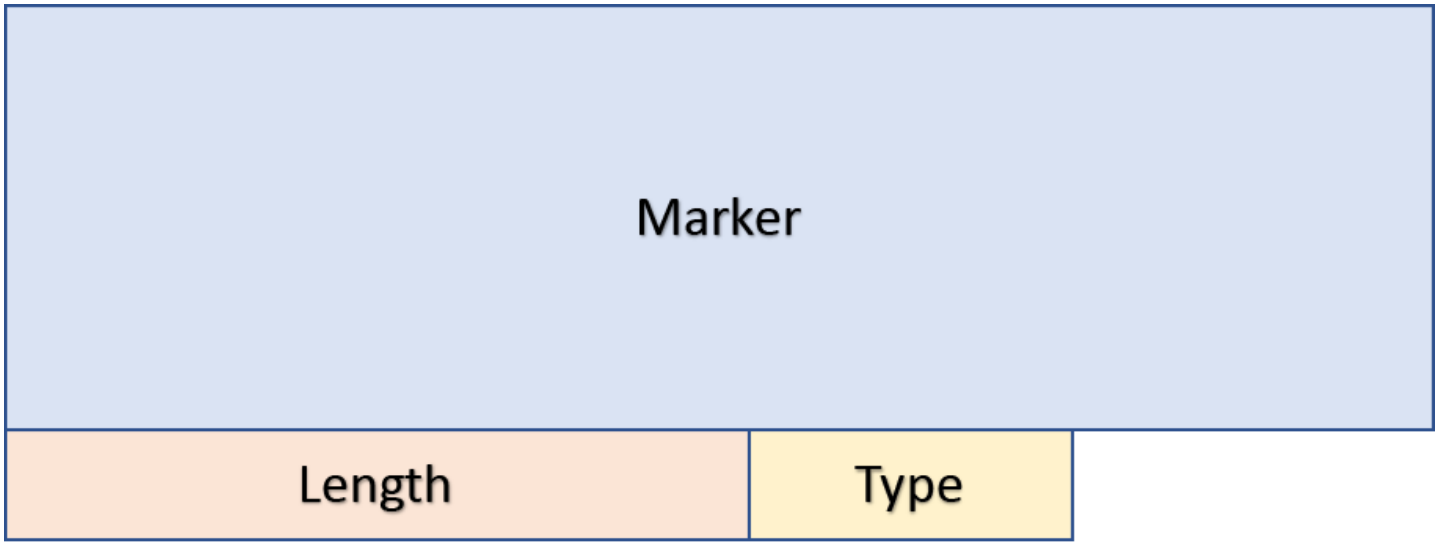
No.	Time	Source	Destination	Protocc ▲	Length	Info
41	59.461243	10.10.10.1	10.10.10.2	BGP	102	OPEN Message
43	59.461810	10.10.10.2	10.10.10.1	BGP	102	OPEN Message
44	59.471556	10.10.10.1	10.10.10.2	BGP	68	NOTIFICATION Message
45	59.471618	10.10.10.2	10.10.10.1	BGP	68	NOTIFICATION Message
59	67.657270	10.10.10.2	10.10.10.1	BGP	102	OPEN Message
60	67.667379	10.10.10.1	10.10.10.2	BGP	102	OPEN Message
61	67.667432	10.10.10.1	10.10.10.2	BGP	63	KEEPALIVE Message
62	67.677751	10.10.10.2	10.10.10.1	BGP	63	KEEPALIVE Message
63	67.677774	10.10.10.2	10.10.10.1	BGP	63	KEEPALIVE Message
64	67.687894	10.10.10.2	10.10.10.1	BGP	67	UPDATE Message
66	67.923125	10.10.10.1	10.10.10.2	BGP	63	KEEPALIVE Message
67	67.923182	10.10.10.1	10.10.10.2	BGP	67	UPDATE Message
78	111.437262	10.10.10.2	10.10.10.1	BGP	98	UPDATE Message

```
▶ Transmission Control Protocol, Src Port: 51812, Dst Port: 179, Seq: 120, Ack: 120, Len: 54
▼ Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 54
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 27
  ▼ Path attributes
    ▶ Path Attribute - ORIGIN: IGP
    ▶ Path Attribute - AS_PATH: 200
    ▶ Path Attribute - NEXT_HOP: 10.10.10.2
    ▶ Path Attribute - MULTI_EXIT_DISC: 0
  ▼ Network Layer Reachability Information (NLRI)
    ▶ 100.100.100.0/24
```

ФОРМАТЫ СООБЩЕНИЙ BGP

В этом части статьи мы еще больше узнаем об эксплуатационных характеристиках Border Gateway Protocol, более подробно рассмотрев типы сообщений BGP.

Каждый тип сообщения имеет заголовок BGP. Этот заголовок показан на рисунке 4. Вы видите, что заголовок BGP имеет большое поле маркера. Можно подумать, что это чрезвычайно важно. Он имеет размер **16 октетов**. Как оказалось, это поле будет заполнено у всех.



Это связано с тем, что использование этого поля маркера было прописано в устаревшем стандарте. Первоначальная идея этого поля состояла в том, что его можно было бы использовать для обнаружения таких событий, как потеря синхронизации между двумя одноранговыми узлами, и также считалось, что это будет область, в которой может храниться аутентификационная информация.

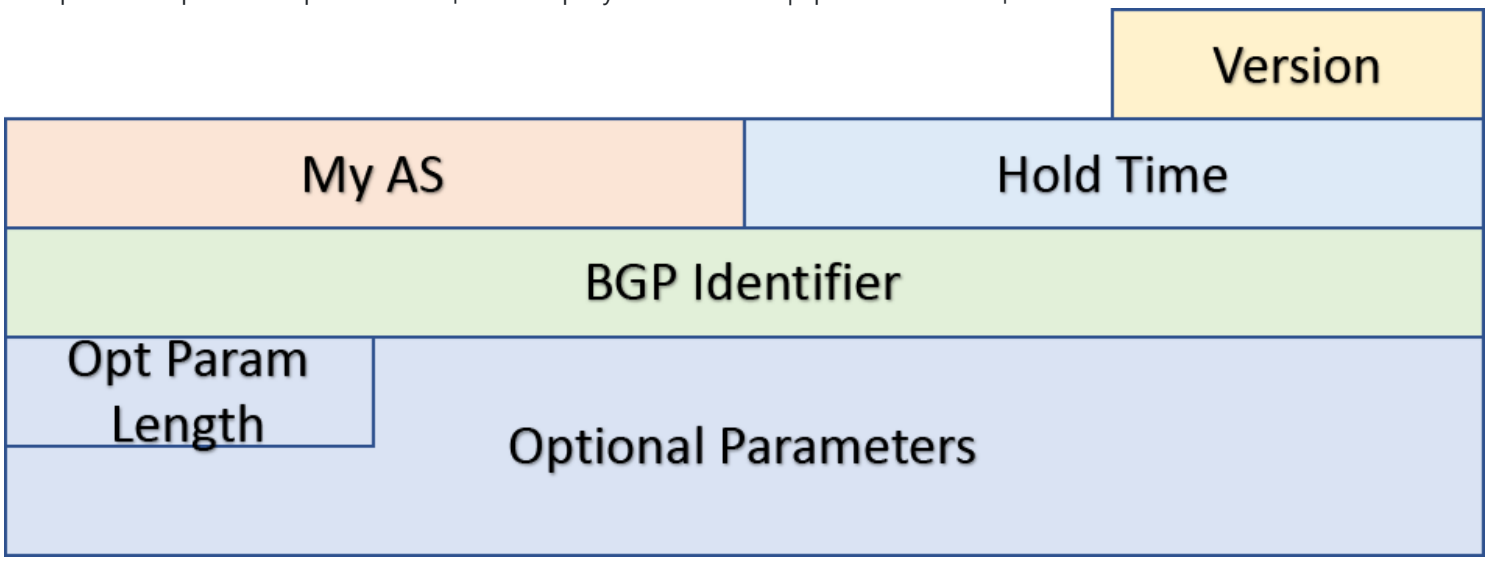
Почему это поле вообще имеется в BGP? Иногда, в очень редком случае, когда необходимо иметь обратную поддержку с каким-то действительно старым устройством BGP, которое ожидает эту информацию из поля маркера.

Важными полями в заголовке, будут длина (**Length**) (то есть длина всего сообщения) и поля типа (**Type**). Поле Тип указывает, с каким типом сообщения BGP мы имеем дело.

Если, например, в этом поле 1, вы имеете дело с открытым (**Open**) сообщением BGP. Значение 2 указывает на сообщение об обновлении (**Update**). А 3 означает уведомление (**Notification**). Значение 4 будет иметь сообщение **Keepalive**. 5 указывает на необязательное **Route Refresh**.

То, что следует за информацией заголовка, конечно же, является данными, за одним важным исключением- это сообщение Keepalive. По определению, в сообщении Keepalive нет никаких данных.

Теперь я надеюсь вы понимаете, что, когда ваша система хочет сформировать BGP-пиринг с другим устройством, она собирается отправить открытое сообщение. На рисунке 5 показан формат этих сообщений.



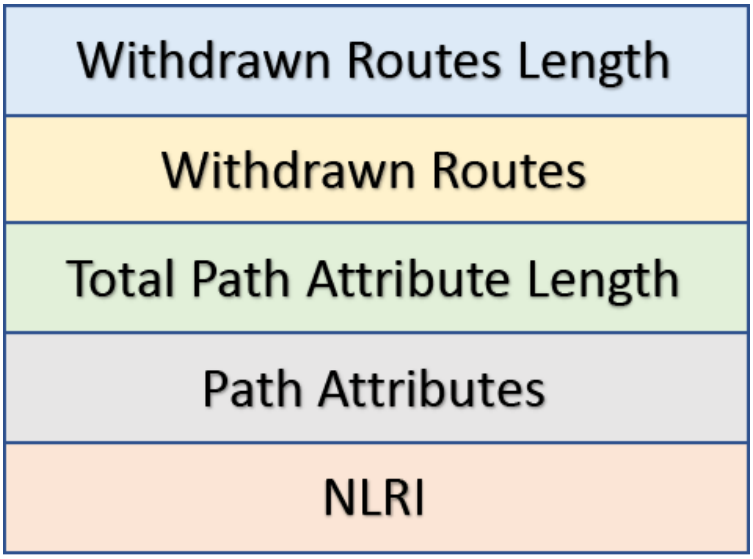
Когда мы смотрим на формат открытого (**Open**) сообщения, мы замечаем, что там есть номер версии. Именно так BGP указывает на версию BGP, которую вы используете.

Ваша система также отправит свой номер AS в открытом сообщении. Это очень важно для такого поведения IBGP по сравнению с EBGP. Существует значение **Hold Time**. Что же такое Hold Time? Когда маршрутизатор, с которым вы хотите свериться, получает Open сообщение, он смотрит время удержания (Hold Time), смотрит на свое собственное настроенное Hold Time, а затем использует меньшее из двух значений. Hold Time должно быть либо нулевым, либо не менее трех секунд.

Есть поле **BGP Identifier**. Это Ваш **BGP Router ID**, и это уникальное значение, которое будет однозначно отличать вашу систему в пирингах BGP.

Наконец, у нас есть дополнительные параметры (**Optional Parameter**), которые можно задать с помощью открытого сообщения. Там есть необязательная длина параметра (**Optional Parameter Length**), а затем сами параметры, дающие дополнительную гибкость работы с протоколом.

Еще одно действительно важное сообщение, которое у нас есть, - это сообщение об обновлении (**Update**) BGP. На рисунке 6 показана эта структура сообщения.



Сообщение об обновлении BGP содержит индикатор длины отозванных маршрутов (**Withdrawn Routes Length**). Это гарантирует, что сообщение обновления является средством для маршрутов, которые будут удалены из таблицы BGP соседа. Примечание: затем в сообщение об обновлении вставляется список изъятых маршрутов.

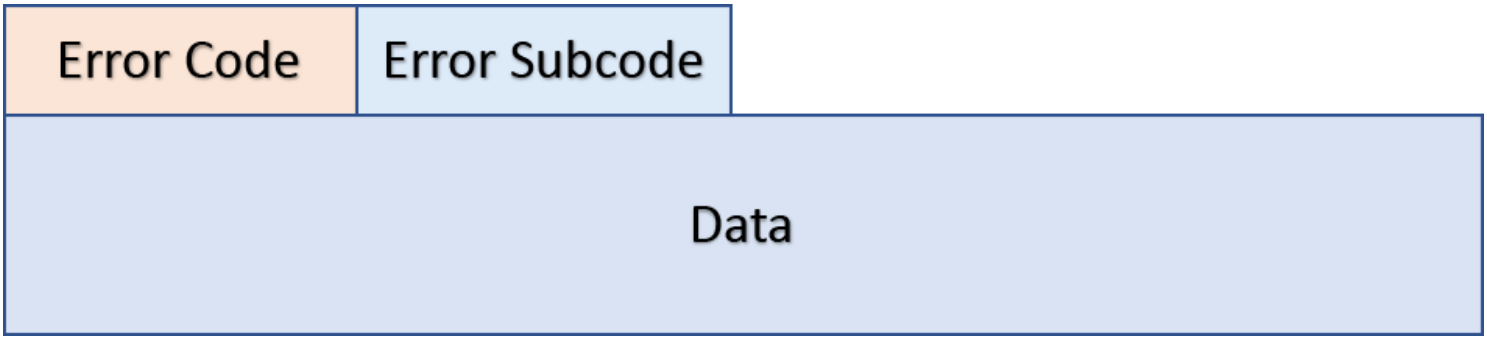
Сообщение об обновлении содержит поля, которые используются для обмена информацией о префиксах сети с соседями и включают в себя очень важную атрибутивную информацию, связанную с префиксами. Помните, что эти атрибуты позволяют Вам принимать важные решения о том, как BGP будет фактически маршрутизировать информацию в сети.

Хорошо известный атрибут, о котором мы уже упоминали, - это путь. Вы помните, что это список автономных систем, которые префикс передал на своем пути по всей инфраструктуре BGP. AS Path будет примером атрибута, который должен быть в сообщении об обновлении, когда он используется для отправки префиксов. Там может быть много атрибутов, которые мы используем, и это является причиной для Total Path Attribute Length в сообщении об обновлении.

Сама информация о префиксе сети находится в поле `NLRI`. Это означает информацию о достижимости сетевого уровня (**Network Layer Reachability Information**). Вы можете вернуться к рисунку 3 и увидеть эти поля в реальном пакете, а также их содержимое.

Создатели BGP сделали гениальную вещь. Они создали протокол для передачи `NLRI` таким образом, чтобы он был гибким по мере изменения сетей и необходимости передачи новой информации. BGP создан для того, чтобы сразу же запускать для нас такие вещи, как IPv6. Он также может легко переносить префиксы VPN IPv4 внутри чего-то вроде **MPLS VPN**.

На рисунке 7 показаны поля сообщения уведомления (Notification).



Самое первое поле - это код ошибки (**Error Code**). Затем поле Подкод ошибки (**Error Subcode**). Эти поля дают нам общий тип ошибки, а затем еще больше информации. Например, если в Error Code у нас есть значение 3, а затем в Error Subcode у нас есть значение 3, это указывает на то, что существует сообщение об ошибке обновления.

СОСЕДСТВО BGP

Точно так же, как мы можем многое узнать о работе BGP, изучая сообщения BGP и их форматы, мы также можем многое узнать о BGP, изучая различные состояния, через которые проходит пиринг BGP. На самом деле, они имеют решающее значение при устранении неполадок. Когда вы проанализируете протокол BGP, вы не удивитесь, узнав, что существует множество встроенных механизмов для обеспечения стабильности.

Многие IGP спроектированы так, чтобы быть максимально быстро сходящимися. Это происходит потому, что в момент, когда происходит изменение внутри сети вашей организации, мы хотим **sub-second** сходимости других устройств, чтобы мы знали об этом изменении. BGP спроектирован по-другому. Таймеры имеют гораздо большую продолжительность, чем мы привыкли бы с нашим IGP, потому что мы хотим стабильности, жертвуя скоростью сходимости. В конце концов, BGP имеет дело с общедоступными таблицами маршрутизации интернета в развертываниях поставщиков услуг. Эти таблицы маршрутизации очень массивны. Нестабильность в этой среде приведет к катастрофе всего публичного Интернета.

Когда вы изучите состояние соседства BGP, вы поймете для чего это. Относительно большое число состояний соседства BGP, показанных на рисунке 8, свидетельствует о тщательных усилиях по обеспечению стабильности протокола маршрутизации.

Обратите внимание, что есть состояние простоя, когда устройство не инициирует ни одно из других состояний, и есть установленное состояние, когда оно полностью установлено со своим узлом. Что несколько удивительно, так это то, что есть все эти "промежуточные" состояния подключения, активного, открытого подтверждения (**OpenConfirm**) и активного.

Состояние — подключения-это состояние, в котором устройство BGP ожидает завершения TCP- соединения с соседним устройством.

В активном состоянии он пытается инициировать TCP - соединение со своим соседом. В состоянии **OpenSent**, как вы можете догадаться, он отправляет свое открытое сообщение и ждет ответа от своего соседа с его открытым сообщением. В режиме OpenConfirm, спикер BGP на самом деле ждет, Keepalive на основе успешного обмена открытыми сообщениями. Будем надеяться, что устройство BGP получит Keepalive. Если будет ошибка, он получит уведомление.

Используя в Cisco CLI специальные команды, можно узнать все о состоянии BGP. Пример 1 показывает использование команды `show ip bgp summary` для проверки соседнего состояния.

```
TPA1#show ip bgp summary
BGP router identifier 10.10.10.1, local AS number 100
BGP table version is 3, main routing table version 3
Neighbor        V       AS      MsgRcvd MsgSent TblVer  InQ     QutQ    Up/down State/PfxRcd
10.10.10.2      4       200     0       0       1       0       0       00:00:00        Idle
```

Обратите внимание на пример 1. Этот пиринг BGP находится в состоянии ожидания (параметр **State/PfxRcd** в состоянии **Idle**). Как только произойдет соединение значение IDLE заменится на 1 (Если ATL использует только один префикс с TPA 1).

2. Построение маршрута протоколом BGP

В первой части статей о протоколе **Border Gateway Protocol (BGP)** мы узнали и разобрали протокол BGP, а затем изучили типы сообщений BGP и состояния соседства. Сегодня, в этой статье, вы узнаете об одном из самых сложных аспектов BGP: как он принимает решение о выборе маршрута.

В то время как протоколы маршрутизации, такие как RIP, OSPF и EIGRP, имеют свои собственные метрики, используемые для выбора «лучшего» пути к целевой сети, BGP использует коллекцию атрибутов пути (**PAs**).

BGP- АТРИБУТЫ ПУТИ (PATH ATTRIBUTES)

Когда ваш спикер BGP получает BGP префикс, к нему будет прикреплено множество атрибутов пути, и мы знаем, что они будут иметь решающее значение, когда речь заходит о том, чтобы BGP выбрал самый лучший путь к месту назначения.

Все атрибуты BGP- маршрута, делятся на четыре основные категории.

- Well-Known Mandatory
- Well-Known Discretionary
- Optional Transitive
- Optional Non-Transitive

Обратите внимание, что две категории начинаются с термина **Well-Known**. Well-Known означает, что все маршрутизаторы должны распознавать этот атрибут пути. Две другие категории начинаются с термина **Optional**. Optional означает, что реализация BGP на устройстве вообще не должна распознавать этот атрибут.

Тогда у нас есть термины **mandatory** и **discretionary**, связанные с термином Well-Known. Mandatory означает, что обновление должно содержать этот атрибут. Если атрибута нет, тогда появится сообщение об ошибке уведомления, и пиринг будет удален. Discretionary, конечно, будет означать, что атрибута не должно быть в обновлении.

У необязательных категорий атрибутов есть- транзитивные и нетранзитивные. Если он транзитивен, то устройство должно передать этот атрибут пути своему следующему соседу. Если он не является транзитивным, то может просто игнорировать это значение атрибута.

Пример 1 показывает проверку нескольких атрибутов пути для префикса, который был получен маршрутизатором TPA1 от маршрутизатора ATL. Обратите внимание, что мы используем команду `show ip bgp` для просмотра этой информации, которая хранится в базе данных маршрутизации BGP. В частности, этот вывод показывает атрибуты `Next Hop`, `Metric (MED)`, `LocPrf (Local Preference)`, `Weight`, и `Path (AS Path)`.

```
TPA1#show ip bgp
BGP table version  is 4, local router ID is 10.10.10.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal, r RIB-failure, S Stale
Origin codes^ I – IGP, e – EGP, ? - incomplete
        Network Next Hop        Metric  LocPrf  Weight  Path
*>      100.100.100.0/24        10.10.10.2      0                       200         i
```

АТРИБУТ ORIGIN

Атрибут **ORIGIN** в BGP-это попытка записать, откуда пришел префикс. Существует три возможности, когда речь заходит о происхождении этого атрибута: **IGP**, **EGP** и **Incomplete**. Как видно из легенды примера 1, коды, используемые Cisco для этих источников, являются `i`, `e`, и `?`. Для префикса, показанного в примере 1, можно увидеть, что источником является IGP. Это указывает на то, что префикс вошел в эту топологию благодаря сетевой команде внутри конфигурации этого исходного устройства. Далее в этой статье мы рассмотрим сетевую команду во всей ее красе. Термин IGP здесь предполагает, что префикс произошел от записи протокола внутреннего шлюза (**Gateway Protocol**). Допустим, у нас есть префикс в нашей таблице маршрутизации OSPF, а затем мы используем сетевую команду внутри BGP, чтобы поместить его в экосистему BGP. Конечно, IGP - не единственный источник префиксов, которые могут нести этот атрибут. Например, вы можете создать локальный интерфейс обратной связи на устройстве, а затем использовать сетевую команду для объявления этого локального префикса в BGP.

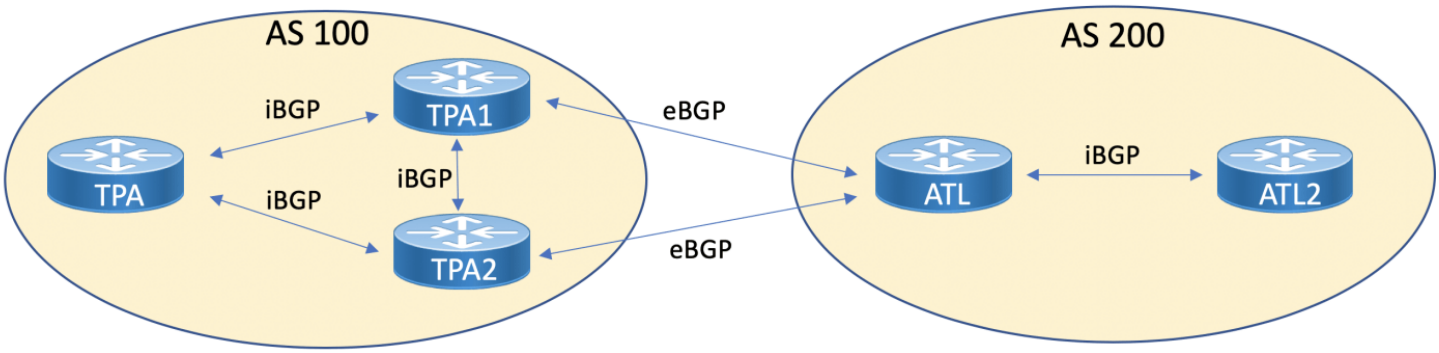
EGP ссылается на ныне устаревший протокол внешнего шлюза (**Exterior Gateway Protocol**), предшественник BGP. В результате вы не увидите этот исходный код.

Incomplete означает, что BGP не уверен в том, как именно префикс был введен в топологию. Наиболее распространенным сценарием здесь является то, что префикс был перераспределен в Border Gateway Protocol из какого-то другого протокола, обычно IGP.

Возникает вопрос, почему исходный код имеет такое значение. Ответ заключается в том, что это ключевой фактор, когда BGP использует свой алгоритм для выбора наилучшего пути к месту назначения в сети. Он может разорвать «связи» между несколькими альтернативными путями в сети. Мы также уделяем этому атрибуту большое внимание, потому что это действительно один из хорошо известных, обязательных атрибутов, которые должны существовать в наших обновлениях.

АТРИБУТ AS PATH

AS Path - это well-known mandatory атрибут. Он очень важен для наилучшего поиска пути, а также для предотвращения петель внутри Border Gateway Protocol.
Рассматривая нашу топологию, показанную на рисунке 1, рассмотрим префикс, возникший в TPA. Обновление отправляется в TPA1, и TPA не добавляет свой собственный AS 100 в AS Path, так как сосед, которому он отправляет обновление, находится в своем собственном AS в соответствии с пирингом iBGP.



Когда TPA1 отправляет обновления на ATL, он добавляет номер 100 в обновления. Следуя этой логике, ATL отправит обновления на ATL2 и не будет добавлять свой собственный номер в качестве AS. Это будет работать до тех пор, пока ATL2 не отправит обновления на какой-то другой AS, предшествующий AS 200. Это означает, что, когда мы рассматриваем образец AS path, как показано в примере 2, крайним правым в пути является AS, который первым создал префикс (100), а крайним левым- AS, который доставил префикс на локальное устройство (342).

Пример 2: Пример BGP AS Path

```
[342 435 200 100]
```

АТРИБУТ NEXT HOP

На самом деле нет ничего удивительного в том, что префикс BGP имеет атрибут под названием **Next Hop**. В конце концов, маршрутизатор должен знать, куда отправлять трафик для этого префикса. Next Hop атрибут удовлетворяет эту потребность. Интересным моментом здесь, однако, является тот факт, что Next Hop в BGP работает не так же, как это происходит в большинстве IGP. Также следует отметить, что правила меняются, когда вы рассматриваете iBGP в сравнении с eBGP.

При рассмотрении протокола внутреннего шлюза, когда устройство отправляет обновление своему соседу, значением Next Hop по умолчанию является IP-адрес интерфейса, с которого отправляется обновление. Этот параметр продолжает сбрасываться каждым маршрутизатором по мере прохождения обновления через топологию. Next Hop принимает простую парадигму «**hop-by-hop**».

С помощью BGP, когда у нас есть пиринг eBGP и отправляется префикс, Next Hop действительно будет (по умолчанию) IP-адресом спикера eBGP, отправляющего обновление. Однако IP-адрес этого спикера eBGP будет сохранен в качестве Next Hop, поскольку префикс передается от спикера iBGP к спикеру iBGP. Очень часто мы видим атрибут Next Hop, являющийся IP-адресом, который не является устройством, передавшим нам обновление. Это действительно адрес, который представляет собой соседний AS, который предоставил нам префикс. Таким образом, правильно думать о BGP как о протоколе «**AS-to-AS**» вместо протокола «hop-to-hop».

Это может вызвать определенные проблемы. Основной вывод состоит в том, что вы должны гарантировать, что все ваши спикеры BGP могут достичь значения Next Hop указанного в атрибуте, чтобы путь считался допустимым. Иначе говоря, спикеры BGP будут считать префикс недопустимым, если они не смогут достичь значения Next Hop.

К счастью, эту проблему можно обойти. Вы можете взять устройство iBGP и проинструктировать его, установив себя в качестве значения Next Hop всякий раз, когда вам это нужно. Это делается с помощью манипуляции пирингом командой `neighbor`, как показано в примере 3.

```
ATL (config)# router bgp 200
ATL (config-router)# neighbor 10.10.10.1 next-hop-self
```

АТРИБУТ BGP WEIGHT (ВЕСА)

Weight (вес) - это очень интересный атрибут BGP, так как он специфичен для Cisco. Хорошая новость заключается в том, что, поскольку Cisco является гигантом в отрасли сетей, то многие другие производители будут поддерживать использование Weight в качестве атрибута.

Weight также является одним из самых уникальных атрибутов, поскольку это значение не передается другим маршрутизаторам. Weight - это значение, которое присваивается нашим префиксам как локально значимое значение. Weight - это простое число в диапазоне от 0 до 65535, и чем выше значение веса, тем выше предпочтение этого пути. Когда префикс генерируется локально, он будет иметь вес 32768. В противном случае вес префикса по умолчанию равен 0.

Как можно использовать вес? Поначалу это покажется странным, так как он не передается другим спикерам BGP. Однако все просто. Допустим, ваш маршрутизатор получает один и тот же префикс от двух разных автономных систем, с которыми он работает. Если администратор хочет предпочесть один из путей по какой-либо причине, он может манипулировать локальным значением веса на предпочтительном пути и мгновенно влиять на процесс принятия решения о наилучшем пути BGP.

BGP BEST PATH (ВЫБОР ЛУЧШЕГО ПУТИ)

Как было сказано ранее, мы знаем, что у IGP есть метрическое значение, которое является ключевым для определения наилучшего пути к месту назначения. В случае с OSPF эта метрика основана на стоимости, которая основана на пропускной способности. У BGP существует множество атрибутов пути, которые может иметь префикс. Все они поддаются алгоритму выбора наилучшего пути BGP. На рисунке 2 показаны шаги (начиная сверху), которые используются в выборе наилучших путей Cisco BGP.

Cisco BGP Best Path Selection
Highest Weight
Highest LOCAL_PREF
Prefer locally originated
Shortest AS_PATH
Lowest origin type
Lowest MED
Prefer eBGP over iBGP
Lowest IGP metric to the BGP NEXT_HOP
Oldest path
Lowest Router ID source
Minimum cluster list length
Lowest neighbor address

Изучая эти критерии выбора пути, вы можете сразу же задаться вопросом, почему он должен быть таким сложным. Помните, когда мы имеем дело с чем-то вроде интернета, мы хотим, чтобы было как можно больше регулировок для политики BGP. Мы хотим иметь возможность контролировать, насколько это возможно, как префиксы используются совместно и предпочтительно в такой большой и сложной сети.

3. Формирование соседства в BGP

Сегодня, в этой статье, вы узнаете, как формируются соседства **BGP** внутри автономной системы, между автономными системами и даже между маршрутизаторами, которые не связаны напрямую. Кроме того, мы рассмотрим аутентификацию BGP.

BGP-ПИРИНГ

Учитывая, что BGP является протоколом маршрутизации **AS-to-AS**, вполне логично, что внешний BGP (т.е. **eBGP**) является ключевым компонентом в его операциях. Самое первое, что нам нужно учитывать при работе с eBGP, - это то, что стандарты построены таким образом, что требуется прямое подключение. Это требование конечно можно обойти, но этот момент необходимо рассмотреть. Поскольку предполагается прямое соединение, протокол BGP выполняет две вещи:

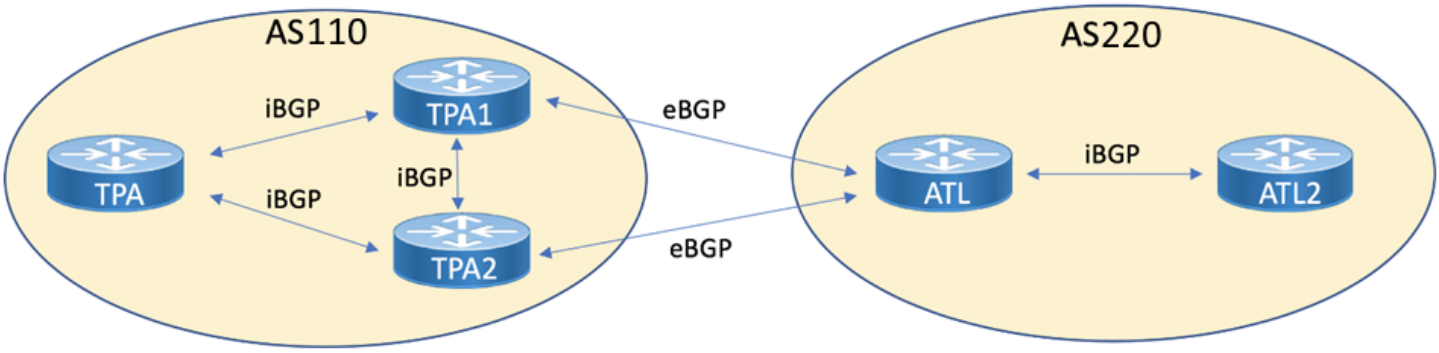
- Он будет проверять значение времени жизни (**TTL**), и что значение **time-to-live** установлено в `1`. Это означает прямую связь между одноранговыми узлами EBGP.
- Осуществляется проверка, что два устройства находятся в одной подсети.

Еще один важный момент рассмотрения пирингов eBGP - это TCP-порты, которые будут использоваться. Это особенно важно для конфигураций брандмауэров, которые защищают автономные системы. Первый спикер BGP, который инициирует изменения состояния, приходящие по мере формирования соседства, будет получать трафик из случайного TCP-порта, а конечным портом будет TCP-порт `179`. Отвечающий спикер BGP будет получать трафик с TCP-порта `179`, а порт назначения будет случайным портом. Брандмауэры должны быть перенастроены с учетом изменений в коммуникации. На основе этих изменений спикер BGP инициирует сеанс, и это, вносит изменения для будущего сеанса. Некоторые администраторы даже создают механизмы для обеспечения того, чтобы сформированные пиринги были получены из известного направления.

А как насчет **IPv6**? Ну, как было сказано ранее в предыдущей статье, BGP очень гибок и работает с IPv6, поскольку протокол был изначально спроектирован с учетом IPv6. Вы можете формировать пиринги eBGP (и iBGP) с использованием IPv6-адресации, даже если вы используются префиксы IPv4 для информации о достижимости сетевого уровня. Чтобы сформировать в нашей сети пиринг eBGP, необходимо выполнить следующие действия:

- Запустите процесс маршрутизации для BGP и укажите локальный AS (`router bgp local_as_number`).
- Предоставить удаленному спикеру eBGP IP- адрес и удаленному AS номер (`neighbor ip-_of_neighbor remote-as remote_as_number`).

Пример 1 демонстрирует конфигурацию и проверку EBGP пиринга между маршрутизаторами TPA1 и ATL.



Пример 1: Настройка пиринга eBGP

```
ATL#conf t
Enter configuration commands, one per line. End with CNTL/Z.
ATL(config)#router bgp 220
ATL(config-router)#neighbor 30.30.30.1 remote-as 110
ATL(config-router)#end
ATL#
TPA1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
TPA1(config)router bgp 110
TPA1(config-router)#neighbor 30.30.30.2 remote-as 220
TPA1(config-router)#end
TPA1#
TPA1#show ip bgp summary
BGP router identifier 30.30.30.1, local AS number 110
BGP table version is 4, main routing table version 4
1 network entries using 120 bytes of memory
1 path entries using 52 bytes of memory
1/1 BGP path/bestpath attribute entries using 124 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 320 total bytes of memory
BGP activity 2/1 prefixes, 2/1 paths, scan interval 60 secs
Neighbor    V  AS  MsgRcvd MsgSent TblVer InQ OutQ Up/Down   State/PfxRcd
30.30.30.2  4  220           413         414          4     0      0      06:12:46          1
TPA1#
```

Примечание: чтобы облегчить понимание BGP, вы можете включить функцию `debug ip bgp`, при настройке пиринга. Это позволит увидеть переходные состояния в соседстве. Кроме того, чтобы получить больше информации о соседствах, вы можете использовать команду `show ip bgp neighbors`.

Создание eBGP пиринга, на основе IPv6, выполняется также очень просто, как и на основе IPv4. Единственное изменение заключается в том, что мы заменяем адресацию в IPv4 на IPv6 и активируем соседство. Семейства адресов в маршрутизаторах Cisco для BGP позволяют запускать множество различных схем информирования о достижимости сетевого уровня (**NLRI**) в рамках одного и того же общего процесса BGP. Пример 2 демонстрирует подход к пирингу IPv6.

Пример 2: конфигурация пиринга EBGP с использованием IPv6

```
ATL#conf t
Enter configuration commands, one per line. End with CNTL/Z.
ATL(config)#router bgp 220
ATL(config-router)#neighbor 2201:1212:1212::2 remote-as 110
ATL(config-router-af)#neighbor 2201:1212:1212::2 activate
ATL(config-router-af)#end
ATL#
```

IBGP-ПИРИНГ

Если вы внимательно посмотрите на топологию, вы можете заметить, что что-то выглядит необычно. Видно, что есть iBGP-пиринг. Почему существует пиринг iBGP, созданный между TPA1 и TPA2? Это выглядит совершенно неуместно. В данном случае, как говорится, внешность может быть обманчива. Главное, что вы должны усвоить относительно BGP, является тот факт, что существует нечто, называемое правилом разделения горизонта (**Split Horizon Rule**) iBGP. Это правило гласит, что ни один спикер iBGP не может принять обновление и затем отправить это же обновление другому узлу iBGP. Так же в требовании говориться, о полном объединении наших спикеров iBGP для обеспечения полной осведомленности о префиксах.

Еще одним важным аспектом, связанным с iBGP, является избыточность. Мы хотим установить несколько физических связей между устройствами, но что произойдет, если связь, используемая для BGP, прервется? Как мы автоматически переключимся к пирингу, используя альтернативное подключение?

Простой способ решить эту проблему заключается в реализации **loopback-адресов** и использовании этих адресов для одноранговогo соединения. Это то, что мы часто делаем с нашими пирингами BGP, и это может потребовать, дополнительной настройки при использовании подключения к провайдеру. Например, в Cisco мы должны специально указать, что источником пиринга является loopback IP- адрес.

Примечание: еще одним важным аспектом при пиринге между петлевыми адресами в iBGP является то, что loopback-адреса фактически доступны между спикерами BGP. Именно здесь очень удобно использовать протокол внутреннего шлюза (IGP), такой как OSPF или EIGRP.

Пример 3 показывает конфигурацию пиринга iBGP между устройствами TPA и TPA1. Обратите внимание, что мы используем петлевой подход в том случае, если мы хотим добавить избыточные связи между устройствами в будущем.

Пример 3: Настройка пиринга iBGP

```
TPA#conf t
Enter configuration commands, one per line. End with CNTL/Z.
TPA(config)router bgp 110
TPA(config-router)#neighbor 8.8.8.8 remote-as 110
TPA(config-router)#neighbor 8.8.8.8 update-source loopback0
TPA(config-router)#end
TPA#
TPA1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
TPA1(config)#router bgp 110
TPA1(config-router)#neighbor 5.5.5.5 remote-as 110
TPA1(config-router)#neighbor 5.5.5.5 update-source loopback0
TPA1(config-router)#end
TPA1#
```

EBGP MULTIHOP

В разделе eBGP-пиринг этой статьи, обсуждалось, что ваши соседи будут связаны напрямую. В разделе iBGP мы обсуждали преимущество пиринга между loopback для избыточности. Теперь пришло время ответить на вопрос: Что делать, если ваши спикеры eBGP не подключены напрямую? На самом деле, если мы хотим пиринговать между loopback с eBGP, чтобы воспользоваться потенциальной избыточностью. Как сделать это, поскольку интерфейсы loopback не связаны напрямую друг с другом?

BGP решает эту проблему с помощью опции **eBGP multihop**. С помощью настройки eBGP multihop вы указываете максимальное количество допустимых прыжков. Это пропускает проверку BGP для TTL на значение равное 1, рассмотренное ранее в этой статье. Но как насчет требования прямого подключения? BGP отключает эту проверку в фоновом режиме автоматически, при использовании функции eBGP multihop. Пример 4 демонстрирует настройку eBGP multihop между TPA1 и ATL. Здесь нужен multihop, потому что мы настраиваем пиринг между loopback устройств.

Пример 4: eBGP Multihop

```
ATL#conf t
Enter configuration commands, one per line. End with CNTL/Z.
ATL(config)#router bgp 220
ATL(config-router)#neighbor 8.8.8.8 remote-as 110
ATL(config-router)#neighbor 8.8.8.8 update-source loopback0
ATL(config-router)#neighbor 8.8.8.8 ebgp-multihop 2
ATL(config-router)#end
ATL#
TPA1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
TPA1(config)router bgp 110
TPA1(config-router)#neighbor 7.7.7.7 remote-as 220
TPA1(config-router)#neighbor 7.7.7.7 update-source loopback0
TPA1(config-router)#neighbor 7.7.7.7 ebgp-multihop 2
TPA1(config-router)#end
TPA1#
```

BGP АУТЕНТИФИКАЦИЯ

Большинство организаций сегодня добавляют аутентификацию в свои настройки BGP, чтобы защитить их от различного рода атак. По общему признанию, аутентификацию немного сложнее настроить на BGP, чем с на других протоколах маршрутизации, поскольку конфигурация — пирингов- это ручной процесс, который должен выполнен на обоих устройствах. Даже с учетом вышесказанного, аутентификация устройств (eBGP или даже iBGP) - отличная идея.

В Cisco настройка аутентификации осуществляется просто. Необходимо задать пароль (т.е. общий секрет) на каждое устройство, настроенное для пиринга. Обязательно усвойте, что этот пароль будет отображаться в открытом виде (по умолчанию) внутри вашей сети. Можно использовать команду `service password-encryption` для выполнения по крайней мере простого шифрования тех незашифрованных текстовых паролей, которые появляются в конфигурации маршрутизатора.

Аутентификация с шифрованием **Message Digest 5 (MD5)** – это результат простого задания пароля на устройствах. Пример 5 отображает аутентификацию, добавленную в конфигурации для TPA1 и ATL.

Пример 5. Настройка аутентификации для BGP-пиринга

```
ATL#conf t
Enter configuration commands, one per line. End with CNTL/Z.
ATL(config)#router bgp 220
ATL(config-router)#neighbor 8.8.8.8 remote-as 110
ATL(config-router)#neighbor 8.8.8.8 update-source loopbackO
ATL(config-router)#neighbor 8.8.8.8 ebgp-multihop 2
ATL(config-router)#neighbor 8.8.8.8 password MySuperSecret121
ATL(config-router)#end
ATL#
TPAl#conf t
Enter configuration commands, one per line. End with CNTL/Z.
TPA1(config)router bgp 110
TPA1(config-router)#neighbor 7.7.7.7 remote-as 220
TPA1(config-router)#neighbor 7.7.7.7 update-source loopbackO
TPA1(config-router)#neighbor 7.7.7.7 ebgp-multihop 2
ATL(config-router)#neighbor 7.7.7.7 password MySuperSecret121
TPA1(config-router)#end
TPA1#
```

4. Оповещения NLRI и политики маршрутизации BGP

В этой статье мы рассмотрим настройку BGP-оповещения для **Network Layer Reachability Information (NLRI)**, а также конфигурацию политики маршрутизации BGP.

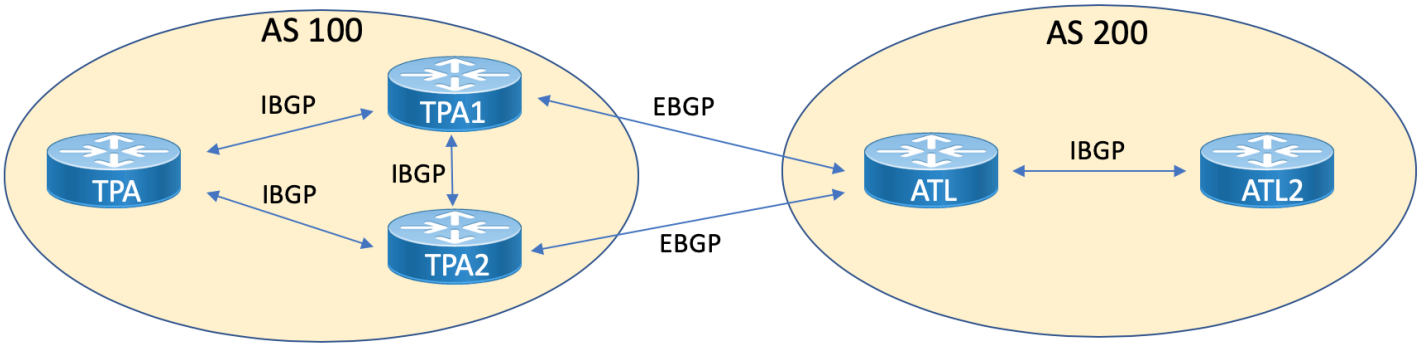
ОПОВЕЩЕНИЯ NLRI

Прежде чем мы начнем настраивать оповещения NLRI, используя различные команды, давайте сначала обсудим старую функцию BGP, которую Cisco отключает по умолчанию. Эта функция называется **синхронизацией BGP**. Для проверки того, что Cisco отключила эту функцию на вашем устройстве, выполните команду `show running-configuration` на одном из устройств BGP, и в выводимой информации, под пунктом «процессы» BGP, вы увидите сообщение `no synchronization`. Если эта функция включена, функция синхронизации не позволяет спикеру BGP вводить префиксы в BGP, если нет коррелированной записи для префикса в базовом IGP (или статических маршрутах). Это помогает предотвратить ситуации типа "черная дыра" (**black hole**), когда устройства на маршруте не работают с BGP и не могут переадресовать префикс BGP, потому что у них нет маршрута к этому префиксу из их IGP. Эта функция отключена по умолчанию из-за создания множества различных механизмов масштабируемости, существующих в BGP, которые позволяют настроить топологию iBGP без требования полной сетки одноранговых узлов iBGP. Еще одна причина, по которой он отключен, заключается в том, что он поощряет перераспределение префиксов BGP в базовый IGP, и это не безопасно.

Существует причина, по которой Cisco уходит от использования команды `network` для настройки IGPs в CLI. Не очень хорошая идея в программировании, чтобы одна команда выполняла очень разные вещи, и когда она используется в разных областях. Это относится и к команде `network`. При использовании в IGP команда включает протокол на интерфейсе (а также влияете на то, какие префиксы объявляются), но в BGP у команды `network` другое назначение. Она не включает BGP на определенных интерфейсах, вместо этого она объявляет префикс, который существует (каким-то образом) на локальном устройстве, и вводит его в BGP.

Хотя префикс, который вы могли бы объявить в BGP, чаще всего встречается в вашем IGPs в таблице маршрутизации. Вы можете использовать другие методы для создания префикса для оповещения. Например, вы можете создать интерфейс обратной связи, который обладает префиксом сети, который вы хотите объявить. Или вы можете создать статический маршрут или даже статический маршрут, указывающий на `Null0`.

Одна маленькая хитрость, связанная с командой `network` в BGP, заключается в том, что, если ваша маска подсети для вашего префикса не находится на классовой границе IP- адреса (например, `10.0.0.0/8`), то вам нужно не забыть использовать ключевое слово `mask` и указать правильную маску при использовании команды. Пример 1 показывает создание двух петлевых интерфейсов и объявление их префиксов в BGP. Обратите внимание, что этот пример также показывает проверку этих префиксных объявлений на маршрутизаторе ATL.



Пример 1: Использование команды Network в BGP

```
TPA1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
TPA1(config)#interface loopback 192
TPA1(config-if)#ip address 192.168.1.1 255.255.255.0
TPA1(config-if)#exit
TPA1(config)#interface loopback 172
TPA1(config-if)#ip address 172.16.10.1 255.255.255.0
TPA1(config-if)#exit
TPA1(config)router bgp 100
TPA1(config-router)#network 192.168.1.0
TPA1(config-router)#network 172.16.10.0 mask 255.255.255.0
TPA1(config-router)#end
TPA1#
ATL#
ATL#show ip bgp
```

```
TPA1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
TPA1(config)#interface loopback 192
TPA1(config-if)#ip address 192.168.1.1 255.255.255.0
TPA1(config-if)#exit
TPA1(config)#interface loopback 172
TPA1(config-if)#ip address 172.16.10.1 255.255.255.0
TPA1(config-if)#exit
TPA1(config)#router bgp 100
TPA1(config-router)#network 192.168.1.0
TPA1(config-router)#network 172.16.10.0 mask 255.255.255.0
TPA1(config-router)#end
TPA1#
ATL#
ATL#show ip bgp
BGP table version is 5, local router ID is 100.100.100.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
        r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.10.0/24   10.10.10.1               0             0 100 i
*> 192.168.1.0      10.10.10.1               0             0 100 i
ATL#
```

Хотя команда `network` проста и удобна, она не была бы эффективной, если бы у вас было много префиксов для оповещения. Другой вариант- перераспределить префиксы в BGP из IGP или статических маршрутов. Пример 2 демонстрирует перераспределение префиксов, которые были получены через EIGRP, в BGP. Обратите внимание при проверке, что исходный код для этих префиксов отображается как `(?)` указывает на неизвестность.

Пример 2: перераспределение префиксов в BGP

```
TPA1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
TPA1(config)router bgp 100
TPA1(config-router)#redistribute eigrp 100
TPA1(config-router)#end
TPA1#
ATL#show ip bgp
```

```
TPA1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
TPA1(config)#router bgp 100
TPA1(config-router)#redistribute eigrp 100
TPA1(config-router)#end
TPA1#
ATL#show ip bgp
BGP table version is 9, local router ID is 100.100.100.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
        r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.10.0/24   10.10.10.1               0             0 100 ?
*> 192.168.1.0      10.10.10.1               0             0 100 ?
ATL#
```

Когда вы начинаете объявлять (оповещать) NLRI в BGP, вы можете столкнуться с префиксами в вашей таблице BGP (показанной с `show ip bgp`), которые имеют код состояния `(r)` вместо ожидаемого допустимого кода состояния `(*)`. Код состояния `(r)` указывает на сбой **RIB**, означающий, что BGP попытался поместить префикс в таблицу BGP, но не смог из- за какой-то проблемы.

Наиболее распространенной причиной отказа RIB является административное расстояние (**AD**). Например, IBGP узнал префиксы несущие ужасные объявления AD из 200. Это означает, что если ваш маршрутизатор получил префикс через IGP (даже такой плохой, как RIP с AD 120), то он будет предпочтительнее префикса IBGP. В результате протокол BGP получивший это объявление AD, не отметит префикс как действующий. Обратите внимание, что это, как правило, не происходит с префиксами **EBGP-learned**, поскольку они имеют очень предпочтительное объявление 20 (по умолчанию).

Очень часто, если желательно иметь префикс в IGP и BGP, администраторы будут манипулировать значениями AD на своих маршрутизаторах, чтобы улучшить AD IBGP. Например, в случае RIP и BGP администратор мог бы установить AD изученных маршрутов IBGP на 119, чтобы сделать их предпочтительными по сравнению с используемым IGP.

В дополнение к выявлению сбоев RIB в результатах команды `show ip bgp`, вы можете использовать более прямую команду `show ip bgp rib-failure`, чтобы увидеть любые префиксы в этом состоянии. Это особенно полезно в случае массивных таблиц BGP.

НАСТРОЙКА ПОЛИТИКИ МАРШРУТИЗАЦИИ BGP

Довольно часто встречаются топологии, в которых вы явно не хотите объявлять префиксы в своей таблице BGP, или вы не хотите получать определенные префиксы от узла BGP. К счастью, в вашем распоряжении есть много инструментов для этого. Например, вот только некоторые методы, которые вы могли бы использовать для фильтрации префиксов:

- Distribute lists
- Extended ACLs
- Prefix lists
- AS Path filters
- Route maps

Пример 3 демонстрирует один из методов фильтрации. Выбран подход `route map`, потому что все (и это правильно) любят карты маршрутов.

Пример 3: Использование route map в качестве префиксного фильтра в BGP

```
ATL# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ATL(config)#ip access-list standard MYPREFIX
ATL(config-std-nacl)#permit 192.168.1.0 0.0.0.255
ATL(config-std-nacl)#exit
ATL(config)#route-map MYMAP deny 10
ATL(config-route-map)#match ip address MYPREFIX
ATL(config-route-map)#exit
ATL(config)#route-map MYMAP permit 20
 ATL(config-route-map)#exit
ATL(config)#router bqp 200
ATL(config-router)#neighbor 10.10.10.1 route-map MYMAP in
ATL(config-router)#end
ATL#
ATL# clear ip bqp * soft
ATL# show ip bqp
```

```
ATL#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ATL(config)#ip access-list standard MYPREFIX
ATL(config-std-nacl)#permit 192.168.1.0 0.0.0.255
ATL(config-std-nacl)#exit
ATL(config)#route-map MYMAP deny 10
ATL(config-route-map)#match ip address MYPREFIX
ATL(config-route-map)#exit
ATL(config)#route-map MYMAP permit 20
ATL(config-route-map)#exit
ATL(config)#router bgp 200
ATL(config-router)#neighbor 10.10.10.1 route-map MYMAP in
ATL(config-router)#end
ATL#
ATL#clear ip bgp * soft
ATL#show ip bgp
BGP table version is 10, local router ID is 100.100.100.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
        r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.10.0/24   10.10.10.1               0             0 100 ?
ATL#
```

Обратите внимание, перед проверкой я запускаю команду `clear ip bgp * soft`. Это гарантирует, что устройство сразу же обновит информацию BGP для меня, так что мне не придется ждать истечения таймера, когда дело дойдет до конвергенции BGP на новых манипуляциях с политикой, которые мы сделали.

Помните, что BGP использует множество различных атрибутов пути вместо простой метрики, чтобы предоставить вам возможность легко настроить способ, по которому происходит маршрутизация. Ниже приведены некоторые из атрибутов пути, которыми вы могли бы манипулировать, чтобы настроить политику:

- Weight
- MED
- Local Preference
- AS Path

Можно спросить себя, как AS Path могут быть использованы в целях маршрутизации. Поскольку манипуляция AS Path часто выполняется с помощью **AS Path Prepending**. Вы отравляете префикс, добавляя свой собственный номер AS к пути, чтобы сделать более длинным (менее предпочтительным) AS Path. Как и большинство наших манипуляций с атрибутом пути, это легко сделать с помощью карты маршрута.

Давайте рассмотрим пример использования **Local Preference** для манипулирования политикой. Мы часто используем Local Preference, чтобы повлиять на то, как мы будем направлять исходящий трафик к префиксу BGP. Мы делаем это, устанавливая значения Local Preference, входящие по нескольким путям. Прежде чем мы начнем, поймите, что Local Preference - это значение, которое рассматривается довольно высоко в процессе принятия решения о наилучшем пути BGP, более высокое значение предпочтительно, и значения передаются только в обновлениях IBGP. Именно так имя LOCAL вошло в название Local Preference.

Для начала я объявил тот же префикс в AS 200 (ATL и ATL2) от маршрутизаторов TPA1 и TPA2 AS 100. Глядя на пример 4, Вы можете видеть, что этот префикс (192.168.1.0) может быть достигнут с помощью следующего прыжка 10.10.10.1 и что это предпочтительный путь. Альтернативный путь, который будет использоваться в случае неудачи этого пути, будет проходить через следующий переход 10.21.21.1.

Пример 4: Подготовка к использованию Local Preference

```
ATL# show ip bqp
```

```
ATL#show ip bgp
BGP table version is 12, local router ID is 100.100.100.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
        r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  192.168.1.0      10.21.21.1               0             0 100 i
*>                  10.10.10.1               0             0 100 i
ATL#
```

Теперь пришло время поэкспериментировать и изменить данное поведение с помощью примера манипуляции атрибутом пути. Мой подход будет состоять в том, чтобы определить префикс, которым мы хотим манипулировать (192.168.1.0), и поднять значение локального предпочтения, чтобы оно было больше, чем значение по умолчанию 100 для пути к TPA2 на следующем прыжке 10.21.21.1. Я делаю это, манипулируя префиксом, когда он входит через путь 10.21.21.1 .

Пример 5 показывает эту конфигурацию.

```
ATL# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
 ATL(config)#ip access-list standard OURPREFIX
ATL(config-std-nacl)#permit 192.168.1.0 0.0.0.255
ATL(config-std-nacl)#exit
ATL(config)#route-map SETLOCALPREF permit 10
ATL(config-route-map)#match ip address OURPREFIX
 ATL(config-route-map)#set local-preference 110
ATL(config-route-map)#exit
ATL(config)#route-map SETLOCALPREF permit 20
ATL(config-route-map)#exit
ATL(config)#router bqp 200
ATL(config-router)#neighbor 10.21.21.1 route-map SETLOCALPREF in
ATL(config-router)#end ATL#
ATL# clear ip bqp * soft
ATL# show ip bqp
```

```
ATL#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ATL(config)#ip access-list standard OURPREFIX
ATL(config-std-nacl)#permit 192.168.1.0 0.0.0.255
ATL(config-std-nacl)#exit
ATL(config)#route-map SETLOCALPREF permit 10
ATL(config-route-map)#match ip address OURPREFIX
ATL(config-route-map)#set local-preference 110
ATL(config-route-map)#exit
ATL(config)#route-map SETLOCALPREF permit 20
ATL(config-route-map)#exit
ATL(config)#router bgp 200
ATL(config-router)#neighbor 10.21.21.1 route-map SETLOCALPREF in
ATL(config-router)#end
ATL#
ATL#clear ip bgp * soft
ATL#show ip bgp
BGP table version is 13, local router ID is 100.100.100.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
        r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      10.21.21.1               0    110      0 100 i
*                   10.10.10.1               0             0 100 i
ATL#
```

Обратите внимание, что предпочтительный путь теперь проходит через следующий переход 10.21.21.1, как мы и хотели. Для этого префикса также отображается значение Local Preference - 110. Это более высокое значение является предпочтительным и изменяет выбор, сделанный процессом выбора наилучшего пути BGP.

5. Масштабируемость протокола BGP

В этой статье мы рассмотрим механизмы масштабируемости **BGP** и связанные с ними концепции.

МЕХАНИЗМЫ МАСШТАБИРУЕМОСТИ BGP

Истощение доступных автономных системных номеров явилось проблемой точно так же, как было проблемой для интернета истощение IP-адресов. Чтобы решить эту проблему, инженеры обратились к знакомому решению.

Они обозначили диапазон номеров AS только для частного использования. Это позволяет вам экспериментировать с AS конструкцией и политикой, например, в лаборатории и использовать числа, которые гарантированно не конфликтуют с интернет-системами.

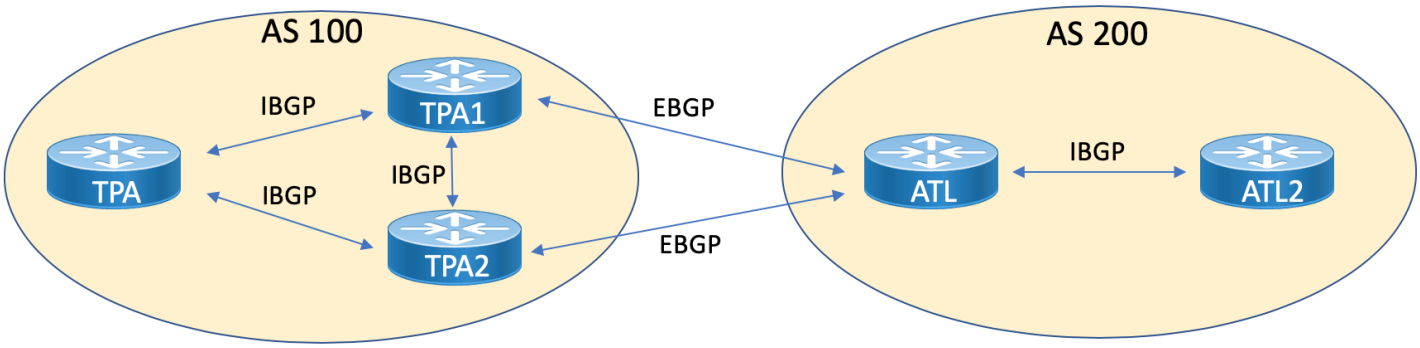
Помните, что число AS-это 16-разрядное число, допускающее до 65 536 чисел AS. Диапазон для частного использования: `64512-65535`.

Еще одним решением проблемы дефицита, стало расширение адресного пространства имен. Было утверждено пространство, представляющее собой 32-разрядное число.

В течение длительного времени, с точки зрения масштабируемости, одноранговые группы Border Gateway Protocol считались абсолютной необходимостью. Мы настраивали одноранговые группы для уменьшения конфигурационных файлов. Так же мы настраивали одноранговые группы для повышения производительности.

Преимущества производительности были нивелированы с помощью значительно улучшенных механизмов, сейчас. Несмотря на это, многие организации все еще используют одноранговые группы, поскольку они поняты и легки в настройке.

Появились в BGP одноранговые группы для решения нелепой проблемы избыточности в BGP конфигурации. Рассмотрим простой (и очень маленький) пример 1. Даже этот простой пример отображает большое количество избыточной конфигурации.



Пример 1: типичная конфигурация BGP без одноранговых групп

```
ATL1(config)#router bgp 200
ATL1( config-router)#neiqhbor 10.30.30.5 remote-as 200
ATL1( config-router)#neiqhbor 10.30.30.5 update- source lo0
ATL1( config= router)#neiqhbor 10.30 .30.5 password S34Dfr112s1WP
ATL1(config-router)#neiqhbor 10.40.40.4 remote-as 200
ATL1( config-router)#neiqhbor 10.40.40 .4 update- source lo0
ATL1(config-router)#neiqhbor 10.40.40.4 password S34Dfr112s1WP
```

Очевидно, что все команды настройки относятся к конкретному соседу. И многие из ваших соседей будут иметь те же самые характеристики. Имеет смысл сгруппировать их настройки в одноранговую группу. Пример 2 показывает, как можно настроить и использовать одноранговую группу BGP.

Пример 2: одноранговые группы BGP

```
ATL2 (config)#router bgp 200
ATL2 (config-router)#neighbor MYPEERGR1 peer-group
ATL2 (config-router)#neighbor MYPEERGR1 remote-as 200
ATL2 (config-router)#neighbor MYPEERG1l update-source lo0
ATL2(config-router)#neighbor MYPEERGRl next-hop-self
ATL2 (config-router)#neighbor 10.40.40 .4 peer-group MYPEERGR1
ATL2 (config-router)#neighbor 10.50.50 .5 peer-group MYPEERGR1
```

Имейте в виду, что, если у вас есть определенные настройки для конкретного соседа, вы все равно можете ввести их в конфигурацию, и они будут применяться в дополнение к настройкам одноранговой группы. Почему же так часто использовались одноранговые группы? Они улучшали производительность. Собственно говоря, это и было первоначальной причиной их создания.

Более современный (и более эффективный) подход заключается в использовании шаблонов сеансов для сокращения конфигураций. А с точки зрения повышения производительности теперь у нас есть (начиная с iOS 12 и более поздних версий) динамические группы обновлений. Они обеспечивают повышение производительности без необходимости настраивать что-либо в отношении одноранговых групп или шаблонов.

Когда вы изучаете одноранговую группу, вы понимаете, что все это похоже на шаблон для настроек. И это позволит вам использовать параметры сеанса, а также параметры политики. Что ж, новая и усовершенствованная методология разделяет эти функциональные возможности на шаблоны сессий и шаблоны политики.

Благодаря шаблонам сеансов и шаблонам политик мы настраиваем параметры, необходимые для правильной установки сеанса, и помещаем эти параметры в шаблон сеанса. Те параметры, которые связаны с действиями политик, мы помещаем в шаблон политики.

Одна из замечательных вещей в использовании этих шаблонов сеансов или политик, а также того и другого, заключается в том, что они следуют модели наследования. У вас может быть шаблон сеанса, который выполняет определенные действия с сеансом. Затем вы можете настроить прямое наследование так, чтобы при создании другого наследования оно включало в себя вещи, созданные ранее. Эта модель наследования дает нам большую гибкость, и мы можем создать действительно хорошие масштабируемые проекты для реализаций BGP.

Вы можете использовать шаблоны или одноранговые группы, но это будет взаимоисключающий выбор. Так что определитесь со своим подходом заранее. Вы должны заранее определиться, что использовать: использовать ли устаревший подход одноранговых групп или же использовать подход шаблонов сеанса и политики. После выбора подхода придерживайтесь его, так как, использовать оба подхода одновременно нельзя.

Теперь можно предположить, что конфигурация для шаблонов сеансов будет довольно простой, и это так. Помните, прежде всего, все что мы делаем здесь и сейчас, относится к конкретной сессии. Поэтому, если мы хотим установить `timers`, нам нужно установить `remote-as` – и это будет считается параметром сеанса.

Например, мы делаем `update source`. Мы настраиваем eBGP multihop. Все это имеет отношение к текущему сеансу, и именно это мы будем прописывать в шаблоне сеанса. Обратите внимание, что мы начинаем с создания шаблона. Поэтому используем команду `template peer-session`, а затем зададим ему имя. И тогда в режиме конфигурации шаблона можем настроить наследование, которое позволит наследовать настройки от другого однорангового сеанса. Можем установить наш remote-as как и/или `update source`. После завершения, мы используем команду `exit-peer-session`, чтобы выйти из режима конфигурации для этого сеанса. Пример 3 показывает конфигурацию шаблона сеанса.

Пример 3: Шаблоны сеансов BGP

```
ATL2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
ATL2 (config)#router bgp 200
ATL2 (config-router)#template peer- session MYNAME
ATL2 (config-router-stmp)#inherit peer- session MYOTHERNAME
ATL2 (config- router-stmp )#remote-as 200
ATL2(config-router-stmp )#password MySecrectPass123
ATL2 (config-router-stmp )#exit-peer-session
ATL2 (config-router)#neiqhbor 10.30.30 .10 inherit peer-session MYNAME
ATL2 (config-router)#end
ATL2#
```

Это простой пример настройки соседства с помощью оператора neighbor и использования наследования одноранговой сеанса. Затем присваивается имя одноранговой сеанса, созданного нами для нашего шаблона сеанса. Это соседство наследует параметры сеанса.

Помните, что, если вы хотите сделать дополнительную настройку соседства, можно просто присвоить соседу IP-адрес, а затем выполнить любые настройки вне шаблона одноранговой сеанса, которые вы хотите дать этому соседу. Таким образом, у вас есть та же гибкость, которую мы видели с одноранговыми группами, где вы можете настроить индивидуальные параметры для этого конкретного соседа вне шаблонного подхода этого соседства.

Вы можете подумать, что шаблоны политик будут иметь сходную конструкцию и использование с шаблонами сеансов, и вы будете правы. Помните, что если ваши шаблоны сеансов находятся там, где мы собираемся настроить параметры, которые будут относиться к сеансу BGP, то, конечно, шаблоны политик будут храниться там, где мы храним параметры, которые будут применяться к политике.

Пример 4 показывает настройку и использование шаблона политики BGP.

Пример 4: Шаблоны политики BGP

```
ATL2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
ATL2 (config)#router bgp 200
ATL2(config-router)#template peer-policy MYPOLICYNAME
ATL2 (config-router-ptmp )#next-hop-self
ATL2 (config-router-ptmp )#route-map MYMAP out
ATL2 (config-router-ptmp )#allowas-in
ATL2 (config-router-ptmp )#exit-peer-policy
ATL2 (config-router)#neighbor 10.40.40.10 remote-as 200
ATL2 (config-router)#neighbor 10.40.40.10 inherit peer-policy MYNAME
ATL2 (config-router)#end
ATL2#
```

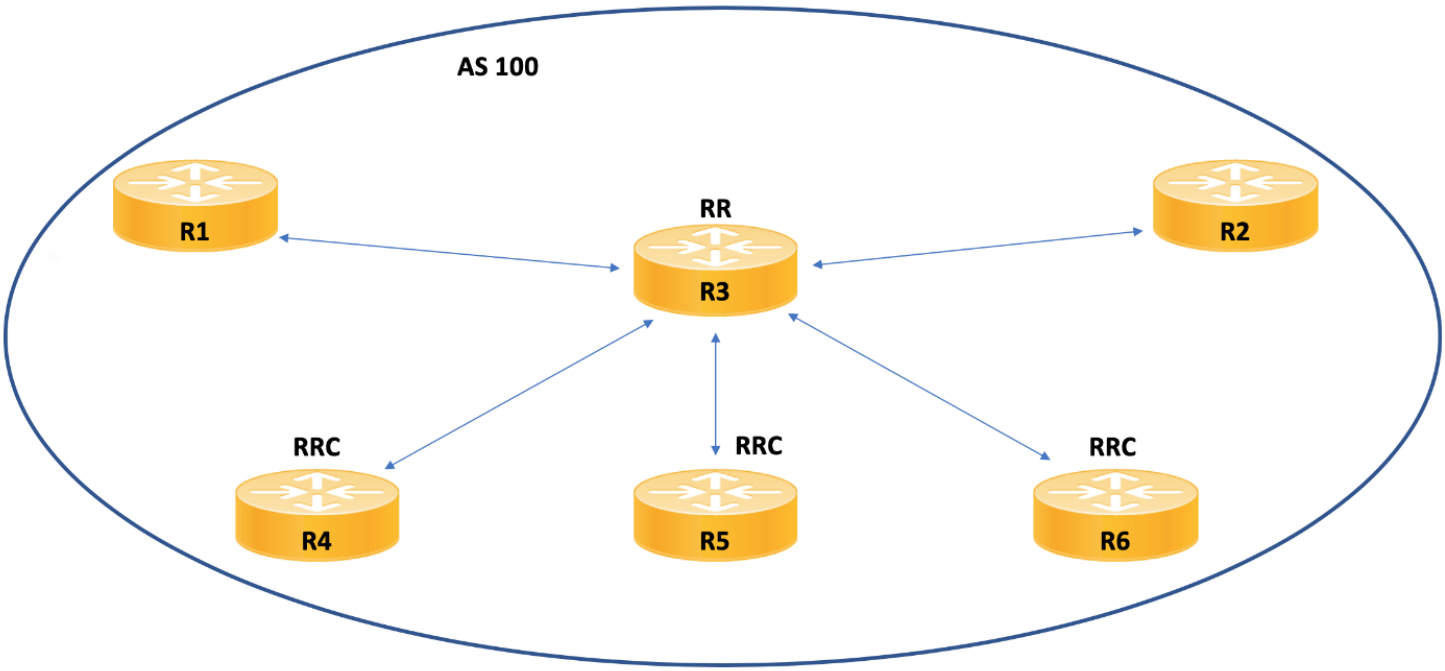
Да, все эти параметры, которые мы обсуждали при изучении манипуляций с политикой, будут тем, что мы будем делать внутри шаблона политики. Однако одним из ключевых отличий между нашим шаблоном политики и шаблоном сеанса является тот факт, что наследование здесь будет еще более гибким.

Например, мы можем перейти к семи различным шаблонам, от которых мы можем непосредственно наследовать политику. Это дает нам еще более мощные возможности наследования с помощью шаблонов политик по сравнению с шаблонами сеансов.

Опять же, если мы хотим сделать независимые индивидуальные настройки политики для конкретного соседа, мы можем сделать это, добавив соответствующие команды соседства.

Благодаря предотвращению циклов и правилу разделения горизонта (**split-horizon rule**) IBGP, среди прочих факторов, нам нужно придумать определенные решения масштабируемости для пирингов IBGP. Одним из таких решений является **router reflector**.

Рис. 1: Пример топологии router reflector



Конфигурация router reflector удивительно проста, поскольку все это обрабатывается на самом router reflector (R3). Клиенты route reflector – это R4, R5 и R6. Они совершенно не знают о конфигурации и настроены для пиринга IBGP с R3 как обычно. Пример 5 показывает пример конфигурации router reflector. Обратите внимание, что это происходит через простую спецификацию клиента router reflector.

Пример 5: BGP ROUTE REFLECTOR

```
R3#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
 R3 (config)#router bgp 200
R3 (config-router)#neighbor 10.50.50.10 remote -as 200
R3 (config-router)#neighbor 10.50.50.10 route-reflector-client
R3 (config-router)#end
R3#
```

Route reflector автоматически создает значение идентификатора (**ID**) кластера для кластера, и это устройство и эти клиенты будут частью того, что мы называем кластером route reflector. Cisco рекомендует разрешить автоматическое назначение идентификатора кластера для идентификации клиента. Это 32-разрядный идентификатор, который BGP извлекает из route reflector.

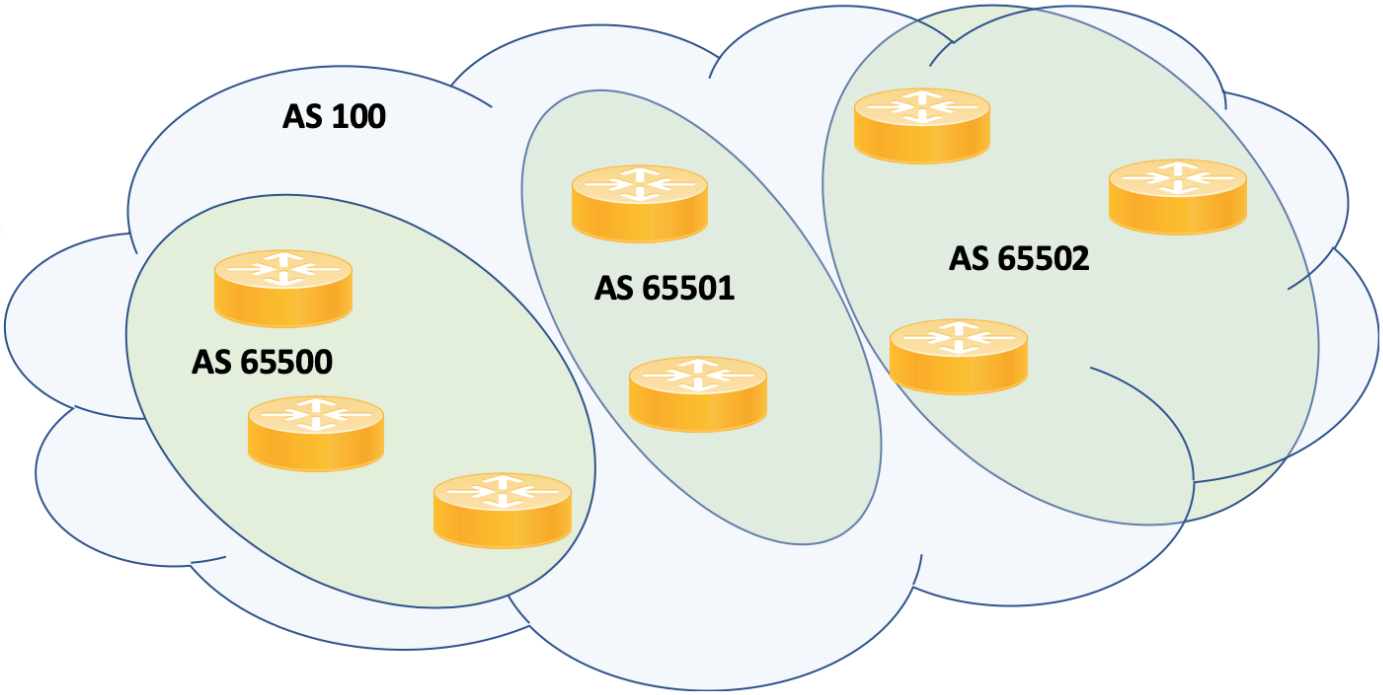
Магия Route reflector заключается в том, как меняются правила IBGP. Например, если обновление поступает от клиента Route reflector (скажем, R4), то устройство R3 «отражает» это обновление своим другим клиентам (R5 и R6), а также своим неклиентам (R1 и R2). Это обновление происходит даже при том, что конфигурация для IBGP значительно короче полной сетки пирингов, которая обычно требуется.

А теперь что будет, если обновление поступит от не клиента Route reflector (R1)? Route reflector отправит это обновление всем своим клиентам Route reflector (R4, R5 и R6). Но тогда R3 будет следовать правилам IBGP, и в этом случае он не будет отправлять обновление через IBGP другому не клиенту Route reflector (R2).

Чтобы решить эту проблему, необходимо будет создать пиринг от R1 к устройству R2 с помощью IBGP. Или, можно добавить R2 в качестве клиента Route reflector R3.

Есть еще один способ, которым мы могли бы решить проблему с масштабируемостью IBGP- это манипулирование поведением EBGP. Мы делаем это с конфедерациями. Вы просто не замечаете, что конфедерации используются так же часто, как Route reflector. И причина состоит в том, что они усложняют нашу топологию, и делают поиск неисправностей более сложным. На рис. 2 показан пример топологии конфедерации.

Рисунок 2: Пример топологии конфедерации



Мы имеем наш AS 100. Для создания конфедерации необходимо создать небольшие субавтономные системы внутри нашей основной автономной системы. Мы их пронумеруем с помощью, номеров автономных систем только для частного использования.

Что мы имеем, когда манипулируем поведением eBGP, что бы имеет конфедерацию EBGP пирингов? Это позволяет нам установить пиринги между соответствующими устройствами, которые хотим использовать в этих автономных системах. Как вы можете догадаться, они не будут следовать тем же правилам, что и наши стандартные пиринги EBGP. Еще один важный момент заключается в том, что все это для внешнего неконфедеративного мира выглядит просто как единый AS 100.

Внутри мы видим реальные AS, и конфедеративные отношения EBGP между ними. Помимо устранения проблемы разделения горизонта IBGP, что же меняется с пирингами конфедерации EBGP? В следующем прыжке поведение должно измениться. Следующий прыжок не меняется тогда, когда мы переходим от одной из этих небольших конфедераций внутри нашей AC к другой конфедерации.

Вновь добавленные атрибуты обеспечивают защиту от цикла из-за конфедерации.
Атрибут `AS_confed_sequence` и `AS_confed_set` используются в качестве механизмов предотвращения циклов.

Пример 6 показывает пример частичной настройки конфедерации BGP.

```
R3#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R3 (config)#router bgp  65501
R3(config-router)#bgp confederation identifier 100
R3 (config-router)#bgp confederation peers 65502
R3 (config-router)#neighbor 10 .20.20.1 remote-as 65502
R3 (config-router)#end
R3#
```

Иногда возникает необходимость применения общих политик к большой группе префиксов. Это делается легко, если вы помечаете префиксы специальным значением атрибута, называемым сообществом (**community**). Обратите внимание, что сами по себе атрибуты сообщества ничего не делают с префиксами, кроме как прикрепляют значение идентификатора. Это 32-разрядные значения (по умолчанию), которые мы можем именовать, чтобы использовать дополнительное значение.

Вы можете настроить значения сообщества таким образом, чтобы они были значимы только для вас или значимы для набора AS. Вы также можете иметь префикс, который содержит несколько значений атрибутов сообщества. Кроме того, можно легко добавлять, изменять или удалять значения сообщества по мере необходимости в вашей топологии BGP.

Атрибуты сообщества могут быть представлены в нескольких форматах. Более старый формат выглядит следующим образом:

- Decimal - 0 to 4294967200 (в десятичном)
- Hexadecimal – 0x0 to 0xffffffa0 (в шестнадцатеричном)

Более новый формат:

- AA:NN

AA - это 16-битное число, которое представляет ваш номер AS, а затем идет 16-битное число, используемое для задания значимости своей политике AS. Таким образом, вы можете задать для AS 100 100:101, где 101- это номер внутренней политики, которую вы хотите применить к префиксам.

Есть также хорошо известные общественные значения. Это:

- No-export - префиксы не объявляются за пределами AS. Вы можете установить это значение, когда отправляете префикс в соседний AS. чтобы заставить его (соседний AS) не объявлять префикс за собственные границы.
- Local-AS - префиксы с этим атрибутом сообщества никогда не объявляются за пределами локального AS
- No-advertise - префиксы с этим атрибутом сообщества не объявляются ни на одном устройстве

Эти хорошо известные атрибуты сообщества просто идентифицируются по их зарезервированным именам.

Есть также расширенные сообщества, которые также можно использовать. Они предлагают 64-битную версию для идентификации сообществ! Задание параметров осуществляется настройкой TYPE:VALUE. Выглядит оно следующим образом:

- 65535:4294967295

Как вы можете догадаться, мы устанавливаем значения сообщества, используя **route maps**. Пример 7 показывает пример настроек. Обратите внимание, что в этом примере также используется список префиксов. Они часто используются в BGP для гибкой идентификации многих префиксов. Они гораздо более гибки, чем списки доступа для этой цели.

Пример 7: Установка значений сообщества в BGP

```
R3#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R3(config)#ip prefix-list MYLIST permit 172.16.0.0/16 le 32
R3(config)#route-map SETCOMM permit 10
R3(config-route-map)#match ip address prefix-list MYLIST
R3(config-route-map)#set community no-export
R3(config-route-map)#route-map SETCOMM permit 20
R3(config)#router bgp 100
R3(config-router)#neighbor 10.20.20.1 route-map SETCOMM out
R3 (config-router)#neighbor 10.20.20.1 send-community
R3(config-router)#end
R3#
```

6. Работа протокола BGP с IPv6

Вы помните из прошлых статей, что BGP был создан для поддержки многих различных протоколов и **NLRI** непосредственно с момента его возникновения. В результате чего BGP поддерживает такие технологии, как **IPV6**, **MPLS**, **VPN** и многое другое.

Вы будете приятно удивлены тем, что как только вы овладеете основами BGP, которые мы рассмотрели в этом цикле статей, работа с BGP в IPv6 покажется очень простой!

BGP С IPV6

BGP настолько удивительно гибок, что, как обсуждалось ранее в этом цикле статей, можно использовать IPv4 в качестве «несущего» протокола для **IPv6 NLRI**. В данном случае мы рассматриваем IPv6 как «пассажирский» протокол. Давайте сначала рассмотрим конфигурацию и используем два простых маршрутизатора, как показано на рисунке 1.

Рисунок 1: Простая топология для IPv6 протокола BGP

Пример 1 показывает конфигурацию и проверку такой сети. Обратите внимание, что эта конфигурация требует установки соответствующего адреса следующего прыжка IPv6 для префиксного объявления. Это не требуется при использовании IPv6 как протокола «перевозчика», так и протокола «пассажира».

Пример 1: IPv4 «перевозящий» IPv6 NLRI

```
ATL#conf t
Enter configuration commands, one per line. End with CNTL/Z.
ATL( config)#ipv6 unicast-routing
ATL(config)#route-map IPV6NH permit 10
ATL(config-route-map)#set ipv6 next-hop 2001:1212:1212::1
ATL(config-route-map)#exit
ATL(config)#int lo 100
ATL(config-if)#ipv6 address 2001:1111:1111: :/64 eui-64
ATL(config-if )#router bqp 200
ATL(config-router)#neiqhbor 10.10.10.2 remote-as 200
 ATL(config-router) #address-family ipv4 unicast
ATL(config-router-af)#neiqhbor 10.10.10.2 activate
ATL(config-router-af)#address-family ipv6 unicast
ATL(config-router-af)#neiqhbor 10.10.10.2 activate
ATL(config-router-af)#neiqhbor 10.10.10.2 route-map IPV6NH out
ATL(config-router-af)#network 2001:1111:1111: :/64
ATL(config-router-af)#end
ATL#
```

Пример 2 показывает проверку этой конфигурации на ATL 2. Обратите внимание, что поскольку `EUI-64` действует на интерфейсе обратной связи ATL, вам нужно будет скопировать полный IPv6-адрес из этого интерфейса, чтобы выполнить тестирование командой `ping`.

Пример 2: проверка настройки BGP IPv4/IPv6

```
ATL#show ip bgp ipv6 unicast
```

```
ATL2#show ip bgp ipv6 unicast
BGP table version is 2, local router ID is 10.10.10.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
    r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
    Network          Next Hop            Metric LocPrf Weight Path

*>i2001:1111:1111::/64
                     2001:1212:1212::1
                                              0    100      0 i
ATL2#ping 2001:1111:1111:0:C801:6FF:FEDB:0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:1111:1111:0:C801:6FF:FEDB:0,
timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/15/24 ms
ATL2#
```

```
ATL2#ping 2001:1111:1111:0:C801:6FF:FEDB:0
```

```
ATL2#show ip bgp ipv6 unicast
BGP table version is 2, local router ID is 10.10.10.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
    r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
    Network          Next Hop            Metric LocPrf Weight Path

*>i2001:1111:1111::/64
                    2001:1212:1212::1
                                             0    100      0 i
ATL2#ping 2001:1111:1111:0:C801:6FF:FEDB:0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:1111:1111:0:C801:6FF:FEDB:0,
timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/15/24 ms
ATL2#
```

Как вы можете догадаться, гораздо более «чистая» конфигурация заключается в использовании IPv6 для передачи информации IPv6 префикса. «Чистая» - это имеется в виду гораздо простая конфигурация. Пример 3 демонстрирует эту конфигурацию. Обратите внимание, что были удалены все IPv4 с устройств, поэтому необходимо установить 32-битный **router ID** для BGP, поскольку он не может установить его автоматически из интерфейса на устройстве.

Пример 3: проверка настройки BGP IPv4/IPv6

```
ATL1#conf t
ATL1(config)#router bgp 200
*Jan 9 03:31:21.039: %BGP-4-NORTRID: BGP could not pick a router-id.
Please configure manually.
ATL1(config-router)#bgp router-id 1.1.1.1
ATL1(config-router)#neighbor 2001:1212:1212::2 remote-as 200
ATL1(config-router)#address-family ipv6 unicast
ATL1(config-router-af)#neighbor 2001:1212:1212::2 activate
ATL1(config-router-af)#network 2001:1111:1111::/64
ATL1(config-router-af)#end
ATL1#
```

Возможно, вам будет интересно проверить соседство BGP после настройки IPv6. Мы очень любим использовать команду `show ip bgp summary` для проверки настроек в IPv4. Для IPv6 используйте команду `show bgp ipv6 unicast summary`.

Как вы помните из предыдущей части этой серии статей, существует много замечательных механизмов фильтрации, которые мы можем применить в IPv4 BGP. Замечательная новость заключается в том, что этот же набор методов, доступны и для IPv6. Ментоды включают в себя такие механизмы, как:

- Prefix lists
- AS Path Filtering
- Route maps

Пример 4 показывает пример конфигурации фильтрации с использованием списка префиксов. Обратите внимание, что эта конфигурация действительно не требует от вас повторного изучения каких-либо технологий.

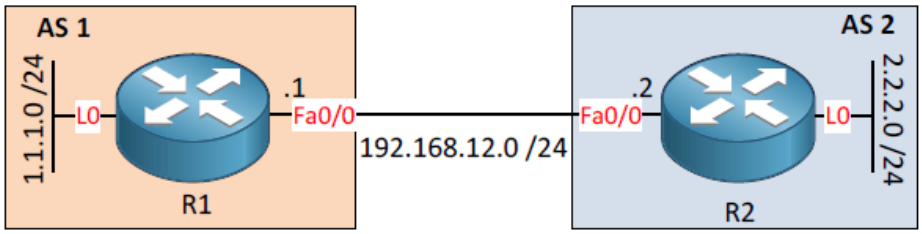
Пример 4: фильтрация префиксов IPv6 в BGP

```
ATL#conf t
ATL(config)#ipv6 prefix-list MYTEST deny 2001:1111:1111::/64
ATL(config)#ipv6 prefix-list MYTEST permit ::/0 le 128
ATL(config)#router bgp 200
ATL(config-router)#address-family ipv6 unicast
ATL(config-router-af)#neighbor 2001:1212:1212:: 2 prefix-list MYTEST out
ATL(config-router-af)#end
ATL#
ATL#clear ip bgp *
```

7. Траблшутинг BGP

BGP - это сложный протокол маршрутизации, и бывают ситуации, когда что-то идет не так как надо. Кроме того, что он сложный, он также совершенно отличается от наших IGP протоколов (OSPF и EIGRP). В этой статье мы начнем с рассмотрения неполадок, возникающих в установлении соседства BGP, и как только это разберем, перейдем к проблемам с объявлением маршрутов, которые должны или не должны появляться!

УРОК 1



Начнем с нескольких простых сценариев. Два маршрутизатора BGP, которые подключены и настроены для EBGP. К сожалению, мы видим это, когда проверяем соседство BGP:

```
R1#show ip bgp summary

BGP router identifier 192.168.12.1, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.12.2    4     2       8       8        0    0    0 00:00:06        Active
```

```
R2#show ip bgp summary

BGP router identifier 192.168.12.2, local AS number 2
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.12.1    4     1       8       8        0    0    0 00:00:59        Active
```

Когда два маршрутизатора EBGP, которые напрямую подключены, не образуют рабочее соседство BGP, может произойти ряд ошибок:

- Layer 2 не позволяет нам добраться до другой стороны.
- Проблема уровня 3: неправильный IP-адрес на одном из маршрутизаторов.
- Список доступа, блокирующий TCP-порт 179 (BGP).
- Неправильный IP-адрес настроен для соседнего маршрутизатора BGP

Мы можем использовать команду `show ip bgp summary`, чтобы проверить IP-адреса маршрутизаторов. Они, совпадают.

```
R1#ping 192.168.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Мы выполним эхо запрос, с помощью команды ping. Видим, что, пакеты не могут добраться до другой стороны.

```
R1#show ip int brief

Interface              IP-Address      OK? Method  Status           Protocol
FastEthernet0/0        192.168.12.1    YES manual      up           up
```

```
R2#show ip int brief

Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0 192.168.12.2           YES manual  administratively down down
```

Проверяем интерфейсы и видим, что кто-то ввел команду отключения интерфейса.

```
R2(config)#interface fa0/0
R2(config-if)#no shutdown
```

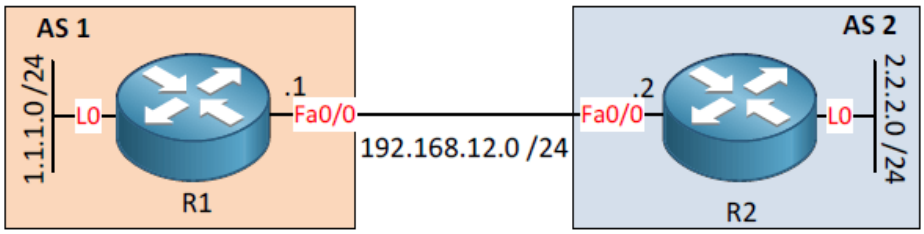
"Поднимаем" интерфейс

R1# **%BGP-5-ADJCHANGE: neighbor 192.168.12.2 Up** R2# **%BGP-5-ADJCHANGE: neighbor 192.168.12.1 Up**

Это прекрасно! Наше соседство BGP установлено. Это было легко!

Итог урока: убедитесь, что ваш интерфейс работает.

УРОК 2



Следующая неполадка похожа на предыдущую, но немного отличается. Мы используем те же маршрутизаторы и номера AS, но на этот раз необходимо установить соседство BGP между интерфейсами обратной связи.

Посмотрим, как выглядит конфигурация BGP:

R1**#show run | section bgp**
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 2.2.2.2 remote-as 2
 no auto-summary

R2**#show run | section bgp**
router bgp 2
 no synchronization
 bgp log-neighbor-changes
 neighbor 1.1.1.1 remote-as 1
 no auto-summary

Вот конфигурация BGP. Как вы видите, мы используем loopback интерфейсы для установления соседства BGP-соседей.

R1**#show ip bgp summary**

BGP router identifier 192.168.12.1, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
2.2.2.2	**4**	**2**	**0**	**0**	**0**	**0**	**0**	**never**	**Idle**

R2**#show ip bgp summary**

BGP router identifier 192.168.12.2, local AS number 2
BGP table version is 1, main routing table version 1

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
1.1.1.1	**4**	**1**	**0**	**0**	**0**	**0**	**0**	**never**	**Idle**

Оба маршрутизатора показывают, что их сосед BGP бездействует. Есть ряд вещей, которые мы **должны проверить** здесь:

- Доступен ли IP-адрес соседа BGP? Мы не используем прямые линии связи, поэтому у нас могут возникнуть проблемы с маршрутизацией.
- TTL IP-пакетов, которые мы используем для внешнего BGP, равен 1. Это работает для сетей с прямым подключением, но, если они не подключены напрямую, нам нужно изменить эту настройку.
- По умолчанию BGP будет получать обновления с IP-адреса, ближайшего к соседу BGP. В нашем примере это интерфейс FastEthernet. Это то, что мы должны изменить.

R1#**show ip route**
C 192.168.12.0/24 is directly connected, FastEthernet0/0

R2#**show ip route**
C 192.168.12.0/24 is directly connected, FastEthernet0/0

Начнем с маршрутизации. Оба маршрутизатора знают только о своих напрямую подключенных сетях. Чтобы достичь loopback интерфейсов друг друга, мы будем использовать статическую маршрутизацию.

```
R1(config)#ip route 2.2.2.2 255.255.255.255 192.168.12.2
R2(config)#ip route 1.1.1.1 255.255.255.255 192.168.12.1
```

Два статических маршрута должны выполнить эту работу.

R1#**ping 2.2.2.2 source loopback 0**

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 1.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms

Отправка ping на IP-адрес 2.2.2.2 и получение его из нашего собственного **loopback интерфейса** доказывает, что оба маршрутизатора знают, как связаться с loopback интерфейсом друг друга.

```
R1(config-router)#neighbor 2.2.2.2 ebgp-multihop 2
R2(config-router)#neighbor 1.1.1.1 ebgp-multihop 2
```

Команда ebgp-multihop изменяет TTL на 2.

R2#**debug ip bgp**
BGP debugging is on for address family: IPv4 Unicast

R2#
BGPNSF state: 1.1.1.1 went from nsf_not_active to nsf_not_active
BGP: 1.1.1.1 went from Active to Idle
BGP: 1.1.1.1 went from Idle to Active
BGP: 1.1.1.1 open active delayed 31810ms (35000ms max, 28% jitter)
BGP: 1.1.1.1 open active, local address 192.168.12.2
BGP: 1.1.1.1 open failed: Connection refused by remote host, open active
delayed 34480ms (35000ms max, 28% jitter)

Мы можем включить отладку, чтобы увидеть прогресс. Ясно видно, что R2 использует IP-адрес 192.168.12.2, а R1 отказывается от соединения.

```
R1(config-router)#neighbor 2.2.2.2 update-source loopback 0
R2(config-router)#neighbor 1.1.1.1 update-source loopback 0
```

Используйте команду **update-source**, чтобы изменить IP-адрес источника для обновлений BGP.

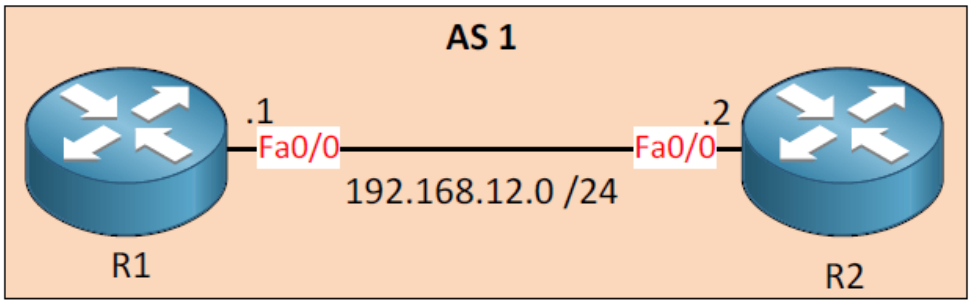
R1#
%BGP-5-ADJCHANGE: neighbor 2.2.2.2 Up

R2#
%BGP-5-ADJCHANGE: neighbor 1.1.1.1 Up

Соседство BGP работает!

Итог урока: маршрутизаторам BGP не требуется устанавливать соседство с использованием напрямую подключенных интерфейсов. Убедитесь, что маршрутизаторы BGP могут связаться друг с другом, что пакеты BGP получены из правильного интерфейса, и в случае EBGP не забудьте использовать команду **multihop**.

УРОК 3



Продолжим рассмотрение некоторых проблем IBGP. Два маршрутизатора в одной AS и вот конфигурация:

```
R1#show run | section bgp
router bgp 1
  no synchronization
  bgp log-neighbor-changes
  neighbor 192.168.12.2 remote-as 1
  no auto-summary
```

```
R2#show run | section bgp
router bgp 1
  no synchronization
  bgp log-neighbor-changes
  neighbor 192.168.12.1 remote-as 1
  no auto-summary
```

Легко и просто. Маршрутизаторы используют напрямую подключенные IP-адреса для соседства BGP.

```
R1#show ip bgp summary
BGP router identifier 192.168.12.1, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.12.2    4     1      46      46        0    0    0 00:05:24        Active
```

```
R2#show ip bgp summary
BGP router identifier 192.168.12.2, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.12.1    4     1      46      46        0    0    0 00:05:30        Active
```

Жаль ... мы не становимся соседями. Что может быть не так? Мы используем напрямую подключенные интерфейсы, поэтому не так много проблем, если не считать **проблемы L2 / L2**.

```
R1#ping 192.168.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

Отправка пинга с одного маршрутизатора на другой доказывает, что L2 и L3 работают нормально. Как насчет L3? У нас могут быть проблемы с транспортным уровнем.

```
R1#telnet 192.168.12.2 179
Trying 192.168.12.2, 179 ...
% Destination unreachable; gateway or host down
```

```
R2#telnet 192.168.12.1 179
Trying 192.168.12.1, 179 ...
```

Я не могу подключиться к TCP-порту 179 с обоих маршрутизаторов. Это звоночек в сторону того, что что-то блокирует BGP?

```
R1#show ip interface fastEthernet 0/0 | include access
      Outgoing access list is not set
      Inbound access list is not set
```

```
R2#show ip interface fastEthernet 0/0 | include access
      Outgoing access list is not set
      Inbound access list is 100
```

Вот оно! Это Служба безопасности....

```
R2#show ip access-lists
Extended IP access list 100
    10 deny tcp any eq bgp any (293 matches)
    15 deny tcp any any eq bgp (153 matches)
    20 permit ip any any (109 matches)
```

Кто-то решил, что было бы неплохо "**обезопасить**" BGP и заблокировать его списком доступа.

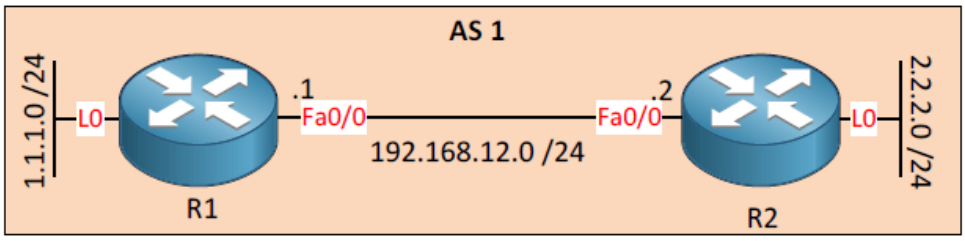
```
R2(config)#interface fastEthernet 0/0
R2(config-if)#no ip access-group 100 in
```

Удалим список доступа.

R1# %BGP-5-ADJCHANGE: neighbor 192.168.12.2 Up R2# %BGP-5-ADJCHANGE: neighbor 192.168.12.1 Up

Итог урока: не блокируйте TCP-порт BGP 179.

УРОК 4



Следующая проблема IBGP. Это похоже на ситуацию с EBGP ранее...мы будем использовать loopback-интерфейсы для установления соседства BGP, вот конфигурации:

```
R1#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 2.2.2.2 remote-as 1
 no auto-summary
```

```
R1#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 2.2.2.2 remote-as 1
 no auto-summary
```

Ничего особенного, IBGP и мы используем loopback интерфейсы.

```
R1#show ip bgp summary | begin Neighbor
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
2.2.2.2	4	1	0	0	0	0	0	never	Active

```
R2#show ip bgp summary | begin Neighbor
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
1.1.1.1	4	1	0	0	0	0	0	never	Active

Не повезло здесь ... нет соседей. Давайте сначала проверим, могут ли маршрутизаторы получить доступ к loopback интерфейсам друг друга:

```
R1#show ip route
C       192.168.12.0/24 is directly connected, FastEthernet0/0
        1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
```

```
R2#show ip route
C       192.168.12.0/24 is directly connected, FastEthernet0/0
        2.0.0.0/24 is subnetted, 1 subnets
C       2.2.2.0 is directly connected, Loopback0
```

Быстрый взгляд на таблицу маршрутизации показывает нам, что это не так. Мы могли бы исправить это с помощью статического маршрута или IGP. Обычно мы используем IGP для IBGP для объявления loopback интерфейсов. Сейчас будем **использовать OSPF**:

```
R1(config)#router ospf 1
R1(config-router)#network 1.1.1.0 0.0.0.255 area 0
R1(config-router)#network 192.168.12.0 0.0.0.255 area 0
R2(config)#router ospf 1
R2(config-router)#network 192.168.12.0 0.0.0.255 area 0
R2(config-router)#network 2.2.2.0 0.0.0.255 area 0
```

Набор правильных команд OSPF должно сделать свою работу!

R1#**ping 2.2.2.2 source loopback 0**

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 1.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms

Отправка **эхо-запроса**, чтобы проверить, знают ли маршрутизаторы и как связаться с сетями друг друга, успешен.

R1#**show ip bgp summary | begin Neighbor**

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
2.2.2.2	**4**	**1**	**0**	**0**	**0**	**0**	**0**	**never**	**Active**

R2#**show ip bgp summary | begin Neighbor**

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
1.1.1.1	**4**	**1**	**0**	**0**	**0**	**0**	**0**	**never**	**Active**

Тем не менее, соседство BGP по-прежнему отсутствует

R1#**debug ip bgp**
BGP debugging is on for address family: IPv4 Unicast
BGP: 2.2.2.2 open active, **local address 192.168.12.1**
BGP: 2.2.2.2 open failed: **Connection refused by remote host, open active**
delayed 32957ms (35000ms max, 28% jitter)

R2#**debug ip bgp**
BGP debugging is on for address family: IPv4 Unicast
BGP: 1.1.1.1 open active, **local address 192.168.12.2**
BGP: 1.1.1.1 open failed: **Connection refused by remote host, open active**
delayed 32957ms (35000ms max, 28% jitter)

Отладка показывает, что в соединении отказано, а также показывает локальный IP-адрес, который используется для BGP. Кажется, кто-то забыл добавить команду **update-source**, так что давайте исправим это!

```
R1(config)#router bgp 1
R1(config-router)#neighbor 2.2.2.2 update-source loopback 0
R2(config)#router bgp 1
R2(config-router)#neighbor 1.1.1.1 update-source loopback 0
```

Точно так же, как EBGP, мы должны установить правильный источник для наших пакетов BGP.

R1# **BGP-5-ADJCHANGE: neighbor 2.2.2.2 Up** R2# **BGP-5-ADJCHANGE: neighbor 1.1.1.1 Up**

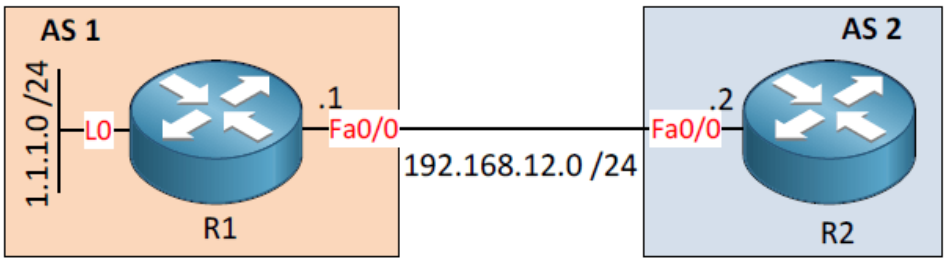
Задача решена! Единственное отличие от EBGP в том, что нам не нужно менять TTL с помощью команды **ebgp-multihop**.

Итог урока: распространенная практика настройки IBGP между loopback интерфейсами. Убедитесь, что эти loopback доступны и обновления BGP получены из loopback интерфейса.

8. Устранение неисправностей в BGP – часть 2

Мы продолжим рассмотрение вопроса об устранении неполадок в объявлениях о маршрутах BGP. Все маршрутизаторы будут иметь рабочие соседние узлы BGP.

УРОК 1



Новый сценарий. R1 и R2 находятся в разных автономных системах. Мы пытаемся объявить сеть 1.1.1.0 / 24 от R1 до R2, но она не отображается на R2. Вот конфигурации:

```
R1#show run | section bgp
no synchronization
bgp log-neighbor-changes
network 1.1.1.0
neighbor 192.168.12.2 remote-as 2
no auto-summary
```

```
R2#show run | section bgp
router bgp 2
 no synchronization
 bgp log-neighbor-changes
 neighbor 192.168.12.1 remote-as 1
 no auto-summary
```

На первый взгляд, здесь все в порядке.

```
R2#show ip bgp summary
BGP router identifier 192.168.12.2, local AS number 2
BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.12.1    4     1       4       4        1    0    0 00:01:26        0
```

Однако R2 не узнал никаких префиксов от R1

```
R1#show ip protocols | include filter
    Outgoing update filter list for all interfaces is not set
    Incoming update filter list for all interfaces is not set
```

```
R2#show ip protocols | include filter
    Outgoing update filter list for all interfaces is not set
    Incoming update filter list for all interfaces is not set
```

Может быть, используется `distribute-list`. Но нет, это не тот случай. Это означает, что нам придется проверять наши все команды network.

```
R1#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 network 1.1.1.0
 neighbor 192.168.12.2 remote-as 2
 no auto-summary
```

Проблема заключается в команде **network**. Она настраивается по-разному для BGP и нашего IGP. Если мы применяем команду network для BGP, она должна быть полной. В этом случае забыли добавить маску подсети

```
R1(config)#router bgp 1
R1(config-router)#network 1.1.1.0 mask 255.255.255.0
```

Мы должны убедиться, что ввели правильную маску подсети.

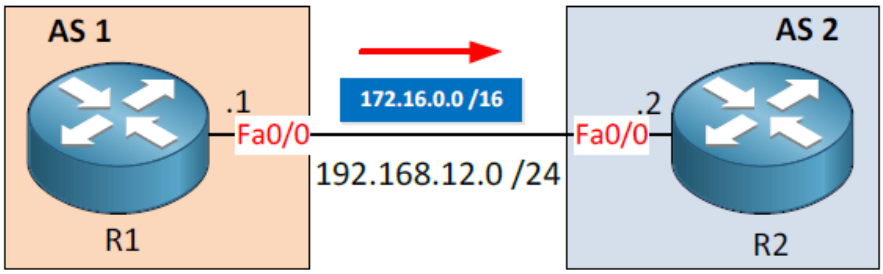
```
R2#show ip bgp summary | begin Neighbor
Neighbor        V    AS  MsgRcvd  MsgSent   TblVer  InQ  OutQ  Up/Down  State/PfxRcd
192.168.12.1    4     1        9        8        2    0     0  00:05:15        1
```

```
R2#show ip route bgp
      1.0.0.0/24 is subnetted, 1 subnets
B        1.1.1.0 [20/0] via 192.168.12.1, 00:01:08
```

Итак, видно, что мы узнали префикс, и R2 устанавливает его в таблицу маршрутизации ... проблема решена!

Итог урока: введите правильную маску подсети ... BGP требователен!

УРОК 2



Давайте перейдем к следующей проблеме. Системный администратор из AS1 хочет объявить summary в AS 2. Системный администратор из AS 2 жалуется, однако, что он ничего не получает..., давайте, выясним, что происходит не так!

```
R1#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 aggregate-address 172.16.0.0 255.255.0.0
 neighbor 192.168.12.2 remote-as 2
 no auto-summary
```

```
R2#show run | section router bgp
router bgp 2
 no synchronization
 bgp log-neighbor-changes
 neighbor 192.168.12.1 remote-as 1
 no auto-summary
```

Вот конфигурация. Вы можете увидеть команду **aggregate-address** на R1 для сети 172.16.0.0 / 16.

```
R2#show ip bgp summary | begin Neighbor
Neighbor        V    AS  MsgRcvd  MsgSent   TblVer  InQ  OutQ  Up/Down  State/PfxRcd
192.168.12.1    4     1       21       19        3    0     0  00:16:21        0
```

Жаль ... префиксы не были получены R2. Здесь мы можем проверить две вещи:

- Проверьте, не блокирует ли distribute-list префиксы, как это мы сделали в предыдущем занятии.
- Посмотрите, что R1 имеет в своей таблице маршрутизации (Правило: "не могу объявлять то, чего у меня нет!").

Давайте начнем с таблицы маршрутизации R1. Из предыдущих уроков вы знаете, как выглядит **distribute-list**.

```
R1#show ip route
C       192.168.12.0/24 is directly connected, FastEthernet0/0
        1.0.0.0/24 is subnetted, 1 subnets
C          1.1.1.0 is directly connected, Loopback0
```

Здесь нет ничего, что выглядело бы даже близко к 172.16.0.0 /16. Если мы хотим объявить **summary**, мы должны сначала поместить что-то в таблицу маршрутизации R1. Рассмотрим различные варианты:

```
R1(config)#interface loopback 0
R1(config-if)#ip address 172.16.0.1 255.255.255.0
R1(config-if)#exit
R1(config)#router bgp 1
R1(config-router)#network 172.16.0.0 mask 255.255.255.0
```

Это вариант 1. Создам интерфейс loopback0 и настроим IP-адрес, который попадает в диапазон команды **aggregate-address**.

```
R2#show ip route bgp
      172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
B        172.16.0.0/24 [20/0] via 192.168.12.1, 00:01:25
B        172.16.0.0/16 [20/0] via 192.168.12.1, 00:01:25
```

Теперь мы видим summary в таблице маршрутизации R2. По умолчанию он все равно будет объявлять другие префиксы. Если вы не хотите этого, вам нужно использовать команду aggregate-address summaryonly!

Второй вариант объявления summary:

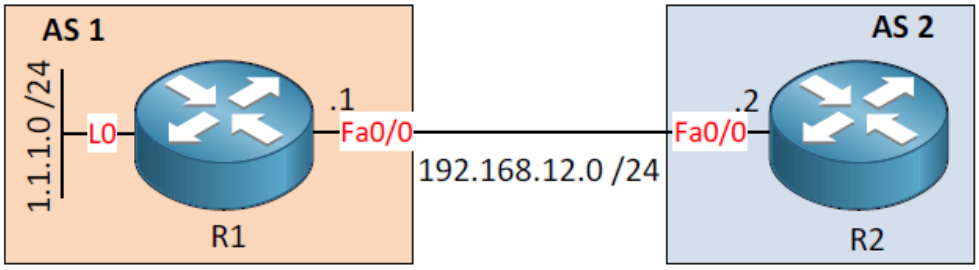
```
R1(config)#ip route 172.16.0.0 255.255.0.0 null 0
R1(config)#router bgp 1
R1(config-router)#network 172.16.0.0 mask 255.255.0.0
```

Сначала мы поместим сеть 172.16.0.0 / 16 в таблицу маршрутизации, создав статический маршрут и указав его на интерфейсе null0. Во-вторых, будем использовать команду network для BGP для объявления этой сети.

```
R2#show ip route bgp
B        172.16.0.0/16 [20/0] via 192.168.12.1, 00:00:45
```

Итог урока: Вы не можете объявлять то, чего у вас нет. Создайте статический маршрут и укажите его на интерфейсе null0, чтобы создать loopback интерфейс с префиксом, который попадает в диапазон суммарных адресов.

УРОК 3



Следующая проблема. Вы работаете системным администратором в AS 1, и однажды получаете телефонный звонок от системного администратора AS 2, который интересуется у вас, почему вы публикуете сводку для 1.0.0.0 / 8. Вы понятия не имеете, о чем, он говорит, поэтому решаете проверить свой роутер.

```
R2#show ip route bgp
B        1.0.0.0/8 [20/0] via 192.168.12.1, 00:02:15
```

Это то, что видит системный администратор на R2.

```
R1#show ip bgp 1.0.0.0
BGP routing table entry for 1.0.0.0/8, version 3
Paths: (1 available, best #1, table Default-IP-Routing-Table)
        Advertised to update-groups:
             1
        Local
          0.0.0.0 from 0.0.0.0 (1.1.1.1)
            Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
```

Мы видим, что у нас есть сеть 1.0.0.0 / 8 в таблице BGP на R1. Давайте проверим его таблицу маршрутизации.

```
R1#show ip route 1.0.0.0
Routing entry for 1.0.0.0/24, 1 known subnets
    Attached (1 connections)
    Redistributing via bgp 1
    Advertised by bgp 1

C        1.1.1.0 is directly connected, Loopback0
```

Сеть 1.1.1.0 / 24 настроена на loopback интерфейс, но она находится в таблице BGP как 1.0.0.0 / 8. Это может означать только одну вещь ... суммирование.

```
R1#show ip protocols
Routing Protocol is "bgp 1"
    Outgoing update filter list for all interfaces is not set
    Incoming update filter list for all interfaces is not set
    IGP synchronization is disabled
    Automatic route summarization is enabled
```

Беглый взгляд на выводы команды **show ip protocols** показывает, что автоматическое суммирование включено. Отключим это:

```
R1(config)#router bgp 1
R1(config-router)#no auto-summary
```

Мы отключим его на R1.

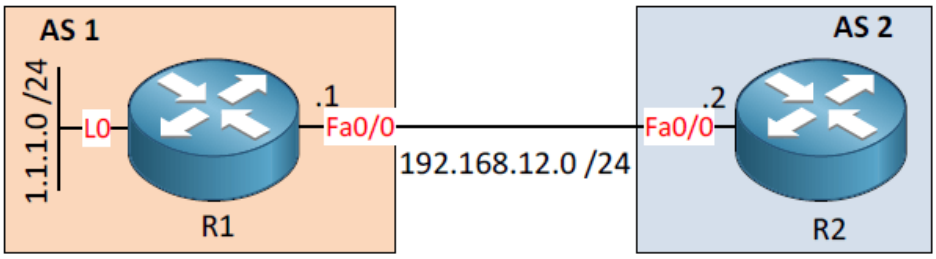
```
R2#show ip route bgp
    1.0.0.0/24 is subnetted, 1 subnets
B       1.1.1.0 [20/0] via 192.168.12.1, 00:00:20
```

Теперь мы видим 1.1.1.0 / 24 на R2 ... проблема решена!

Итог урока: если вы видите **classful** сети в своей таблице BGP, возможно, вы включили автоматическое суммирование.

Некоторые из проблем, которые были рассмотрены, можно легко решить, просто посмотрев и/или сравнив результаты команды "**show run**". И это правда, но имейте в виду, что у вас не всегда есть доступ ко ВСЕМ маршрутизаторам в сети, поэтому, возможно, нет способа сравнить конфигурации. Между устройствами, на которых вы пытаетесь устранить неисправности или которые вызывают проблемы, может быть коммутатор или другой маршрутизатор. Использование соответствующих команд show и debug покажет вам, что именно делает ваш маршрутизатор и что он сообщает другим маршрутизаторам.

УРОК 4



Та же топология, другая проблема. Персонал из AS 2 жалуются, что они ничего не получают от AS 1. Для усложнения проблемы, конфигурация не будет показана.

```
R2#show ip bgp summary | begin Neighbor
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.12.1    4     1      51      48        1    0    0 00:08:51        0
```

Для начала, мы видим, что R2 не получает никаких префиксов.

```
R1#show ip protocols | include filter
    Outgoing update filter list for all interfaces is not set
    Incoming update filter list for all interfaces is not set
```

Так же можем убедиться, что R1 не имеет каких-либо distribute-lists.

```
R1#show ip bgp 1.1.1.0
BGP routing table entry for 1.1.1.0/24, version 4
Paths: (1 available, best #1, table Default-IP-Routing-Table)
    Not advertised to any peer
    Local
      0.0.0.0 from 0.0.0.0 (1.1.1.1)
        Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
```

Мы видим, что R1 действительно имеет сеть 1.1.1.0 /24 в своей таблице маршрутизации, так почему же он не объявляет ее в R2?

Давайте посмотрим, может на R1 есть какие-то особенные настройки для своего соседа R2:

```
R1#show ip bgp neighbors 192.168.12.2
BGP neighbor is 192.168.12.2, remote AS 2, external link
  BGP version 4, remote router ID 192.168.12.2
  BGP state = Established, up for 00:03:34
  Last read 00:00:33, last write 00:00:33, hold time is 180, keepalive interval is 60 seconds
 Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
 Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                 11         11
    Notifications:          0          0
    Updates:                7          0
    Keepalives:            85         86
    Route Refresh:          0          0
    Total:                103         97
Default minimum time between advertisement runs is 30 seconds

For address family: IPv4 Unicast
  BGP table version 3, neighbor version 3/0
Output queue size : 0
  Index 1, Offset 0, Mask 0x2
  1 update-group member
  Outbound path policy configured
  Route map for outgoing advertisements is NEIGHBORS
                         Sent       Rcvd
Prefix activity:         ----       ----
  Prefixes Current:         0          0
  Prefixes Total:           0          0
  Implicit Withdraw:        0          0
  Explicit Withdraw:        0          0
  Used as bestpath:       n/a          0
  Used as multipath:      n/a          0
```

Будем использовать команду **show ip bgp neighbors**, чтобы увидеть подробную информацию о R2. Мы видим, что route-map была применена к R2 и называется "**NEIGHBORS**". Имейте в виду, что помимо **distribute-lists** мы можем использовать также route-map для фильтрации BGP.

```
R1# show route-map
route-map NEIGHBORS, permit, sequence 10
   Match clauses:
     ip address prefix-lists: PREFIXES
   Set clauses:
   Policy routing matches: 0 packets, 0 bytes
```

Существует только оператор соответствия для prefix-list "**PREFIXES**".

```
R1#show ip prefix-list
ip prefix-list PREFIXES: 1 entries
   seq 5 deny 1.1.1.0/24
```

Вот наш нарушитель спокойствия ... он запрещает сеть 1.1.1.0 / 24!

```
R1(config)#router bgp 1
R1(config-router)#no neighbor 192.168.12.2 route-map NEIGHBORS out
```

Удалим **route-map**

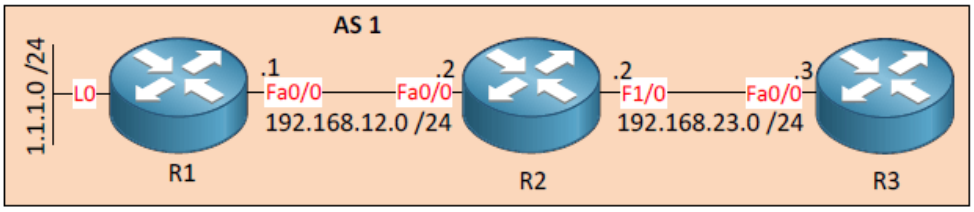
```
R2#show ip route bgp
1.0.0.0/24 is subnetted, 1 subnets
B 1.1.1.0 [20/0] via 192.168.12.1, 00:00:03
```

И наконец R2 узнал об этом префиксе ... проблема решена!

Итог урока: убедитесь, что нет route-map, блокирующих объявление префиксов.

BGP иногда может быть очень медленным, особенно когда вы ждете результатов, когда вы работаете на тестовом или лабораторном оборудовании. "**Clear ip bgp ***" - это хороший способ ускорить его ... просто не делайте этого на маршрутизаторах в производственной сети)

УРОК 5



Наконец, третий участник выходит на арену, чтобы продемонстрировать новую проблему. R1-это объявляемая сеть 1.1.1.0 / 24, но R3 не изучает эту сеть. Здесь представлены конфигураций:

```
R1#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 network 1.1.1.0 mask 255.255.255.0
 neighbor 192.168.12.2 remote-as 1
 no auto-summary
```

```
R2#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 192.168.12.1 remote-as 1
 neighbor 192.168.23.3 remote-as 1
 no auto-summary
```

```
R3#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 192.168.23.2 remote-as 1
 no auto-summary
```

Соседство настроено, R1 - объявляемая сеть 1.1.1.0 / 24.

```
R2#show ip route bgp
      1.0.0.0/24 is subnetted, 1 subnets
B           1.1.1.0 [200/0] via 192.168.12.1, 00:00:23
```

```
R3#show ip route bgp
```

Мы можем видеть сеть 1.1.1.0 / 24 в таблице маршрутизации R2, но она не отображается на R3.

Технически проблем нет. Если вы внимательно посмотрите на конфигурацию BGP всех трех маршрутизаторов, то увидите, что существует только соседство BGP между R1 и R2 и между R2 и R3. Из-за **split horizon** IBGP R2 не пересылает сеть 1.1.1.0 / 24 в направлении R3. Чтобы это исправить, нам нужно настроить R1 и R3, чтобы они стали соседями.

```
R1(config)#ip route 192.168.23.3 255.255.255.255 192.168.12.2
R3(config)#ip route 192.168.12.1 255.255.255.255 192.168.23.2
```

Если мы собираемся настроить соседство BGP между R1 и R3, нам нужно убедиться, что они могут достигать друг друга. Мы можем использовать статическую маршрутизацию или IGP ... чтобы упростить задачу, на этот раз мы будем использовать статический маршрут.

```
R1(config)#router bgp 1
R1(config-router)#neighbor 192.168.23.3 remote-as 1
R3(config)#router bgp 1
R3(config-router)#neighbor 192.168.12.1 remote-as 1
```

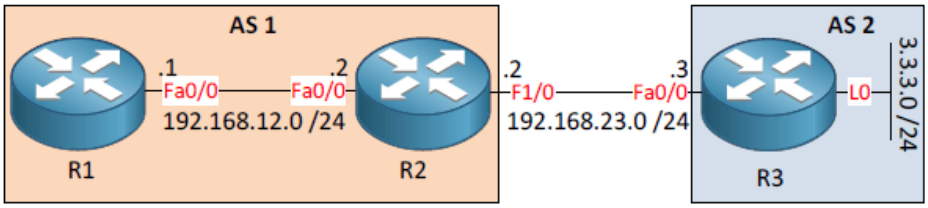
Примените правильные настройки команды neighbor BGP.

```
R3#show ip route bgp
      1.0.0.0/24 is subnetted, 1 subnets
B           1.1.1.0 [200/0] via 192.168.12.1, 00:00:08
```

И R3 имеет доступ к сети 1.1.1.0 / 24!

Итог урока: соседство по IBGP должно быть полным циклом! Другим решением было бы использование **route-reflector** или confederation.

УРОК 6



Очередная проблема. R3 является объявляемой сетью 3.3.3.0 / 24 через EBGP, а R2 устанавливает ее в таблицу маршрутизации. R1, однако, не имеет этой сети в своей таблице маршрутизации. Вот конфигурации:

```
R1#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 192.168.12.2 remote-as 1
 no auto-summary
```

```
R2#show run | section router bgp
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 192.168.12.1 remote-as 1
 neighbor 192.168.23.3 remote-as 2
 no auto-summary
```

```
R3#show run | section router bgp
router bgp 2
 no synchronization
 bgp log-neighbor-changes
 network 3.3.3.0 mask 255.255.255.0
 neighbor 192.168.23.2 remote-as 1
 no auto-summary
```

Вот конфигурации. Для простоты мы используем IP-адреса физического интерфейса для настройки соседей BGP.

```
R2#show ip route bgp
      3.0.0.0/24 is subnetted, 1 subnets
B        3.3.3.0 [20/0] via 192.168.23.3, 00:09:37
```

Мы можем проверить, что сеть 3.3.3.0 / 24 находится в таблице маршрутизации R2.

```
R1#show ip route bgp
```

Однако в таблице маршрутизации R1 ничего нет. Первое, что мы должны проверить - это таблицу BGP.

```
R1#show ip bgp
BGP table version is 1, local router ID is 192.168.12.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal, r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
* i3.3.3.0/24      192.168.23.3             0    100      0 2 i
```

Мы видим, что он находится в таблице BGP, и * указывает, что это допустимый маршрут. Однако мы не видим символа >, который указывает лучший путь. По какой-то причине BGP не может установить эту запись в таблице маршрутизации. Внимательно посмотрите на следующий IP-адрес прыжка (192.168.23.3). Доступен ли этот IP-адрес?

```
R1#show ip route 192.168.23.3
% Network not in table
```

R1 понятия не имеет, как достичь 192.168.23.3, поэтому наш следующий прыжок недостижим. Есть два способа, как мы можем справиться с этой проблемой:

- Используйте статический маршрут или IGP, чтобы сделать этот next hop IP-адрес доступным.
- Измените next hop IP-адрес.

Мы изменим IP-адрес следующего прыжка, так как мы достаточно изучили применение статических маршрутов и IGPs.

```
R2(config)#router bgp 1
R2(config-router)#neighbor 192.168.12.1 next-hop-self
```

Эта команда изменит IP-адрес следующего перехода на IP-адрес R2.

```
R1#show ip bgp
BGP table version is 2, local router ID is 192.168.12.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? – incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i3.3.3.0/24       192.168.12.2             0    100      0 2 i
```

Теперь мы видим символ >, который указывает, что этот путь был выбран как лучший. IP-адрес следующего перехода теперь 192.168.12.2.

```
R1#show ip route bgp
      3.0.0.0/24 is subnetted, 1 subnets
B        3.3.3.0 [200/0] via 192.168.12.2, 00:10:52
```

Ура! Теперь он есть в таблице маршрутизации. Мы уже закончили? Если наша цель состояла в том, чтобы она отобразилась в таблице маршрутизации, то мы закончили...однако есть еще одна проблема.

```
R1#ping 3.3.3.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
.....
  Success rate is 0 percent (0/5)
```

Наш пинг не удался. R1 и R2 оба имеют сеть 3.3.3.0 / 24 в своей таблице маршрутизации, поэтому мы знаем, что они знают, куда пересылать IP-пакеты.

Давайте взглянем на R3:

```
R3#show ip route
      3.0.0.0/24 is subnetted, 1 subnets
C        3.3.3.0 is directly connected, Loopback0
C     192.168.23.0/24 is directly connected, FastEthernet0/0
```

R3 получит IP-пакет с пунктом назначения 3.3.3.3 и источником 192.168.12.1. Из таблицы маршрутизации видно, что она не знает, куда отправлять IP-пакеты, предназначенные для 192.168.12.1. Исправим это:

```
R2(config)#router bgp 1
R2(config-router)#network 192.168.12.0 mask 255.255.255.0
```

Мы будем объявлять сеть 192.168.12.0 / 24 на R2.

```
R3#show ip route bgp

B     192.168.12.0/24 [20/0] via 192.168.23.2, 00:00:33
```

Теперь R3 знает, куда отправлять трафик для 192.168.12.0 / 24.

```
R1#ping 3.3.3.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/11/28 ms
```

Проблема устранена!

Итог урока: убедитесь, что IP-адрес следующего перехода доступен, чтобы маршруты могли быть установлены в таблице маршрутизации, и чтобы все необходимые сети были достижимы.