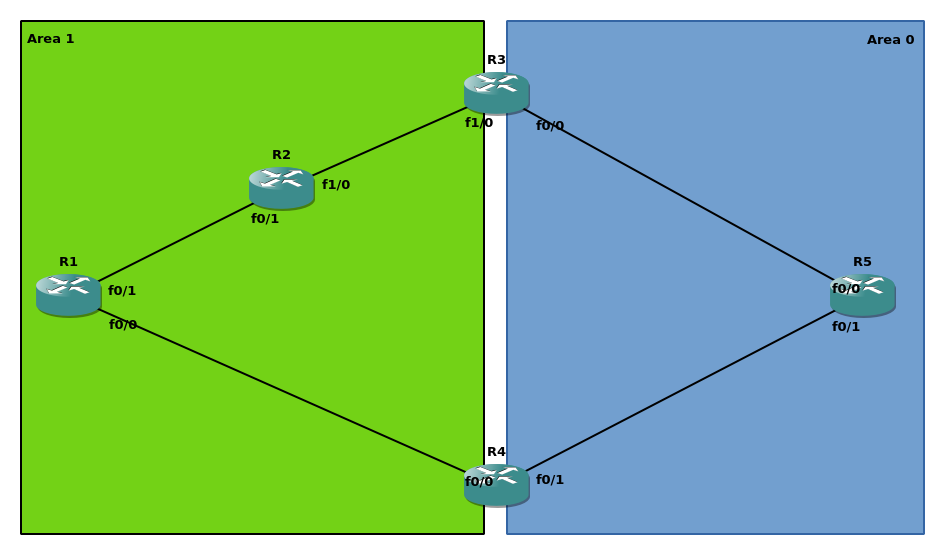
**Краткое пособие по созданию петель маршрутизации в OSPF в домашних условиях**

OSFP, будучи link-state протоколом, исключает петли в топологии за счёт построения дерева кратчайшего пути в рамках одной зоны с помощью алгоритма Дейкстры. Однако поведение OSPF между зонами напоминает скорее поведение distance-vector протоколов, которые обмениваются всего лишь префиксами и соответствующими метриками без каких-либо данных о фактической топологии; по этой причине некоторые авторы могут называть OSPF гибридным протоколом маршрутизации. Механизм защиты от петель маршрутизации между зонами, однако, довольно прост: все зоны должны обмениваться маршрутной информацией через backbone зону, зону 0, прямой обмен маршрутами между зонами невозможен.

Впрочем, тёмный гений изобрёл инструмент разрушения стройной идеи OSPF – речь о функции OSPF Virtual Link (VL). Даже маленькие инженеры знают, что использовать VL – плохая затея; все согласны с тем, что VL оправдан только в крайних случаях для временного и быстрого исправления критичной ситуации. Однако обнаружить подробное объяснение, чем же именно VL так плох помимо дополнительного уровня сложности, оказалось не так-то просто. Сложность инженерам не помеха, поэтому давайте поищем более весомые аргументы против VL.

Нет ничего лучше чашки кофе с утра, рабочей лабы и широкополосного доступа к google.com. Для эмуляции топологии в GNS3 были использованы образы Cisco 7200:



На каждом маршрутизаторе в соответствующей зоне настроен виртуальный интерфейс (loopback0) для назначения OSPF RID и других инфраструктурных задач; loopback на ABR находятся в зоне 0. Схема адресации: 192.168.xy.x|y/24 для соединения Rx и Ry (например, 192.168.12.1 на интерфейсе f0/1 R1). Помимо штатной настройки OSPF, между R1 и R3 создан VL.

Если у читателя много свободного времени, можно убедиться в доступности всех префиксов из любой точки сети; я же сконцентрируюсь на связности R1 и R5:

*R1#ping 5.5.5.5 so lo 0*

*Type escape sequence to abort.*

*Sending 5, 100-byte ICMP Echos to 5.5.5.5, 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 = 20/30/40 ms*

*R1#traceroute 5.5.5.5 so lo 0 numeric*

*Type escape sequence to abort.*

*Tracing the route to 5.5.5.5*

*VRF info: (vrf in name/id, vrf out name/id)*

*1 192.168.14.4 16 msec 24 msec 20 msec*

*2 192.168.45.5 48 msec 16 msec 24 msec*

Трафик следует по кратчайшему маршруту, как и задумывалось. Перейдем к гвоздю сегодняшней программы:

*R1(config)#router os 1*

*R1(config-router)#no capability transit*

Transit area capability – это способность OSPFv2 выбрать более оптимальный маршрут для трафика, следующего по VL. Идея довольно проста: необходимо взять префикс, доступный через VL, и сравнить его с имеющимися LSA3; если найдено полное совпадение и маршрут через LSA3 оптимальнее, то следует использовать оптимальный маршрут. OSPFv1 не обладал такой функцией, и весь трафик следовал по тому же пути, что и VL. Если читатель хочет глубже погрузиться в данную тему, могу порекомендовать [статью Петра Лапухова]. <https://blog.ine.com/2009/09/14/understanding-ospf-transit-capability>). А мы продолжаем:

*R1#traceroute 5.5.5.5 so lo 0 n*

*Type escape sequence to abort.*

*Tracing the route to 5.5.5.5*

*VRF info: (vrf in name/id, vrf out name/id)*

*1 192.168.12.2 44 msec 16 msec 20 msec*

*2 192.168.23.3 20 msec 40 msec 40 msec*

*3 192.168.35.5 76 msec 44 msec 44 msec*

Изменения есть, но они некритичны: теперь R1 выбирает путь вдоль VL в полном соответствии с поведением OSPFv1 без transit capability. Где же обещанная петля? Добавим нашей топологии некоторой пикантности:

|  |  |
| --- | --- |
| **R2** | **R3** |
| R2(config)#int f1/0  R2(config-if)#ip os cost 100 | R3(config)#int f1/0  R3(config-if)#ip os cost 100 |

Ухудшили плохой маршрут, и что с того?

*R1#traceroute 5.5.5.5 so lo 0 n*

*Type escape sequence to abort.*

*Tracing the route to 5.5.5.5*

*VRF info: (vrf in name/id, vrf out name/id)*

*1 192.168.12.2 20 msec 16 msec 16 msec*

*2 192.168.12.1 24 msec 16 msec 16 msec*

*3 192.168.12.2 36 msec 32 msec 44 msec*

*4 192.168.12.1 28 msec 36 msec 40 msec*

*5 192.168.12.2 44 msec 48 msec 64 msec*

*6 192.168.12.1 60 msec 60 msec 60 msec*

*7 192.168.12.2 80 msec 80 msec 80 msec*

*8 192.168.12.1 84 msec 76 msec 76 msec*

*<output greedily omitted>*

Дамы и господа, это – петля. Теперь взглянем на проделанную работу:

* VL между R1-R3;
* Transit capability отключена;
* Метрика R2-R3 увеличена.

Последний пункт стоит отметить особо: в результате изменения метрики R1 является next-hop для 5.5.5.5/32 с точки зрения R2:

*R2#sho ip ro 5.5.5.5 255.255.255.255 longer-prefixes*

*5.0.0.0/32 is subnetted, 1 subnets*

*O IA 5.5.5.5 [110/4] via 192.168.12.1, 00:06:21, FastEthernet0/1*

Выбор маршрута с точки зрения R2 выглядит довольно естественно; выбор R1 же может показаться странным на первый взгляд:

*R1#sho ip ro 5.5.5.5 255.255.255.255 longer-prefixes*

*5.0.0.0/32 is subnetted, 1 subnets*

*O 5.5.5.5 [110/103] via 192.168.12.2, 00:08:07, FastEthernet0/1*

Однако такое поведение вполне соответствует правилам выбора маршрута в OSPFv1:

* 5.5.5.5/32 доступен через VL в зоне 0;
* трафик следует по тому же пути, что и VL.

R2 ничего не знает ни про VL, ни про transit area; он всего ли маршрутизирует пакеты согласно таблице маршрутизации, построенной на основе LSA3.

На данном этапе может показаться, что причина всех бед – transit capability, а не VL. Впрочем, разрешается этот вопрос довольно просто: подобная проблема была обнаружена в OSPFv1 и решена в OSPFv2 посредством transit area. Читатель может заметить, что описанное поведение родственно по природе микропетлям, возникающим в ходе перестроения дерева, и с ним трудно не согласиться. Впрочем, разница есть и существенная: микропетли представляют собой переходное состояние. тогда как петли из-за VL в OSPFv1 носили постоянный характер.

Немного английского оригинала из OSPFv2 RFC про отличия от OSPFv1:

***“When summarizing information into a virtual link's transit area, version 2 of the OSPF specification prohibits the collapsing of multiple backbone IP networks/subnets into a single summary link.”***

Проблема, завуалированная авторами RFC в этой фразе, очень похожа по характеру на рассматриваемую в статье. Если бы маршруты зоны 0 можно было бы суммаризовать на ABR, то полученный префикс в LSA3 не смог бы быть использован маршрутизатором с VL. Передача трафика была бы нарушена вследствие разного представления о топологии маршрутизаторами зоны:

1. Маршрутизатор с VL выбрал бы маршрут из зоны 0; из-за отсутствия точно совпадающего LSA3 был бы выбран путь вдоль VL;
2. Остальные же маршрутизаторы использовали бы суммарный маршрут из LSA3.

В нашем случае, если бы R3 суммаризовал маршрут до 5.5.5.0/24, то R2 выбрал бы более точный маршрут 5.5.5.0/25 через R4, что привело бы к петле маршрутизации. Решение? Нужно обеспечить всем маршрутизаторам зоны равный доступ к маршрутной информации из зоны 0, запретив изменение последней, т.е. суммаризацию. Стоит отметить, что это не относится к маршрутам из других зон, поскольку эти суммарные маршруты будут одинаковы как в зоне 0, так и других зонах; только ABR зоны-источника могут суммаризовать информацию о топологии из LSA1/2 в простой LSA3. Более того, запрещена только суммаризация префиксов из зоны 0, фильтрация же не ограничена. Последняя не изменяет сам префикс; в результате у маршрутизаторов зоны будет меньше пар LSA1-LSA3 для выбора более оптимального маршрута, что не может привести к описанной в статье проблеме.

Вывод: некоторые малоизвестные настройки по умолчанию лучше не трогать.

P.S. Существует не менее интересный способ выстрелить себе в ногу:

<https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/117824-config-ospf-00.html>