Настройка OSPF

Вступление

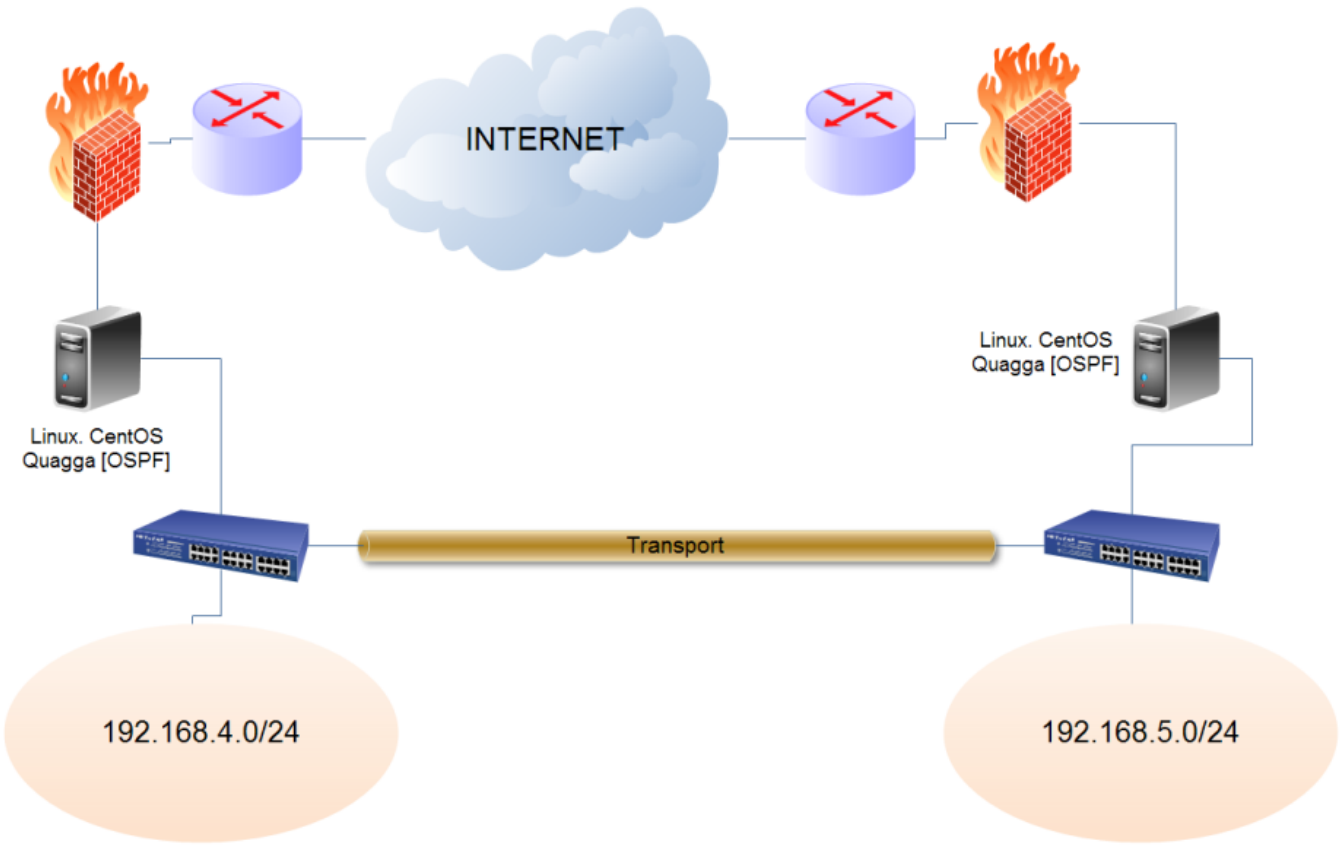
По мере роста сети и количества компьютеров, а также требований к доступности тех или иных ресурсов, системным администраторам приходится выбирать соответствующие технические и программные средства для реализации поставленных задач и требований. И наступает момент, когда использование самописанных скриптов для переключения каналов уже не подходит по множеству причин.

Итак в нашем распоряжении два офиса, которые соединены между собой с помощью транспорта, для решения данной задачи не имеет значение что выступает в роли этого транспорта - оптика/витая пара/коммутируемая линия. Также у каждого из офисов есть свой отдельный провайдер, через которого они и выходят в мир.

Наша задача - обеспечить надежную связь между офисами для совместного использования общих ресурсов, например, веб сервера (apache), базы данных (MySQL, PostgreSQL,...), файл сервер (win 2k3/samba) и т.д.

Нам необходимо сделать так, чтобы подсети двух офисов «видели» друг друга в любой момент времени, не зависимо от доступности транспорта между ними.

Для лучшего понимания задачи ниже привожу упрощенную логическую схему.



Как видно из рисунка у нас в любой момент времени доступны два канала связи между

офисами - транспорт и интернет. Для того, чтобы подсети двух офисов видели друг друга мы поднимаем vpn тунели между ними. Как именно вы будете делать - не имеет особого значения, я это реализовал с помощью openvpn. Вы можете использовать для этих целей например ipsec. Один vpn тунель у нас поднят поверх транспорта и этот канал является основным для обмена информацией между офисами. Второй vpn тунель поднят поверх интернет каналов каждого из офисов. И данный тунель будет использоваться нами только в случае недоступности основного vpn тунеля.

Для решения этой задачи можно было бы воспользоваться скриптом, который проверяет доступность удаленной точки vpn тунеля и в случае ее недоступности переключал бы роутинг с основного канала на резервный, а после восстановления делал бы обратное действие.

Но я предлагаю рассмотреть использование протокола динамической маршрутизации для решения данной задачи.

Краткий обзор OSPF

OSPF (англ. Open Shortest Path First) — протокол динамической маршрутизации, основанный на технологии отслеживания состояния канала (link-state technology) и использующий для нахождения кратчайшего пути Алгоритм Дейкстры (Dijkstra's algorithm).

Протокол OSPF был разработан IETF в 1988 году. Последняя версия протокола представлена в RFC 2328. Протокол OSPF представляет собой протокол внутреннего шлюза (Interior Gateway Protocol — IGP). Протокол OSPF распространяет информацию о доступных маршрутах между маршрутизаторами одной автономной системы. В то время как BGP относится к протоколу внешнего шлюза (сокр. от англ. Exterior Gateway Protocol).

OSPF предлагает решение следующих задач:

- Увеличение скорости сходимости (в сравнении с протоколом RIP2, т.к. нет необходимости выжидания многократных таймаутов по 30с);
- Поддержка сетевых масок переменной длины (VLSM);
- Достижимость сети (быстро обнаруживаются отказавшие маршрутизаторы, и топология сети изменяется соответствующим образом);
- Оптимальное использование пропускной способности;
- Метод выбора пути.

Терминология OSPF

- Интерфейс (interface) — соединение маршрутизатора и одной из подключенных к нему сетей. При обсуждении OSPF термины интерфейс и канал (link) часто употребляются как синонимы.
- Объявление о состоянии канала (link-state advertisement, LSA) — объявление описывает все каналы маршрутизатора, все интерфейсы и состояние каналов.
- Состояние канала (link state) — состояние канала между двумя маршрутизаторами; обновления происходят при помощи пакетов LSA.
- Метрика (metric) — условный показатель «стоимости» пересылки данных по каналу;
- Автономная система (autonomous system) — группа маршрутизаторов, обменивающаяся маршрутизирующей информацией с помощью одного протокола маршрутизации.

- Зона (area) — совокупность сетей и маршрутизаторов, имеющих один и тот же идентификатор зоны.
- Соседи (neighbours) — два маршрутизатора, имеющие интерфейсы в общей сети.
- Состояние соседства (adjacency) — взаимосвязь между определёнными соседними маршрутизаторами установленная с целью обмена информацией маршрутизации.
- Hello-протокол (hello protocol) — используется для поддержания соседских отношений.
- База данных соседей (neighbours database) — список всех соседей.
- База данных состояния каналов (link state database, LSDB) — список всех записей о состоянии каналов. Встречается также термин топологическая база данных (topological database), употребляется как синоним базы данных состояния каналов.
- Идентификатор маршрутизатора (router ID, RID) — уникальное 32-битовое число, которое уникально идентифицирует маршрутизатор в пределах одной автономной системы.

Описание работы OSPF

1. Маршрутизаторы обмениваются hello-пакетами через все интерфейсы, на которых активирован OSPF. Маршрутизаторы, разделяющие общий канал передачи данных, становятся соседями, когда они приходят к договоренности об определённых параметрах, указанных в их hello-пакетах.
2. На следующем этапе работы протокола маршрутизаторы будут пытаться перейти в состояние соседства с маршрутизаторами, находящимися с ним в пределах прямой связи (на расстоянии одного хопа). Переход в состояние соседства определяется типом маршрутизаторов, обменивающихся hello-пакетами, и типом сети, по которой передаются hello-пакеты. OSPF определяет несколько типов сетей и несколько типов маршрутизаторов. Пара маршрутизаторов, находящихся в состоянии соседства, синхронизирует между собой базу данных состояния каналов.
3. Каждый маршрутизатор посылает объявление о состоянии канала маршрутизаторам, с которыми он находится в состоянии соседства.
4. Каждый маршрутизатор, получивший объявление от соседа, записывает передаваемую в нём информацию в базу данных состояния каналов маршрутизатора и рассылает копию объявления всем другим своим соседям.
5. Рассылая объявления через зону, все маршрутизаторы строят идентичную базу данных состояния каналов маршрутизатора.
6. Когда база данных построена, каждый маршрутизатор использует алгоритм «кратчайший путь первым» для вычисления графа без петель, который будет описывать кратчайший путь к каждому известному пункту назначения с собой в качестве корня. Этот граф — дерево кратчайшего пути.
7. Каждый маршрутизатор строит таблицу маршрутизации из своего дерева кратчайшего пути.

Типы объявлений о состоянии канала

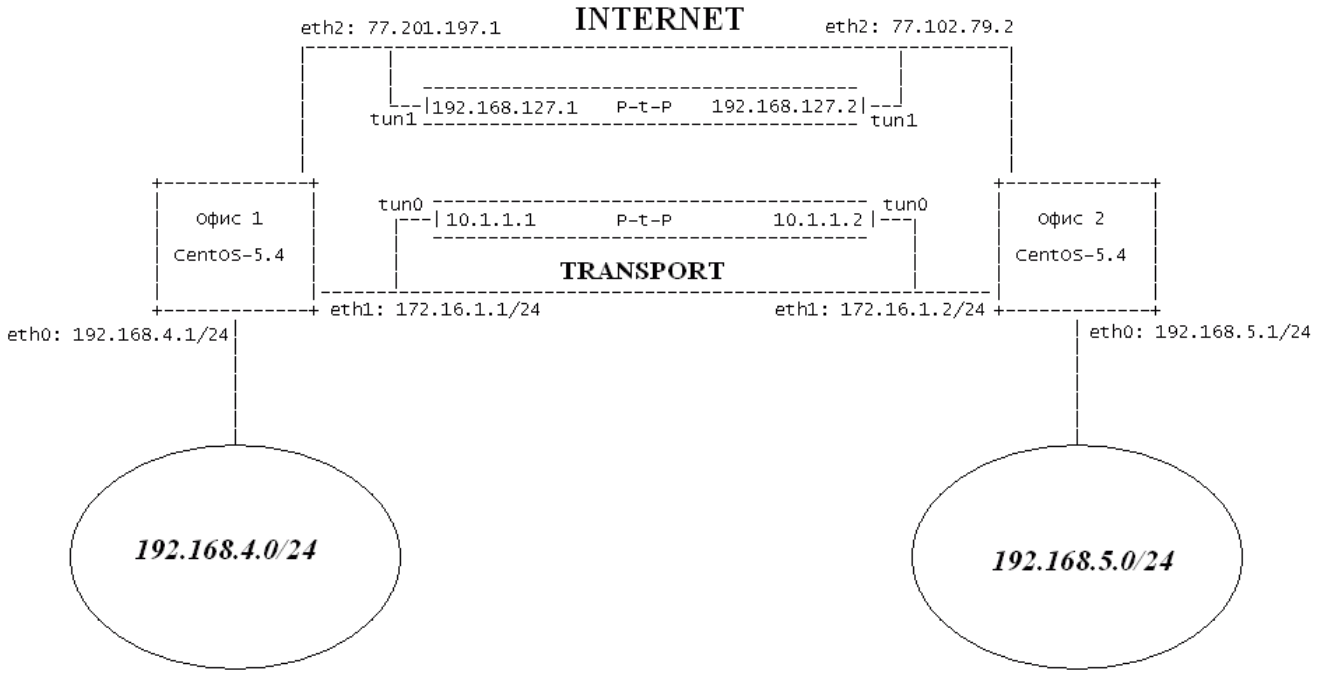
- Type 1 LSA — Router LSA — объявление о состоянии каналов маршрутизатора. Эти LSA распространяются всеми маршрутизаторами. В LSA содержится описание всех каналов маршрутизатора и стоимость (cost) каждого канала. Распространяются только в пределах одной зоны.

- Type 2 LSA — Network LSA — объявление о состоянии каналов сети. Распространяется DR в сетях со множественным доступом. В LSA содержится описание всех маршрутизаторов присоединенных к сети, включая DR. Распространяются только в пределах одной зоны.

- Type 3 LSA — Network Summary LSA — суммарное объявление о состоянии каналов сети. Объявление распространяется пограничными маршрутизаторами. Объявление описывает только маршруты к сетям вне зоны и не описывает маршруты внутри автономной системы. Пограничный маршрутизатор отправляет отдельное объявление для каждой известной ему сети.
- Type 4 LSA — ASBR Summary LSA — суммарное объявление о состоянии каналов пограничного маршрутизатора автономной системы. Объявление распространяется пограничными маршрутизаторами. ASBR Summary LSA отличается от Network Summary LSA тем, что распространяется информация не о сети, а о пограничном маршрутизаторе автономной системы.
- Type 5 LSA — AS External LSA — объявления о состоянии внешних каналов автономной системы. Объявление распространяется пограничным маршрутизатором автономной системы в пределах всей автономной системы. Объявление описывает маршруты внешние для автономной системы OSPF или маршруты по умолчанию (default route) внешние для автономной системы OSPF.
- Type 7 LSA — AS External LSA for NSSA — объявления о состоянии внешних каналов автономной системы в NSSA зоне. Это объявление может передаваться только в NSSA зоне. На границе зоны пограничный маршрутизатор преобразует type 7 LSA в type 5 LSA.

Для более детального ознакомления с OPSF рекомендую прочитать следующую статью http://xgu.ru/wiki/Ospf

Настройка Quagga



Устанавливаем необходимые пакеты на обоих серверах

```
# yum install quagga openvpn
```

Как настроить OpenVPN для режима point-to-point можно прочитать в соответствующей статье - OpenVPN: point to point

Перед непосредственной настройкой OSPF сначала произведем небольшую настройку самой quagga. Для этого редактируем файл **/etc/sysconfig/quagga**

```
#
# Default: Bind all daemon vtys to the loopback(s) only
#
QCONFDIR="/etc/quagga"
OSPFD_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/ospfd.conf"
ZEBRA_OPTS="-A 127.0.0.1 -f ${QCONFDIR}/zebra.conf"

# Watchquagga configuration (please check timer values before using):
WATCH_DAEMONS="zebra ospfd"
```

Так как quagga поддерживает не только OSPF, а также BGP, RIP, IS-IS, то по умолчанию в папке **/etc/quagga** будут заготовки для каждого из демонов. Для реализации нашей задачи нам необходимо оставить только два файла - **ospfd.conf** и **zebra.conf**.

Для того, чтобы начать настройку OSPF сначала мы должны запустить главный демон - zebra. Для этого создаем минимально необходимый для запуска конфигурационный файл, его содержимое приведено ниже

```
!
hostname router1.vmware.local
password zebra
log file /var/log/quagga/quagga.log
!
line vty
!
```

После этого можно запускать сам демон

```
# service zebra start
Starting zebra: Nothing to flush.
                                                           [  OK  ]
```

При этом в log-файле должна быть следующая строка

```
# cat /var/log/quagga/quagga.log
2009/10/29 19:08:57 ZEBRA: Zebra 0.99.15 starting: vty@2601
```

Теперь мы можем уже подключиться к vty на порт 2601

```
# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
```

```
Hello, this is Quagga (version 0.99.15).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password: *******
router1.vmware.local> enable
router1.vmware.local# show version
Quagga 0.99.15 (router1.vmware.local).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

router1.vmware.local# show running-config

Current configuration:
!
hostname router1.vmware.local
password zebra
log file /var/log/quagga/quagga.log
!
interface eth0
 ipv6 nd suppress-ra
!
interface eth1
 ipv6 nd suppress-ra
!
interface eth2
 ipv6 nd suppress-ra
!
interface lo
!
interface sit0
 ipv6 nd suppress-ra
!
interface tun0
 ipv6 nd suppress-ra
!
interface tun1
 ipv6 nd suppress-ra
!
!
!
line vty
!
end
```

Синтаксис команд очень схож с Cisco IOS. Как мы видим из вывода последней команды quagga определила все интерфейсы в системе. Все что нам необходимо это сохранить текущие настройки. Для этого выполняем следующую команду

```
router1.vmware.local# write
```

```
Configuration saved to /etc/quagga/zebra.conf
```

Для выхода с vty необходимо использовать комбинацию CTRL + D

Основы работы с vty

Если вы уже работали с Cisco IOS можете смело пропускать этот раздел.

Quagga предоставляет доступ к нескольким разным командным режимам. Каждый режим предоставляет свой набор команд. В целях безопасности введено два уровня доступа к командам: пользовательский и привилегированный. Не привилегированный пользовательский режим называется - «**user EXEC mode**». Привилегированный режим называется - «**privileged EXEC mode**».

Следующая таблица описывает наиболее часто используемые режимы, как в них зайти, и строку приглашения, которая характерна для данного режима. Строка приглашения помогает вам определить в каком режиме вы находитесь, и как следствие какие команды вам доступны.

Режим команд	Назначение	Как попасть в режим	Вид приглашения
User EXEC	Изменение основных настроек терминала, получение системной информации	telnet localhost 2604	router1.vmware.local>
Privileged EXEC	Системное администрирование, установка рабочих параметров	В режиме **User EXEC** набрать команду **enable**	router1.vmware.local#
Global Config	Изменение конфигурации, которое влияет на всю систему в целом	В режиме **Privileged EXEC** набрать команду **configure terminal** или **conf t**	router1.vmware.local(config)#
Interface Config	Изменение режима работы интерфейса	В режиме глобальной настройки набрать команду **interface *ifname***	router1.vmware.local(config-if)#
OSPF Config	Изменение настроек протокола OSPF	В режиме глобальной настройки набрать команду **router ospf**	router1.vmware.local(config-router)#

- **User EXEC Mode** - Когда вы подключаетесь к маршрутизатору вы попадаете в пользовательский режим. Набор команд пользовательского режима является подмножеством команд привилегированного режима.
- **Privileged EXEC Mode** - Команды привилегированного режима включают следующие:

Configure – изменение настроек маршрутизатора. **Debug** – отображение программных и аппаратных событий. Для выхода из привилегированного режима и возврата в пользовательский наберите команду **disable**.

- **Configuration Mode** - режим глобальной настройки имеет ряд под режимов для изменения настроек интерфейса, настроек протоколов маршрутизации и т.д. Внимательно и аккуратно используйте этот режим, так как все изменения немедленно отображаются на работе системы. Для выхода из режима используйте следующее сочетание клавиш - **Ctrl-Z**.

Практически все режимы настройки имеют отрицательную форму команд. В общем случае приставка no перед командой отключает функцию или возможность. Для включения функции, отключенной с помощью приставки no, используйте команду без приставки.

Например, OSPF отключен по умолчанию. Для включения этого режима используйте следующую команду

```
router1.vmware.local(config)# router ospf
```

Для отключения OSPF просто добавьте приставку no

```
router1.vmware.local(config)# no router ospf
```

В любом режиме вы можете получить список доступных команд набрав знак вопроса (?)

```
router1.vmware.local(config-if)# ?
  description  Interface specific description
  end          End current mode and change to enable mode.
  exit         Exit current mode and down to previous mode
  help         Description of the interactive help system
  ip           IP Information
  list         Print command list
  mpls-te      MPLS-TE specific commands
  no           Negate a command or set its defaults
  ospf         OSPF interface commands
  quit         Exit current mode and down to previous mode
  show         Show running system information
  write        Write running configuration to memory, network, or terminal
```

Для получения списка команд начинающихся с определенной последовательности символов, наберите эту последовательность, а потом без пробела знак вопроса

```
router1.vmware.local# co?
  configure  Configuration from vty interface
  copy       Copy configuration
```

Для получения списка аргументов команды наберите саму команду и через пробел наберите знак вопроса

```
router1.vmware.local(config)# ip ?
  prefix-list  Build a prefix list
```

```
router1.vmware.local(config)# ip prefix-list ?
  WORD             Name of a prefix list
  sequence-number  Include/exclude sequence numbers in NVGEN

# ip prefix-list TEST ?
  deny         Specify packets to reject
  permit       Specify packets to forward
  description  Prefix-list specific description
  seq          sequence number of an entry

# ip prefix-list TEST seq ?
  <1-4294967295>  Sequence number

# ip prefix-list TEST seq 10 ?
  deny    Specify packets to reject
  permit  Specify packets to forward
```

Вы также можете использовать аббревиатуры команд, для этого достаточно ввести символы, однозначно определяющие команду. Например, в привилегированном режиме для того, чтобы перейти в режим глобальной настройки достаточно набрать

```
router1.vmware.local# conf t
router1.vmware.local(config)#
```

Хотя вам никто не запрещает набирать команду полностью

```
router1.vmware.local# configure terminal
router1.vmware.local(config)#
```

Настройка OSPF

Для того, чтобы начать настройку демона OSPF сначала мы должны создать минимально необходимый для запуска конфигурационный файл - **/etc/quagga/ospfd.conf**, его содержимое приведено ниже

```
!
hostname router1.vmware.local
password zebra
log file /var/log/quagga/ospfd.log
!
line vty
!
```

После этого запускаем демон ospfd

```
# service ospfd start
Starting ospfd:                                            [  OK  ]
```

При этом в log-файле должна быть следующая строка

```
# cat /var/log/quagga/ospfd.log
2009/10/29 19:55:00 OSPF: OSPFd 0.99.15 starting: vty@2604
```

Из этой строки видно, что vty доступен на порту 2604.

```
# telnet localhost 2604
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is Quagga (version 0.99.15).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
router1.vmware.local> enable
router1.vmware.local# conf t
router1.vmware.local(config)# router ospf
router1.vmware.local(config-router)# ospf router-id 10.1.1.1
router1.vmware.local(config-router)# redistribute connected route-map
Local_Network
router1.vmware.local(config-router)# network 10.1.1.1/30 area 0
router1.vmware.local(config-router)# network 192.168.127.1/30 area 0
router1.vmware.local(config-router)# area 0.0.0.0 authentication message-
digest
router1.vmware.local(config-router)# write
Configuration saved to /etc/quagga/ospfd.conf
```

После этих команд OSPF должен быть активен на интерфейсах tun0 и tun1

```
router1.vmware.local# show ip ospf interface
eth0 is up
  ifindex 2, MTU 1500 bytes, BW 0 Kbit <UP,BROADCAST,RUNNING,MULTICAST>
  OSPF not enabled on this interface
eth1 is up
  ifindex 3, MTU 1500 bytes, BW 0 Kbit <UP,BROADCAST,RUNNING,MULTICAST>
  OSPF not enabled on this interface
eth2 is up
  ifindex 4, MTU 1500 bytes, BW 0 Kbit <UP,BROADCAST,RUNNING,MULTICAST>
  OSPF not enabled on this interface
lo is up
  ifindex 1, MTU 16436 bytes, BW 0 Kbit <UP,LOOPBACK,RUNNING>
  OSPF not enabled on this interface
sit0 is down
  ifindex 5, MTU 1480 bytes, BW 0 Kbit <NOARP>
  OSPF not enabled on this interface
tun0 is up
  ifindex 7, MTU 1500 bytes, BW 0 Kbit
```

```
<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>
  Internet Address 10.1.1.1/32, Peer 10.1.1.2, Area 0.0.0.0
  MTU mismatch detection:enabled
  Router ID 10.1.1.1, Network Type POINTOPOINT, Cost: 10
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No designated router on this network
  No backup designated router on this network
  Multicast group memberships: OSPFAllRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 1.405s
  Neighbor Count is 0, Adjacent neighbor count is 0
tun1 is up
  ifindex 6, MTU 1500 bytes, BW 0 Kbit
<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>
  Internet Address 192.168.127.1/32, Peer 192.168.127.2, Area 0.0.0.0
  MTU mismatch detection:enabled
  Router ID 10.1.1.1, Network Type POINTOPOINT, Cost: 10
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No designated router on this network
  No backup designated router on this network
  Multicast group memberships: OSPFAllRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 1.984s
  Neighbor Count is 0, Adjacent neighbor count is 0
```

О чем также свидетельствуют следующие строки в log-файле

```
2009/10/29 21:11:27 OSPF: interface 10.1.1.1 [7] join AllSPFRouters
Multicast group.
2009/10/29 21:11:38 OSPF: interface 192.168.127.1 [6] join AllSPFRouters
Multicast group.
```

Теперь вкратце опишу для чего каждая команда

1. router ospf - включаем процесс OSPF. На данный момент quagga не поддерживает множественные OSPF процессы
2. ospf router-id 10.1.1.1 - задаем id роутера. Этот id должен быть уникальным в пределах всего OSPF домена.
3. redistribute connected route-map Local_Network -
4. network 10.1.1.1/30 area 0 -
5. network 192.168.127.1/30 area 0 -
6. area 0.0.0.0 authentication message-digest -
7. write -

Теперь настраиваем интерфейсы, которые будет участвовать в OSPF процессе, а именно tun0 и tun1

```
router1.vmware.local# conf t
router1.vmware.local(config)# int tun0
router1.vmware.local(config-if)# ip ospf authentication message-digest
router1.vmware.local(config-if)# ip ospf message-digest-key 1 md5 SHdJLapbQ1
```

```
router1.vmware.local(config-if)# ip ospf cost 10
router1.vmware.local(config-if)# write
Configuration saved to /etc/quagga/ospfd.conf
```

Единственная разница между настройками двух интерфейсов, это значение параметра cost, т.е. т.н. стоимости передачи информации через интерфейс. Так как на интерфейсе tun0 стоимость 10, а на tun1 стоимость 20, то при доступности tun0 информация будет передаваться через данный интерфейс, так как у него стоимость ниже, чем у tun1. Если же интерфейс tun0 будет не доступен, то информацию будет передаваться через tun1.

```
router1.vmware.local# conf t
router1.vmware.local(config)# int tun1
router1.vmware.local(config-if)# ip ospf authentication message-digest
router1.vmware.local(config-if)# ip ospf message-digest-key 1 md5 SHdJLapbQ1
router1.vmware.local(config-if)# ip ospf cost 20
router1.vmware.local(config-if)# write
Configuration saved to /etc/quagga/ospfd.conf
```

```
router1.vmware.local# conf t
router1.vmware.local(config)# ip prefix-list Local_Network seq 10 permit
192.168.4.0/24
router1.vmware.local(config)# ip prefix-list Local_Network seq 100 deny any
router1.vmware.local(config)# write
Configuration saved to /etc/quagga/ospfd.conf
```

```
router1.vmware.local# conf t
router1.vmware.local(config)# route-map Local_Network permit 10
router1.vmware.local(config-route-map)# match ip address prefix-list
Local_Network
router1.vmware.local(config-route-map)# write
Configuration saved to /etc/quagga/ospfd.conf
```

На этом настройку OSPF процесса можно считать завершенной. В Офисе 2 настройка аналогичная за исключением, router-id и подсети в prefxi-list Local_Network.

Ниже привожу содержимое файла ospfd.conf для второго офиса

```
!
! Zebra configuration saved from vty
!   2009/10/19 21:42:15
!
hostname router2.vmware.local
password zebra
log file /var/log/quagga/ospfd.log
!
!
!
interface eth0
!
interface eth1
!
```

```
interface eth2
!
interface lo
!
interface sit0
!
interface tun0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 SHdJLapbQ1
 ip ospf cost 10
!
interface tun1
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 SHdJLapbQ1
 ip ospf cost 20
!
router ospf
 ospf router-id 10.1.1.2
 redistribute connected route-map Local_Network
 network 10.1.1.1/30 area 0.0.0.0
 network 192.168.127.1/30 area 0.0.0.0
 area 0.0.0.0 authentication message-digest
!
ip prefix-list Local_Network seq 10 permit 192.168.5.0/24
ip prefix-list Local_Network seq 100 deny any
!
route-map Local_Network permit 10
 match ip address prefix-list Local_Network
!
line vty
!
```

Тестирование

Как только вы запустите демон ospfd во втором офисе, в log-файле ospfd первого офиса появятся следующие сообщения

```
2009/10/30 19:52:30 OSPF: Packet[DD]: Neighbor 10.1.1.2 Negotiation done
(Slave).
2009/10/30 19:52:30 OSPF: Packet[DD]: Neighbor 10.1.1.2 Negotiation done
(Slave).
2009/10/30 19:52:30 OSPF: nsm_change_state(10.1.1.2, Loading -> Full):
scheduling new router-LSA origination
2009/10/30 19:52:30 OSPF: nsm_change_state(10.1.1.2, Loading -> Full):
scheduling new router-LSA origination
```

После этого вы должны увидеть router2.vmware.local в качестве соседа

```
router1.vmware.local# show ip ospf neighbor
```

```
    Neighbor ID Pri State           Dead Time Address         Interface
RXmtL RqstL DBsmL
10.1.1.2          1 Full/DROther      31.556s 10.1.1.2        tun0:10.1.1.1
0     0     0
10.1.1.2          1 Full/DROther      31.556s 192.168.127.2
tun1:192.168.127.1       0     0     0
```

Также мы должны видеть маршрут в подсеть 192.168.5.0/24

```
router1.vmware.local# show ip ospf route
============ OSPF network routing table ============
N    10.1.1.2/32           [10] area: 0.0.0.0
                           directly attached to tun0
N    192.168.127.2/32      [20] area: 0.0.0.0
                           directly attached to tun1

============ OSPF router routing table =============
R    10.1.1.2              [10] area: 0.0.0.0, ASBR
                           via 10.1.1.2, tun0

============ OSPF external routing table ===========
N E2 192.168.5.0/24        [10/20] tag: 0
                           via 10.1.1.2, tun0
```

После этого маршрут в данную подсеть должен быть виден и системе

```
# ip ro sh | grep zebra
192.168.5.0/24 via 10.1.1.2 dev tun0  proto zebra  metric 20
```

Чтобы убедиться что маршрутизация работает правильно используем команду ping

```
# ping -c4 192.168.5.1
PING 192.168.5.1 (192.168.5.1) 56(84) bytes of data.
64 bytes from 192.168.5.1: icmp_seq=1 ttl=64 time=0.765 ms
64 bytes from 192.168.5.1: icmp_seq=2 ttl=64 time=0.430 ms
64 bytes from 192.168.5.1: icmp_seq=3 ttl=64 time=0.444 ms
64 bytes from 192.168.5.1: icmp_seq=4 ttl=64 time=0.549 ms

--- 192.168.5.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 5760ms
rtt min/avg/max/mdev = 0.430/0.547/0.765/0.134 ms
```

Также можно убедиться, что OSPF процесс активен только на интерфейсах tun0 и tun1

```
# tcpdump -npi tun0 ip multicast
tcpdump: WARNING: arptype 65534 not supported by libpcap - falling back to
cooked socket
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type LINUX_SLL (Linux cooked), capture size 96 bytes
20:05:38.942443 IP 10.1.1.2 > 224.0.0.5: OSPFv2, Hello, length: 48
```

```
20:05:39.392188 IP 10.1.1.1 > 224.0.0.5: OSPFv2, Hello, length: 48
20:05:48.944342 IP 10.1.1.1 > 224.0.0.5: OSPFv2, Hello, length: 48
20:05:48.945603 IP 10.1.1.2 > 224.0.0.5: OSPFv2, Hello, length: 48
```

```
# tcpdump -npi tun1 ip multicast
tcpdump: WARNING: arptype 65534 not supported by libpcap - falling back to
cooked socket
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun1, link-type LINUX_SLL (Linux cooked), capture size 96 bytes
20:07:31.857488 IP 192.168.127.2 > 224.0.0.5: OSPFv2, Hello, length: 48
20:07:31.857650 IP 192.168.127.1 > 224.0.0.5: OSPFv2, Hello, length: 48
20:07:41.958583 IP 192.168.127.2 > 224.0.0.5: OSPFv2, Hello, length: 48
20:07:41.958793 IP 192.168.127.1 > 224.0.0.5: OSPFv2, Hello, length: 48
```

Как видно обмен Hello пакетами осуществляется каждые 10 секунд. Аналогичные проверки можно сделать и на маршрутизаторе во втором офисе.

Ну а теперь собственно наступает момент истины, ради которого мы и настраивали всю эту систему. Итак, напомню, что нашей задачей было обеспечить постоянную связь между офисами и автоматическое изменение маршрутизации, при падении основного канала.

Для того, чтобы проверить систему в работе, мы отключим интерфейс tun0 (основной канал). И посмотрим, что у нас произойдет с маршрутизацией.

```
# ifconfig tun0 down
```

При этом в ospfd.log появились следующие сообщения

```
2009/10/30 20:15:12 OSPF: nsm_change_state(10.1.1.2, Full -> Deleted):
scheduling new router-LSA origination
2009/10/30 20:15:12 OSPF: interface 10.1.1.1 [7] leave AllSPFRouters
Multicast group.
```

На маршрутизаторе во втором офисе я запустил команду ping

```
# ping 192.168.4.1
PING 192.168.4.1 (192.168.4.1) 56(84) bytes of data.
64 bytes from 192.168.4.1: icmp_seq=1 ttl=64 time=0.789 ms
64 bytes from 192.168.4.1: icmp_seq=2 ttl=64 time=0.456 ms
64 bytes from 192.168.4.1: icmp_seq=3 ttl=64 time=0.412 ms
64 bytes from 192.168.4.1: icmp_seq=7 ttl=64 time=0.626 ms
64 bytes from 192.168.4.1: icmp_seq=8 ttl=64 time=0.385 ms
64 bytes from 192.168.4.1: icmp_seq=9 ttl=64 time=0.898 ms
64 bytes from 192.168.4.1: icmp_seq=10 ttl=64 time=0.392 ms

--- 192.168.4.1 ping statistics ---
10 packets transmitted, 7 received, 30% packet loss, time 17232ms
rtt min/avg/max/mdev = 0.385/0.565/0.898/0.194 ms
```

Как видно, что за время переключения было потеряно всего 3 пакета.

А теперь давайте посмотрим таблицу маршрутизации в системе

```
# ip ro sh | grep zebra
192.168.5.0/24 via 192.168.127.2 dev tun1  proto zebra  metric 20
```

Теперь в подсеть 192.168.5.0/24 мы попадаем через интерфейс tun1.

```
router1.vmware.local# show ip ospf neighbor

    Neighbor ID Pri State           Dead Time Address         Interface
RXmtL RqstL DBsmL
10.1.1.2          1 Full/DROther      30.547s 192.168.127.2
tun1:192.168.127.1       0     0     0
```

Давай те посмотрим, что будет, когда интерфейс tun0 снова станет доступным.

Поднимаем интерфейс tun0

```
# ifconfig tun0 up
```

Смотрим информацию о соседях в ospfd

```
router1.vmware.local# show ip ospf neighbor

    Neighbor ID Pri State           Dead Time Address         Interface
RXmtL RqstL DBsmL
10.1.1.2          1 Full/DROther      30.330s 10.1.1.2        tun0:10.1.1.1
0     0     0
10.1.1.2          1 Full/DROther      30.330s 192.168.127.2
tun1:192.168.127.1       0     0     0
```

```
router1.vmware.local# show ip ospf route
============ OSPF network routing table ============
N    10.1.1.2/32           [10] area: 0.0.0.0
                           directly attached to tun0
N    192.168.127.2/32      [20] area: 0.0.0.0
                           directly attached to tun1

============ OSPF router routing table =============
R    10.1.1.2              [10] area: 0.0.0.0, ASBR
                           via 10.1.1.2, tun0

============ OSPF external routing table ===========
N E2 192.168.5.0/24        [10/20] tag: 0
                           via 10.1.1.2, tun0
```

```
# ip ro sh | grep zebra
192.168.5.0/24 via 10.1.1.2 dev tun0  proto zebra  metric 20
```

Как мы видим маршрутизация автоматически перенастроилась на основной канал через

интерфейс tun0

О чем также свидетельствуют следующие сообщения в log-файле

```
2009/10/30 20:23:45 OSPF: interface 10.1.1.1 [7] join AllSPFRouters
Multicast group.
2009/10/30 20:23:47 OSPF: Packet[DD]: Neighbor 10.1.1.2 Negotiation done
(Slave).
2009/10/30 20:23:47 OSPF: nsm_change_state(10.1.1.2, Exchange -> Full):
scheduling new router-LSA origination
```