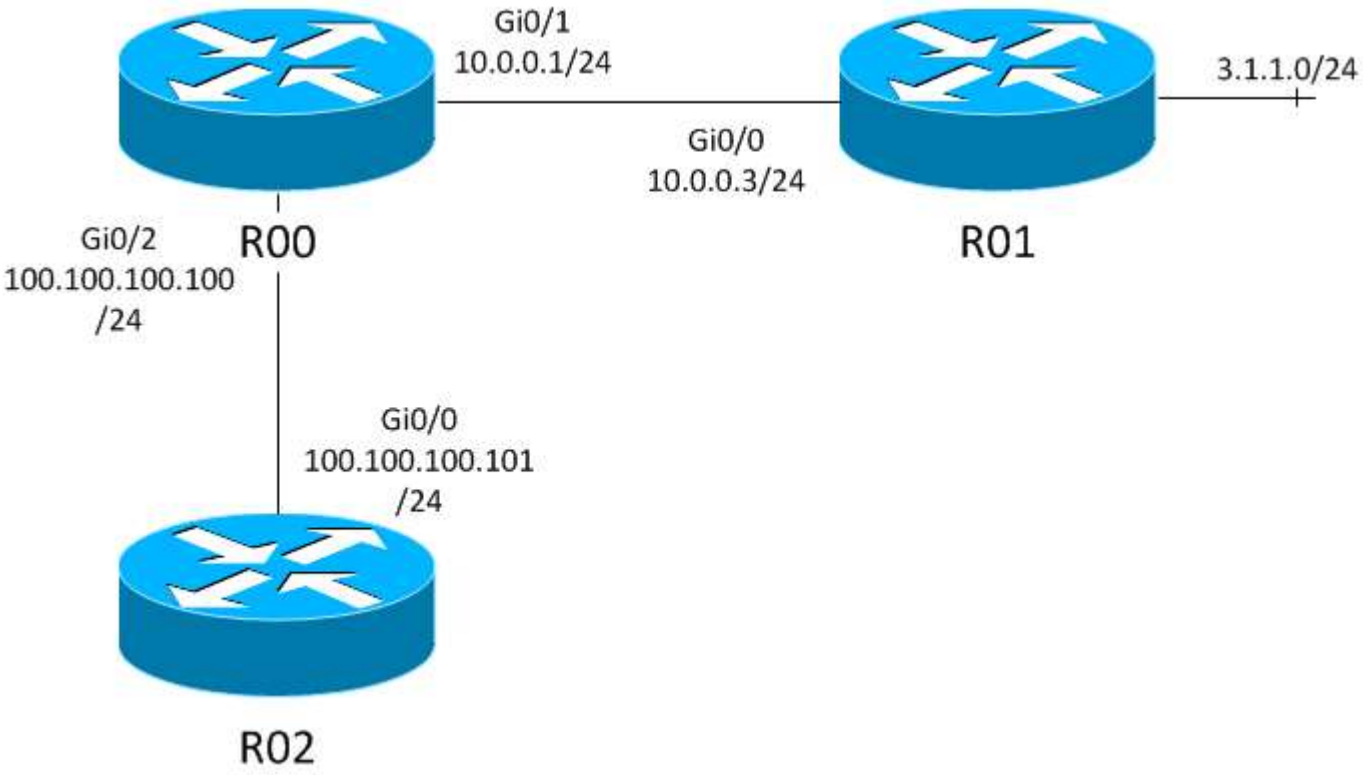
А вы хорошо знаете статическую маршрутизацию?

По материалам: https://habr.com/ru/post/174167/

Статический маршрут — первое, с чем сталкивается любой человек при изучении понятия маршрутизации IP пакетов. Считается, что это — наиболее простая тема из всех, в ней всё просто и очевидно. Я же постараюсь показать, что даже настолько примитивная технология может содержать в себе множество нюансов.

Оговорка. При написании топика я исхожу из того, что читатель знаком с концепцией маршрутизации, умеет делать статические маршруты и не считает слово «ARP» ругательным. Впрочем, даже бывалые связисты наверняка найдут тут что-то новое.
Все примеры были проверены на IOS линейки 15.2M. Поведение других ОС может различаться.
И никакого динамического роутинга тут не будет.

Мы работаем со следующей топологией:



Как появляется статический маршрут?

Для начала, выполним команду, которую знает каждый, и посмотрим дебагами, что произойдет:

```
R00(config)#ip route 3.1.1.0 255.255.255.0 10.0.0.3
```

```
IP-ST(default): updating same distance on 3.1.1.0/24
```

```
IP-ST(default):  3.1.1.0/24 [1], 10.0.0.3 Path = 8, no change, not active s
tate
```

```
IP-ST(default):  3.1.1.0/24 [1], 10.0.0.3 Path = 2 3 7
RT: updating static 3.1.1.0/24 (0x0):
    via 10.0.0.3
RT: add 3.1.1.0/24 via 10.0.0.3, static metric [1/0], add succeed, active s
tate
IP ARP: creating incomplete entry for IP address: 10.0.0.3 interface Gigabi
tEthernet0/1
IP ARP: sent req src 10.0.0.1 30e4.db16.7791,
                dst 10.0.0.3 0000.0000.0000 GigabitEthernet0/1
IP ARP: rcvd rep src 10.0.0.3 0019.aad6.ae10, dst 10.0.0.1 GigabitEthernet0
/1
```

IOS создал маршрут, и сразу послал arp запрос в поисках next hop, который у нас – 10.0.0.3. И сразу вопрос: откуда роутер узнал, что запрос надо слать в интерфейс Gi0/1? Наверняка кто-то скажет «из списка локальных интерфейсов», и жестоко ошибется. Маршрутизация так не работает. На самом деле, IOS сделал рекурсивный запрос к таблице маршрутизации, чтобы узнать, как добраться до next hop:

```
R00#show ip route 10.0.0.3
Routing entry for 10.0.0.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Routing Descriptor Blocks:
  * directly connected, via GigabitEthernet0/1
      Route metric is 0, traffic share count is 1
```

И вот он, наш Gi0/1. IOS узнает, что с рекурсивными запросами к RIB надо заканчивать, как только находит маршрут с флагом «directly connected». Но что если ему в ответ на изначальный запрос к 10.0.0.3 вернется вовсе не connected маршрут, а промежуточный, ссылающийся на другой next hop? Вернемся к этому чуть позже, а пока вспомним, что такое CEF.

Примерно во всей документации, ориентированной на начинающих, говорится, что каждый пакет перемещается в соответствии с таблицей маршрутизации. На самом деле на всех более-менее современных платформах это уже не так, ведь таблица маршрутизации (далее – RIB) вовсе не оптимизирована для быстрой передачи данных. Оценить масштаб бедствия позволяет эта таблица (хотя у process switching'а множество недостатков помимо неоптимальных запросов – например, постоянное переключение шедулера CPU между контекстами, что весьма затратно). CEF является серьезной оптимизацией. В современной реализации он строит две таблицы – FIB (Forwarding Information Base, таблица передачи пакетов, в основе нее – связный граф со страшным названием 256-way mtrie) и adjacency table (таблица соседств). Первая из них строится на основе таблицы маршрутизации и за один проход позволяет получить всю нужную информацию. Строится она заранее, еще до того, как появится первый соответствующий ей пакет.

Вернемся к нашему статическому маршруту. Вот запись в таблице маршрутизации:

```
R00#show ip route 3.1.1.0
Routing entry for 3.1.1.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 10.0.0.3
      Route metric is 0, traffic share count is 1
```

Куда слать пакет? Где искать 10.0.0.3? Непонятно. Надо еще раз запросить таблицу маршрутизации, на этот раз по поводу 10.0.0.3, и, если надо, выполнить еще несколько итераций, пока не выясним connected интерфейс. И вот примерно таким образом мы фактически в несколько раз снижаем производительность маршрутизатора.

А вот что говорит CEF:

```
R00#show ip cef 3.1.1.0 detail
3.1.1.0/24, epoch 0
  recursive via 10.0.0.3
    attached to GigabitEthernet0/1
```

Просто и лаконично. Есть интерфейс, есть next hop, к которому надо слать пакет. Что там говорилось про adjacency table?

```
R00#show adjacency 10.0.0.3 detail
Protocol Interface                 Address
IP       GigabitEthernet0/1        10.0.0.3(10)
                                   0 packets, 0 bytes
                                   epoch 0
                                   sourced in sev-epoch 2
                                   Encap length 14
                                   0019AAD6AE1030E4DB1677910800
                                   ARP
```

Обратим внимание на какую-то длинную последовательность в предпоследней строке. Что-то это напоминает… Смотрим mac 10.0.0.3:

```
R00#show arp | in 10.0.0.3
Internet  10.0.0.3                1   0019.aad6.ae10  ARPA   GigabitEtherne
t0/1
```

Смотрим свой mac адрес на gi0/1:

```
R00#show int gi0/1
GigabitEthernet0/1 is up, line protocol is up
  Hardware is CN Gigabit Ethernet, address is 30e4.db16.7791 (bia 30e4.db16
.7791)
```

Ага. Та страшная строка – всего лишь два мака, которые надо подставить в заголовок Ethernet на этапе инкапсуляции, и ethertype 0x0800, т.е. банальный IPv4. И в двух таблицах CEF есть абсолютно вся информация, какая нужна для успешной отправки пакета дальше по цепочке.

Если у кого-то возникнет вопрос, зачем железке держать сразу две таблицы вместо одной, то дам очевидный ответ: обычно у маршрутизатора мало интерфейсов (а заодно и соседей) и много маршрутов. Какой смысл тысячи раз дублировать одни и те же маки в FIB? Памяти много не бывает, особенно на аппаратных платформах, будь то новомодные ASR'ы или даже L3 свитчи линейки Catalyst. Все они задействуют CEF при передаче пакетов.

И кстати, вернемся на минутку к изначальному дебагу. Отключим CEF командой no ip cef (никогда так не делайте) и сравним результат:

```
IP-ST(default): updating same distance on 3.1.1.0/24
IP-ST(default):  3.1.1.0/24 [1], 10.0.0.100 Path = 8, no change, not active
state
IP-ST(default):  3.1.1.0/24 [1], 10.0.0.100 Path = 2 3 7
RT: updating static 3.1.1.0/24 (0x0):
    via 10.0.0.100
RT: add 3.1.1.0/24 via 10.0.0.100, static metric [1/0], add succeed, active
state
```

Маршрут добавлен. Arp запроса не было. И правильно – зачем RIB сдался mac адрес? Если пустить пинг до, к примеру, 3.1.1.1, то скорее всего будет так:

```
R00#ping 3.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.1.1.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/4 ms
```

Первый пакет отбрасывается, и роутер посылает arp запрос с целью узнать mac адрес 10.0.0.3, если он ранее не был известен. CEF же всегда заранее узнает mac адрес next hop'a.

С этим разобрались. Теперь вернемся к вопросу, что будет, если next hop статического маршрута вовсе не на directly connected интерфейсе. Поступим просто:

```
R00(config)#ip route 10.0.0.3 255.255.255.255 100.100.100.101
```

, где Gi0/2 имеет адрес 100.100.100.100/24.

```
R00#show ip cef 3.1.1.0 detail
3.1.1.0/24, epoch 0
  recursive via 10.0.0.3
    recursive via 100.100.100.101
      recursive via 100.100.100.0/24
        attached to GigabitEthernet0/2
```

Как все плохо-то… А что если у нас есть маршрут на целую суперсеть?

```
R00(config)#no ip route 10.0.0.3 255.255.255.255 100.100.100.101
R00(config)#ip route 10.0.0.0 255.0.0.0 100.100.100.101
```

```
R00#show ip cef 3.1.1.0 detail
3.1.1.0/24, epoch 0
  recursive via 10.0.0.3
    attached to GigabitEthernet0/1
```

Сейчас наша таблица маршрутизации выглядит так:

```
R00#show ip route
      10.0.0.0/8 is variably subnetted, 2 subnets, 3 masks
S        10.0.0.0/8 [1/0] via 100.100.100.101
C        10.0.0.0/24 is directly connected, GigabitEthernet0/1
L        10.0.0.1/32 is directly connected, GigabitEthernet0/1
      100.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        100.100.100.0/24 is directly connected, GigabitEthernet0/2
L        100.100.100.100/32 is directly connected, GigabitEthernet0/2
```

Вроде хорошо. Новый маршрут на 100.100.100.101 не применяется для 10.0.0.3, так как его маска /8 намного короче, чем /24 у connected интерфейса. Но вдруг Gi0/1, содержавший next hop для 3.1.1.0/24, по какой-то непонятной причине ушел в down, и его connected маршрут пропал из RIB.

```
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to administrat
ively down
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed
state to down
RT: interface GigabitEthernet0/1 removed from routing table
RT: del 10.0.0.0 via 0.0.0.0, connected metric [0/0]
RT: delete subnet route to 10.0.0.0/24
RT: del 10.0.0.1 via 0.0.0.0, connected metric [0/0]
RT: delete subnet route to 10.0.0.1/32
```

```
IP-ST(default):  updating GigabitEthernet0/1
```

Вот что стало:

```
R00#show ip cef 3.1.1.0 detail
3.1.1.0/24, epoch 0
  recursive via 10.0.0.3
    recursive via 10.0.0.0/8
      recursive via 100.100.100.101
        recursive via 100.100.100.0/24
          attached to GigabitEthernet0/2
```

Ой. Теперь пакеты на сеть 3.1.1.0/24 идут куда-то не туда. Я не могу представить себе сценарий, когда ожидаемое поведение статического маршрута – переключение на другой интерфейс. Если за тем интерфейсом находится резервный путь, то все-таки надо создавать еще один статический маршрут…

Что делать? Указывать сразу в маршруте интерфейс. Пересоздадим маршрут:

```
R00(config)#no ip route 3.1.1.0 255.255.255.0 10.0.0.3
R00(config)#ip route 3.1.1.0 255.255.255.0 Gi0/1 10.0.0.3
```

Поднимаем Gi0/1. Смотрим, куда теперь ведет маршрут на 3.1.1.0/24:

```
R00#show ip route 3.1.1.0
Routing entry for 3.1.1.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
```

```
  * 10.0.0.3, via GigabitEthernet0/1
```

```
      Route metric is 0, traffic share count is 1
```

Тут уже указан интерфейс. Поэтому не будет рекурсивных запросов к таблице маршрутизации. Проверяем FIB:

```
R00#show ip cef 3.1.1.0
```

```
3.1.1.0/24
```

```
  nexthop 10.0.0.3 GigabitEthernet0/1
```

Да, никакого «recursive». А если снова погасить gi0/1? Маршрут исчез.

```
R00#show ip route 3.1.1.0
```

```
% Network not in table
```

```
R00#show ip cef 3.1.1.0
```

```
0.0.0.0/0
```

```
  no route
```

И это притом, что маршрут до 10.0.0.3 все еще был:

```
R00#show ip cef 10.0.0.3
```

```
10.0.0.0/8
```

```
  nexthop 100.100.100.101 GigabitEthernet0/2
```

А что будет, если путь к next hop даст маршрут по умолчанию, а маршрут на 3.1.1.0/24 не ссылается на интерфейс?

```
R00(config)#no ip route 10.0.0.0 255.0.0.0 100.100.100.101
```

```
R00(config)#ip route 0.0.0.0 0.0.0.0 100.100.100.101
R00(config)#no ip route 3.1.1.0 255.255.255.0 Gi0/1 10.0.0.3
R00(config)#ip route 3.1.1.0 255.255.255.0 10.0.0.3
R00#show ip route 3.1.1.0
% Network not in table
R00#show ip cef 3.1.1.0 detail
0.0.0.0/0, epoch 0, flags default route
  recursive via 100.100.100.101
    recursive via 100.100.100.0/24
      attached to GigabitEthernet0/2
```

Обратите внимание, что первой строкой после «show ip cef» идет «0.0.0.0/0», а не «3.1.1.0/24». Несмотря на то, что next hop формально есть, по факту все итерации опроса таблицы маршрутизации (кроме первой) игнорируют маршрут по умолчанию, что логично, иначе любой запрос к таблице маршрутизации почти всегда бы резолвился (под «резолвиться» понимается нахождение интерфейса, в который нужно отправить пакет). Поэтому наш статический маршрут отсутствует, но пакеты все равно улетают к Gi0/2. Вроде бы все то же самое, что и без явного указания интерфейса? Не совсем. Допустим, протоколу маршрутизации сказали «redistribute static». Если статический маршрут пропал, то анонс тоже отзывается. А если нет, то маршрутизатор продолжит говорить всем «туда идти через меня», и это почти наверняка обернется L3 кольцом для префикса 3.1.1.0/24, который мог бы быть доступен откуда-нибудь еще. Но стоп, мы договаривались не трогать динамический роутинг…

А что если в статическом маршруте указать интерфейс, но не указывать IP адрес следующего хопа? Ответ: в случае Ethernet, если на next hop не отключен proxy arp, связность не нарушится, но роутеру может ОЧЕНЬ поплохеть. Подробнее. Если сказать «ip route 3.1.1.0 255.255.255.0 gi0/1», то ничего особо страшного не случится, даже пару сотен записей в arp таблице любой роутер переварит (и существуют сценарии-workaround'ы, в которых оптимальным решением является именно такой костыль), но вот «ip route 0.0.0.0 0.0.0.0 gi0/1» на пограничном маршрутизаторе наверняка убьет его. Потому запомните общее правило: если создается статический маршрут с next hop'ом на Ethernet интерфейсе, то его IP адрес должен указываться всегда. Исключения – только когда вы очень хорошо представляете себе, что делаете, зачем делаете и почему нельзя сделать иначе.

И напоследок, сделаем одну очень нехорошую штуку.

```
R00(config)# ip route 3.1.1.0 255.255.255.0 10.0.0.3
```

```
R00(config)#ip route 10.0.0.3 255.255.255.255 3.1.1.1
```

Первый маршрут в порядке, сто раз протестирован. А вот второй странный – он ведет через первый. А первый теперь ссылается на второй, и у нас бесконечная рекурсия. Вот что произошло:

```
IP-ST(default): updating same distance on 3.1.1.0/24
IP-ST(default):  3.1.1.0/24 [1], 10.0.0.3 Path = 8, no change, not active s
tate
IP-ST(default):  3.1.1.0/24 [1], 10.0.0.3 Path = 2 3 7
RT: updating static 3.1.1.0/24 (0x0):
    via 10.0.0.3
RT: add 3.1.1.0/24 via 10.0.0.3, static metric [1/0], add succeed, active s
tate
IP-ST(default): updating same distance on 10.0.0.3/32
IP-ST(default):  10.0.0.3/32 [1], 3.1.1.1 Path = 8, no change, not active s
tate
IP-ST(default):  10.0.0.3/32 [1], 3.1.1.1 Path = 2 3 7
RT: updating static 10.0.0.3/32 (0x0):
    via 3.1.1.1
RT: add 10.0.0.3/32 via 3.1.1.1, static metric [1/0], add succeed, active s
tate
```

Добавилось успешно. Но затем в дебагах высветилось:

```
RT: recursion error routing 3.1.1.1 - probable routing loop
```

```
RT: recursion error routing 10.0.0.3 - probable routing loop
```

И появилась запись в лог с severity 3:

```
%IPRT-3-RIB_LOOP: Resolution loop formed by routes in RIB
```

```
R00#show ip cef 10.0.0.3 detail
```

```
10.0.0.3/32, epoch 0
```

```
  Adj source: IP adj out of GigabitEthernet0/1, addr 10.0.0.3 359503C0
```

```
   Dependent covered prefix type adjfib cover 10.0.0.0/24
```

```
  1 RR source [no flags]
```

```
  recursive via 3.1.1.1, unresolved
```

```
  recursive-looped
```

```
R00#show ip cef 3.1.1.0 detail
```

```
3.1.1.0/24, epoch 0, flags cover dependents
```

```
  Covered dependent prefixes: 1
```

```
    notify cover updated: 1
```

```
  recursive via 10.0.0.3, unresolved
```

```
  recursive-looped
```

Однако, RIB никакого криминала не видит:

```
Routing entry for 3.1.1.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 10.0.0.3
      Route metric is 0, traffic share count is 1
```

```
R00#show ip route 10.0.0.3
Routing entry for 10.0.0.3/32
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
  * 3.1.1.1
      Route metric is 0, traffic share count is 1
```

Вывод – никогда так не делайте.

Почему статический маршрут может не попасть в таблицу маршрутизации?

Любой сетевик должен сходу дать одно из объяснений, касающееся любого источника маршрутов в IOS: существует другой маршрут на тот же самый префикс, но с меньшим AD (все помнят Administrative Distance?). Маршрут, источник которого – “connected”, всегда имеет AD=0, и ни один другой источник маршрутов не может привнести ничего ниже, чем «1», даже статический маршрут с явным указанием интерфейса. Пример connected:

```
R00# show ip route
C        10.0.0.0/24 is directly connected, GigabitEthernet0/1
```

Т.е. пока интерфейс Gi0/1 находится в состоянии up и имеет адрес из подсети 10.0.0.0/24, ни один статический маршрут на этот префикс в таблице маршрутизации не появится.

Еще есть вариант «разные источники маршрутов добавляют маршруты на один и тот же префикс с одинаковым AD». Поведение IOS в данном случае не документировано, общая рекомендация –

«никогда так делайте».

Но посмотрим другие, менее очевидные примеры. Например, статические маршруты можно создать со словом «permanent», которое переводится как «постоянный», и тогда они будут всегда висеть в таблице маршрутизации. Правильно? Нет.

Добавляем его и смотрим:

```
R00(config)#ip route 3.1.1.0 255.255.255.0 10.0.0.3 permanent
```

```
R00#show ip cef 3.1.1.0
```

```
3.1.1.0/24
```

```
  nexthop 10.0.0.3 GigabitEthernet0/1
```

Кладем Gi0/1, и видим:

```
R00#show ip cef 3.1.1.0
```

```
3.1.1.0/24
```

```
  unresolved via 10.0.0.3
```

В RIB он есть, и другие протоколы маршрутизации могут его использовать:

```
R00#show ip route 3.1.1.0
```

```
Routing entry for 3.1.1.0/24
```

```
  Known via "static", distance 1, metric 0
```

```
  Routing Descriptor Blocks:
```

```
  * 10.0.0.3, permanent
```

```
      Route metric is 0, traffic share count is 1
```

А теперь, не поднимая Gi0/1:

```
R00(config)#no ip route 3.1.1.0 255.255.255.0 10.0.0.3 permanent
```

```
R00(config)#ip route 3.1.1.0 255.255.255.0 10.0.0.3 permanent
```

Просто пересоздали его, ничего не меняя. И вот что произошло:

```
IP-ST(default): updating same distance on 3.1.1.0/24
```

```
IP-ST(default):  3.1.1.0/24 [1], 10.0.0.3 Path = 8, no change, not active s
tate
```

```
IP-ST(default): cannot delete, PERMANENT
```

```
R00#show ip route 3.1.1.0
```

```
% Network not in table
```

```
R00#show ip cef 3.1.1.0
```

```
0.0.0.0/0
  no route
```

Постоянный, говорите? Нет. Есть один маленький нюанс: чтобы перманентный маршрут навеки вписался в таблицу маршрутизации, нужно, чтобы он хотя бы на долю секунды резолвился. Хотя какое еще «навеки»? Когда он остался висеть в воздухе без резолвящегося интерфейса, достаточно сказать «clear ip route *» или тем более «reload», чтобы он исчез из RIB.

Но продолжим. Сделаем вот так:

```
R00(config)#ip route 3.1.1.0 255.255.255.0 Gi0/1 10.0.0.3
```

```
R00(config)#ip route 3.1.1.10 255.255.255.255 3.1.1.1
```

Вроде нормальные маршруты. Что произойдет? Со вторым – ровным счетом ничего.

```
IP-ST(default):  3.1.1.10/32 [1], 3.1.1.1 Path = 8, no change, not active s
tate
IP-ST(default):  3.1.1.10/32 [1], 3.1.1.1 Path = 2 3 6 8, no change, not ac
tive state
```

Суть вот в чем. Допустим, есть маршрут на X.X.X.X через Y.Y.Y.Y. Мы добавляем маршрут на X1.X1.X1.X1 (этот префикс полностью покрывается X.X.X.X) через X2.X2.X2.X2 (а он тоже покрывается X.X.X.X). IOS делает закономерный вывод: второй маршрут не несет в себе никакой новой информации и совершенно бесполезен, поэтому его можно не устанавливать в RIB.

А теперь финт ушами.

```
R00(config)#no ip route 3.1.1.10 255.255.255.255 3.1.1.1
R00(config)#ip route 3.1.1.10 255.255.255.255 Gi0/1 3.1.1.1

IP-ST(default): updating same distance on 3.1.1.10/32
IP-ST(default):  3.1.1.10/32 [1], GigabitEthernet0/1 Path = 1
RT: updating static 3.1.1.10/32 (0x0):
    via 3.1.1.1 Gi0/1
RT: network 3.0.0.0 is now variably masked
RT: add 3.1.1.10/32 via 3.1.1.1, static metric [1/0], add succeed, active s
tate
IP ARP: creating incomplete entry for IP address: 3.1.1.1 interface Gigabit
Ethernet0/1
IP ARP: sent req src 10.0.0.1 30e4.db16.7791,
```

```
                dst 3.1.1.1 0000.0000.0000 GigabitEthernet0/1
```

```
R00#show ip cef 3.1.1.10
```

```
3.1.1.10/32
```

```
  nexthop 3.1.1.1 GigabitEthernet0/1
```

И вот это подводит нас к еще одному важному моменту. Указание интерфейса в статическом маршруте позволяет обойти многие проверки, так как статическому маршруту больше не требуется выполнять рекурсивные запросы к RIB в поисках пути до next hop, и при своем добавлении он не заденет триггеры на других маршрутах. Но это не отменяет главного требования: next hop обязан резолвиться в конкретный интерфейс, а тот интерфейс обязан быть в up. Тот факт, что рекурсивных запросов к RIB больше не будет, означает, что указанный IP адрес next hop'а находится прямо за интерфейсом, и наверняка отзовется на arp запрос (с точки зрения роутера). Если у соседнего по Gi0/1 роутера включен proxy arp, то он в ответ на arp запрос наверняка вернет свой mac адрес, и всё будет хорошо. Разве что лишняя запись в arp таблице…

Но все равно так делать не стоит.

Необходимо упомянуть и о еще одном важном моменте. Статический маршрут должен по идее исчезнуть из таблицы маршрутизации, как только он перестанет резолвиться. Но на практике есть множество ситуаций, когда next hop пропадает, но при этом статический маршрут на какое-то время остается. К примеру, когда next hop резолвится через маршрут, полученный от протокола динамической маршрутизации. Все дело в том, что процесс, отслеживающий наличие next hop в RIB, не всегда может получить уведомление об исчезновении маршрута, и он вынужден периодически (раз в 60 секунд по умолчанию) перепроверять, все ли хорошо. Это вызовет заметную задержку сходимости сети.

Поменять интервал проверки, к примеру, на 10 секунд можно с помощью команды:

```
ip route static adjust-time 10
```