# NSM路由

1. 静态路由  
   2.ospf  
   3.ecmp选路  
   4.链路状态变化（down、up）  
   5.recursive

6.MPLS路由

rib,nexthop struct

struct rib

{

...

u\_char type; 标记rib类型，如connect,ospf,rip等等

u\_char distance;

u\_char flags; 记录fib的标志RIB\_FLAG\_SELECTED，CHANGED

u\_int32\_t metric;

u\_char nexthop\_num; nexthop总数量

u\_char nexthop\_active\_num; nexthop active的数量

u\_char nexthop\_fib\_num; nexthop 被安装的数量

u\_char nexthop\_fib\_quota; nexthop 经过计算后的配额

struct nexthop \*nexthop;

...

}

struct nexthop

{

...

u\_int8\_t type; 记录type为IFINDEX ，IFNAME,IPV4

u\_int8\_t flags; 记录是ACTIVE，FIB，RECURSIVE

struct nsm\_ptree\_node \*rrn; 记录递归下的node

gate; 记录gateway地址

...

}

## 静态路由

Gige 5/0 ip 10.10.10.1

ip route 3.3.3.0/24 10.10.10.2

1. nsm\_cli\_ip\_route\_prefix 检查ip4地址合法性，获取ip4 prefix，检查gw的合法性
2. nsm\_ipv4\_route\_set 查询vrf表，获取到一个nm(vrf)表，然后将static加入到vrf中
3. nsm\_static\_add
4. nsm\_static\_get 检查下一跳的类型是否合法(看是地址还是接口名字 inf index等)，使用ip4 prefix在tree ip\_static中查找static的rib。因为是新加的路由，所以会ip\_static在中新建一个node。
5. nsm\_static\_nexthop\_same 在ip\_static找到的rib中去匹配这个新增的static 路由，看是否有重复的，有的话就什么都不做。因为是新增，所以不会匹配到。
6. nsm\_static\_new 新建一个nsm\_static结构体然后把新的路由信息赋值(gateway,distance等等)给它
7. nsm\_static\_nexthop\_add\_sort 将这个nsm\_static加入到ip\_static中，会进行排序，按照谁的gateway更小，谁的vrf\_id更小，谁的distance更小进行排序插入
8. nsm\_static\_install 先在nsm\_ptree\_table rib中通过prefix查找到对应的结构体rib node
9. nsm\_rib\_static\_lookup 在rib node中通过nsm\_static结构体的信息查找这个路由是否已经加过，由于是新路由，所以不会找到
10. nsm\_rib\_new 为这个路由新建一个struct rib,，并将nsm\_static的值赋给结构体。
11. casa\_ribnode\_add\_sort 按照更小distance和mertic的优先级，将rib插入到rib node中
12. nsm\_rib\_static\_nexthop\_add 将这个路由的nexthop信息加入到rib的nexthop链中
13. nsm\_rib\_process 调用casa\_rib\_select\_check ，nsm\_nexthop\_active\_update，nsm\_nexthop\_active\_ipv4\_vrf检查下一跳是否active，递归找到prefix为10.10.10.0的node，路由为直连（type==NEXTHOP\_TYPE\_IPV4\_IFINDEX ，active）。
14. nsm\_rib\_process\_add 在这里被设置RIB\_FLAG\_SELECTED标志位
15. nsm\_rib\_install\_kernel 将这条路由安装到kernel去
16. nsm\_rib\_fib\_add 检测是否还有空间加这条路由
17. nsm\_fea\_ipv4\_add
18. pal\_kernel\_ipv4\_add 调用netlink\_route\_multipath 用netlink装载到kernel中
19. nl\_route\_info\_add\_ipv4 在这里组装netlink包，并使用msg\_enqueue将msg入队发送。
20. zebra\_remote\_route\_add发送给zebra remote
21. create\_info\_ipv4装载路由信息到zebra\_remote\_msg\_t msg
22. msg\_enqueue\_efd\_post 加入zebra发送队列，发送

Static

补充：

1. 直连路由建立
2. netlink\_route\_multipath的逻辑
3. msg dequeue的接收
4. zebra remote消息的接收
5. Add delete merge功能

### 直连路由建立

ip address 7.7.7.1 255.255.255.0

1. nsm\_ip\_address\_install 根据ifname查询ip，查是否有ip和要设置的ip重复，检测到if running跑2
2. nsm\_connected\_up\_ipv4
3. nsm\_rib\_add\_connected\_ipv4 保存地址，新建一个rib为IPI\_ROUTE\_CONNECT，给rib加上nexthop,nexthop类型为NEXTHOP\_TYPE\_IFINDEX
4. nsm\_rib\_add
5. nsm\_rib\_process

### netlink\_route\_multipath的逻辑

1.判断rib flags是否是黑洞路由，若是，给rib nexthop设置 NEXTHOP\_FLAG\_FIB，然后调用nl\_route\_info\_add\_ipv4安装。

1. 检测rib的递归下一跳，nexthop 是NEXTHOP\_FLAG\_RECURSIVE并且是RTM\_NEWROUTE，检查rib->nexthop->rrn是否有，还要检查其是否被设置RIB\_FLAG\_SELECTED。如果没递归，则nl\_route\_info\_add\_ipv4进行安装。
2. 如有递归，调用casa\_netlink\_route\_multipath会检查rib flag是否是NEXTHOP\_FLAG\_RECURSIVE，是的话会安装其nexthop->rrn->info flags为RIB\_FLAG\_SELECTED的rib，然后再次调用casa\_netlink\_route\_multipath，casa\_netlink\_rhop\_exist\_in\_list检查找到的recursive rib的NEXTHOP\_FLAG\_FIB nexthop是否已经在rnh\_list中。如果已经有了，代表这个递归已经安装了，检查下一个nexthop，如list中没有，则用nl\_route\_info\_add\_ipv4安装。

### msg dequeue

1. casa\_nl\_route\_install\_main
2. get\_high\_priority\_message 这里读取最高优先级的ipv4 ipv6 connected down事件
3. msg\_deqeueue\_timedwait 调用sem\_trywait等待信号量锁定，等待msg\_enqueue的sem\_post
4. nl\_route\_info\_ipv4\_process 用新的node 在route\_cache\_ipv4，看是否有过一个一样的，有的话就nl\_route\_info\_cache\_update，没有就nl\_route\_info\_cache\_add
5. route\_install
6. route\_install\_ipv4 检查cache node中的nexthop的ifindex是否running,running状态才可以安装
7. netlink\_route

### zebra remote消息的接收

1. zebra\_client\_main 接收消息
2. client\_parse\_msg 对消息进行解析
3. zremote\_decode\_route\_ipv4 解封装msg中的路由信息
4. zebra\_client\_route\_cb msg\_enqueue(&msg, zebra\_circ\_q\_id\_recv); 将接收到的消息保存起来后入队
5. zebra\_client\_read\_timer msg\_deqeueue\_timedwait 在read timer出队，并根据msg.mtype进入相应处理函数
6. nsm\_sibling\_rib\_ipv4
7. nsm\_rib\_process

### Zebra remote 消息的发送

1. zebra\_remote\_main 调用event\_new(base, efd, EV\_READ | EV\_PERSIST, efd\_msg\_deqeueue, zebra\_circ\_q\_id\_send);读取zebra\_circ\_q\_id\_send 消息队列
2. efd\_msg\_deqeueue
3. process\_ipv4\_route\_add 保存zebra\_route\_info\_ipv4\_t 到tmp\_entry
4. info\_tree\_delete\_cp 把tmp\_entry 从ipv4\_info\_tree中删除。
5. 然后对tmp\_entry赋值out\_label，vpn\_label。并判断flag NEXTHOP\_DIRECT\_VPN来设置不同的flags，给带flags NEXTHOP\_DIRECT\_VPN的去掉NEXTHOP\_RECURSIVE；给不是NEXTHOP\_RECURSIVE，但有vpn label的设置NEXTHOP\_RECURSIVE。
6. info\_tree\_add 把tmp\_entry 从ipv4\_info\_tree中加回来。
7. message\_send\_consolidate 发送给standby
8. nsm\_ffe\_client\_routes\_update\_sync 轮询所有ffe client ，并用nsm\_ffe\_msg\_send发送
9. ffe\_client\_msg\_create
10. zremote\_encode\_route\_ipv4 里面用TLV\_ENCODE\_PUTX 把各种info 装进pnt，pnt是一个char数组，encode时按固定顺序吧info中的信息写进去，到时读取的时候也要按顺序读取。最后计算出msg len
11. casa\_nsm\_ffe\_message\_encode\_header 给消息体安装一个头部。
12. write client->sock\_fd 发送

为什么发送给DP的时候，还要加到一个ipv4\_info\_tree里面，是因为这个tree记录的是转换格式后的路由格式吗  
  
1.主要是为了 standby cp/dp 重新连接的时候，能直接从 remote 同步多一次，不用主线程重新算。  
2.有一些简单的转换逻辑

1. zebra\_remote\_route\_ftn4\_add
2. msg\_enqueue\_efd\_post(&msg, zebra\_circ\_q\_id\_send) 调用 write(efd, &u, sizeof(uint64\_t))
3. zebra\_remote\_main 会收到efd消息，然后调用efd\_msg\_deqeueue
4. efd\_msg\_deqeueue 定时用read消息，如果有路由要添加，则调用efd\_msg\_deqeueue
5. nsm\_ffe\_client\_routes\_update\_sync 发送给ffe
6. nsm\_ffe\_msg\_send 会用write 写到sock\_fd，但如果写不成功（errno == EAGAIN || errno == EWOULDBLOCK || errno == EINTR），则会用nsm\_ffe\_server\_enqueue把没写成功的消息入队
7. efd\_msg\_deqeueue 会被周期性的调用（10ms），如上，若read没消息，那么将会进行client的调度nsm\_ffe\_client\_schedule
8. nsm\_ffe\_client\_schedule 将会检查ffe clinet是否disconnect了，并且还会把enqueue的消息用nsm\_ffe\_server\_dequeue重发。

## OSPF

R1:

Gige 5/0 ip 10.10.10.1

router ospf 1

network 10.10.10.0/24 area 0

R2:

Gige 5/0 ip 10.10.10.2

Lo20 ip 3.3.3.3

router ospf 1

network 10.10.10.0/24 area 0

在R2运行

network 3.3.3.3/32 area 0

1. nsm\_rib\_set\_server\_callback 注册callback nsm\_parse\_route\_ipv4 nsm\_server\_recv\_route\_ipv4
2. message\_server\_read 收到ospfd的消息，调用MESSAGE\_EVENT\_READ\_HEADER callback读取(nsm\_server\_read\_msg)
3. nsm\_server\_read\_msg 读取nsm\_msg\_header 获取vr\_id,vrf\_id,msg type（31 NSM\_MSG\_ROUTE\_IPV4）等信息，读出msg body，调用对应TYPE的parser callback。
4. nsm\_parse\_route\_ipv4 parse出nsm\_msg\_route\_ipv4中的flags(为1 ==NSM\_MSG\_ROUTE\_FLAG\_ADD),prefix\_len,prefix,type,distance,metric等值。然后调用callback
5. nsm\_server\_recv\_route\_ipv4 通过获取到的vr\_id找到nm(vrf)表，新建struct rib，将路由信息填入(type==IPI\_ROUTE\_OSPF,flags,distance,metric,uptime等等)。转化nexthop的信息为rib的格式。
6. nsm\_nexthop\_ipv4\_ifindex\_add 这里把nexthop信息保存下来（把type设置为NEXTHOP\_TYPE\_IPV4\_IFINDEX，保存gateway ip，ifindex）
7. nsm\_nexthop\_add 将这个nexthop加入到struct rib的nexthop链表中
8. nsm\_rib\_add 试图把struct rib加入到rib tree中。先在 vrf->afi[AFI\_IP].rib[SAFI\_UNICAST]用prefix，搜索是否已经有同个prefix的nsm\_ptree\_node已经被加入tree。由于是新加入，所以返回一个新的nsm\_ptree\_node。然后搜索这个node是否已经有一个相同的struct rib。如果有会free掉接收消息时新建的rib内存，然后退出。由于是新增所以不会搜索到。
9. casa\_ribnode\_add\_sort 按照更小distance和mertic的优先级，将rib插入到rib nsm\_ptree\_node中
10. nsm\_rib\_process 调用casa\_rib\_select\_check ，nsm\_nexthop\_active\_update，nsm\_nexthop\_active\_ipv4\_vrf检查下一跳是否active，将nexthop->gate.ipv4在afi[AFI\_IP].rib[SAFI\_UNICAST]在递归查找，最后找到prefix为10.10.10.0的node，路由为直连（type==NEXTHOP\_TYPE\_IPV4\_IFINDEX ，active）。
11. select\_rib\_add\_sort (&select\_list\_head , rib) 把rib加入到select\_list\_head中，采用谁有更多的reminder nexthop,谁在前的顺序
12. casa\_rib\_ospf\_install\_nhop\_num\_calc 进行计算，目的是为了把同cost的最多剩余nexthop的rib进行安装
13. nsm\_rib\_process\_add 在这里被设置RIB\_FLAG\_SELECTED标志位
14. nsm\_rib\_install\_kernel 将这条路由安装到kernel去
15. nsm\_rib\_fib\_add 检测是否还有空间加这条路由
16. nsm\_fea\_ipv4\_add
17. pal\_kernel\_ipv4\_add 调用netlink\_route\_multipath 用netlink装载到kernel中
18. nl\_route\_info\_add\_ipv4 在这里组装netlink包，并使用msg\_enqueue将msg入队发送。
19. zebra\_remote\_route\_add发送给zebra remote
20. create\_info\_ipv4装载路由信息到zebra\_remote\_msg\_t msg
21. msg\_enqueue\_efd\_post 加入zebra发送队列，发送

补充OSPF

1. redistribute connected

### redistribute connected

本机运行redistribute connected

发送事件 ospfd->nsm->ospfd

redistribute connected 是由ospfd 发送消息给nsm

Ospfd

1. 通过cli发送NSM\_MSG\_REDISTRIBUTE\_SET

nsm

1. nsm\_server\_recv\_redistribute\_set nsm收到ospfd的消息
2. nsm\_redistribute 收到ospfd的消息，把type为IPI\_ROUTE\_CONNECT rib都通过4发送
3. nsm\_send\_ipv4\_add
4. nsm\_server\_send\_route\_ipv4
5. nsm\_server\_send\_message 发送给ospfd

Ospfd

1. ospf\_nsm\_recv\_route\_ipv4
2. ospf\_redist\_info\_add 收到nsm消息，并把redistribute发布出去。

在R2运行redistribute connected后

Nsm对于每个新的prefix，会跑多次OSPF路由安装的过程。

## ECMP

R1

Gige 5/0 ip 10.10.10.1

Gige 5/0 vlan 200 ip 20.20.20.1

router ospf 1

network 10.10.10.0/24 area 0

network 20.20.20.0/24 area 0

R2

Gige 5/0 ip 10.10.10.2

Gige 5/0 vlan 200 ip 20.20.20.2

router ospf 1

network 3.3.3.3/32 area 0

network 10.10.10.0/24 area 0

运行

network 20.20.20.0/24 area 0

1. nsm\_rib\_set\_server\_callback 注册callback nsm\_parse\_route\_ipv4 nsm\_server\_recv\_route\_ipv4
2. message\_server\_read 收到ospfd的消息，调用MESSAGE\_EVENT\_READ\_HEADER callback读取(nsm\_server\_read\_msg)
3. nsm\_server\_read\_msg 读取nsm\_msg\_header 获取vr\_id,vrf\_id,msg type（31 NSM\_MSG\_ROUTE\_IPV4）等信息，读出msg body，调用对应TYPE的parser callback。
4. nsm\_parse\_route\_ipv4 parse出nsm\_msg\_route\_ipv4中的flags(为1 ==NSM\_MSG\_ROUTE\_FLAG\_ADD),prefix\_len,prefix,type,distance,metric等值。然后调用callback
5. nsm\_server\_recv\_route\_ipv4 通过获取到的vr\_id找到nm(vrf)表，新建struct rib，将路由信息填入(type==IPI\_ROUTE\_OSPF,flags,distance,metric,uptime等等)。转化nexthop的信息为rib的格式。
6. nsm\_nexthop\_ipv4\_ifindex\_add 这里把nexthop信息保存下来（把type设置为NEXTHOP\_TYPE\_IPV4\_IFINDEX，保存gateway ip，ifindex）
7. nsm\_nexthop\_add将这个nexthop加入到struct rib的nexthop链表中，这个函数将会跑两次，因为这次OSPFD发送来的nsm\_msg会包含两个nexthop
8. nsm\_rib\_add 先在 vrf->afi[AFI\_IP].rib[SAFI\_UNICAST]用prefix，搜索是否已经有同个prefix的nsm\_ptree\_node已经被加入tree。由于3.3.3.3的prefix已经存在，所以这时会找到一个nsm\_ptree\_node。然后搜索node中是否有相同的rib，此时会搜索到一个same rib。
9. nsm\_rib\_delnode 对same rib进行删除,这里并没有free掉这个rib,只是把他从链表中移出
10. casa\_ribnode\_add\_sort 按照更小distance和mertic的优先级，将rib插入到rib nsm\_ptree\_node中
11. nsm\_rib\_process 会带一个same rib参数被调用，它会调用casa\_rib\_select\_check ，nsm\_nexthop\_active\_update，nsm\_nexthop\_active\_ipv4\_vrf检查下一跳是否active，将nexthop->gate.ipv4在afi[AFI\_IP].rib[SAFI\_UNICAST]在递归查找，最后找到prefix为10.10.10.0的node，路由为直连（type==NEXTHOP\_TYPE\_IPV4\_IFINDEX ，active）。
12. select\_rib\_add\_sort 把rib加入到select\_list\_head中，采用谁有更多的reminder nexthop,谁在前的顺序
13. nsm\_rib\_process\_delete 这里会将same rib从kernel中删除
14. nsm\_rib\_uninstall\_kernel 调用nsm\_fea\_ipv4\_delete，netlink 删除，zebra\_remote\_route\_del通知对端删除
15. casa\_rib\_ospf\_install\_nhop\_num\_calc 有可能其他rib安装了一定数量的nexthop，这个function进行计算得出一个 nexthop\_fib\_quota，这个值是这个rib可安装nexthop的数目。对于一个prefix nsm\_ptree\_node来说，最大的nexthop不能超过8个。原则是尽量不去改动installed fib。尽量给new select rib安装最大的nexthop数量，尽量给changed的fib安装最大的nexthop数量
16. nsm\_rib\_process\_add 在这里被设置RIB\_FLAG\_SELECTED标志位
17. nsm\_rib\_install\_kernel 将这条路由安装到kernel去
18. nsm\_rib\_fib\_add 检测是否还有空间加这条路由
19. nsm\_fea\_ipv4\_add
20. pal\_kernel\_ipv4\_add 调用netlink\_route\_multipath,里面会用安装两个路由，用netlink装载到kernel中
21. nl\_route\_info\_add\_ipv4 在这里组装netlink包，并使用msg\_enqueue将msg入队发送。
22. zebra\_remote\_route\_add发送给zebra remote
23. create\_info\_ipv4装载路由信息到zebra\_remote\_msg\_t msg
24. msg\_enqueue\_efd\_post 加入zebra发送队列，发送

### down 一个interface

R1

Gige 5/0 ip 10.10.10.1

Gige 5/0 vlan 200 ip 20.20.20.1

router ospf 1

network 10.10.10.0/24 area 0

network 20.20.20.0/24 area 0

R2

Gige 5/0 ip 10.10.10.2

Gige 5/0 vlan 200 ip 20.20.20.2

router ospf 1

network 3.3.3.3/32 area 0

network 10.10.10.0/24 area 0

network 20.20.20.0/24 area 0

Shutdown interface R1 gige 5/0 vlan 200

先删除本地路由，再收到OSPFD的消息，删除掉ECMP（preifx 3.3.3.3）两个nexthop的rib，然后再装一个nexthop的rib

1. kernel\_if\_link\_evnt\_read 收到kernel的link event，解析到RTM\_NEWLINK type。
2. netlink\_new\_link 解析msg，通过名字找到struct interface,对比kernel状态后得知if down
3. nsm\_if\_down 解析type为NSM\_IF\_TYPE\_L3
4. \_nsm\_l3\_if\_down 获取ifp的ip地址，从内核删除连接的路由
5. nsm\_connected\_down\_ipv4 目标通过ifp地址，准备删除NSM的rib
6. nsm\_rib\_delete\_connected\_ipv4 将prefix(20.20.20.0)，并将ifindex，type为IPI\_ROUTE\_CONNECT传入下面函数进行删除
7. nsm\_rib\_delete\_by\_ifindex 通过prefix 在vrf->afi[afi].rib[SAFI\_UNICAST]中找到20.20.20.0的nsm\_ptree\_node, 在node中通过ifindex找到rib，先标记它。
8. nsm\_rib\_delete 用nsm\_rib\_delnode断开rib在链表中的连接
9. nsm\_rib\_process nsm\_nexthop\_active\_update识别20.20.20.0 nexthop type为NEXTHOP\_TYPE\_IFINDEX，在vrf->ifv中查找到并检测到if正在running，nexthop检测为NEXTHOP\_FLAG\_ACTIVE
10. nsm\_rib\_process\_delete 调用nsm\_rib\_uninstall\_kernel 调用netlink把这条本地路由删除
11. nsm\_server\_recv\_route\_ipv4 收到OSPF新消息
12. nsm\_rib\_add
13. nsm\_rib\_process 删除掉这个OSPF的ECMP 3.3.3.3 via 10.10.10.0 20.20.20.0 加上OSPF 3.3.3.3 via 10.10.10.0

### 在ospf ECMP时配置多一个static

1. nsm\_static\_add
2. nsm\_static\_install 把新的 rib 排序后插入rib tree, 由于是static路由，它的distance比ospf小，所以在node中，新rib会在ospf前。
3. nsm\_rib\_process 先把ospf rib加入fib数组，记录下已经安装的路由，然后从前到后轮询tree node中的rib链，先访问的是static，再访问ospf。
4. casa\_rib\_select\_check 这里先将static加入select\_list\_head，表示最高优先级的路由，然后check ospf rib, 但由于select\_list\_head的type不是ospf，则代表已经有比高优先级的路由了，且不是ospf ecmp，所以这个ospf不会被check nexthop active，则它不会被设置RIB\_FLAG\_CANDIDATE。
5. nsm\_rib\_process\_delete 这里检查fib数组中，没有RIB\_FLAG\_CANDIDATE标志的rib，表示他们已经没被选中，则进行kernel route delete.
6. nsm\_rib\_process\_add 把static 路由加入kernel

## 链路状态变化（down、up）

R1

Gige 5/0 ip 10.10.10.1

Gige 5/0 vlan 200 ip 20.20.20.1

router ospf 1

network 10.10.10.0/24 area 0

network 20.20.20.0/24 area 0

R2

Gige 5/0 ip 10.10.10.2

Gige 5/0 vlan 200 ip 20.20.20.2

router ospf 1

network 3.3.3.3/32 area 0

network 10.10.10.0/24 area 0

network 20.20.20.0/24 area 0

在R1 运行 interface gige 5/0 vlan 200

shutdown

1. kernel\_if\_link\_evnt\_read 收到kernel的link event，解析到RTM\_NEWLINK type。
2. netlink\_new\_link 解析msg，通过名字找到struct interface,对比kernel状态后得知if down
3. nsm\_if\_down 解析type为NSM\_IF\_TYPE\_L3
4. \_nsm\_l3\_if\_down 获取ifp的ip地址，从内核删除连接的路由
5. nsm\_connected\_down\_ipv4 目标通过ifp地址，准备删除NSM的rib
6. nsm\_rib\_delete\_connected\_ipv4 将prefix(20.20.20.0)，并将ifindex，type为IPI\_ROUTE\_CONNECT传入下面函数进行删除
7. nsm\_rib\_delete\_by\_ifindex 通过prefix 在vrf->afi[afi].rib[SAFI\_UNICAST]中找到20.20.20.0的nsm\_ptree\_node, 在node中通过ifindex找到rib，先标记它。
8. nsm\_rib\_delete 用nsm\_rib\_delnode断开rib在链表中的连接
9. nsm\_rib\_process nsm\_nexthop\_active\_update识别20.20.20.0 nexthop type为NEXTHOP\_TYPE\_IFINDEX，在vrf->ifv中查找到并检测到if正在running，nexthop检测为NEXTHOP\_FLAG\_ACTIVE
10. nsm\_rib\_process\_delete 调用nsm\_rib\_uninstall\_kernel 调用netlink把这条本地路由删除
11. nsm\_server\_recv\_route\_ipv4 然后会收到OSPFD的新路由请求
12. nsm\_rib\_add
13. nsm\_rib\_process
14. nsm\_rib\_process\_delete 对原来的OSPF路由3.3.3.3 via 20.20.20.2 ，via 10.10.10.1进行删除
15. nsm\_rib\_process\_add 对3.3.3.3 via 10.10.10.1的OSPF路由进行添加

R1

Gige 5/0 ip 10.10.10.1

Gige 5/0 vlan 200 ip 20.20.20.1

router ospf 1

network 10.10.10.0/24 area 0

network 20.20.20.0/24 area 0

R2

Gige 5/0 ip 10.10.10.2

Gige 5/0 vlan 200 ip 20.20.20.2

router ospf 1

network 3.3.3.3/32 area 0

network 10.10.10.0/24 area 0

network 20.20.20.0/24 area 0

在R1 运行 interface gige 5/0 vlan 200

No shutdown

1. kernel\_if\_link\_evnt\_read 收到kernel的link event，解析到RTM\_NEWLINK type。
2. netlink\_new\_link 解析msg，通过名字找到struct interface,对比kernel状态后得知if down
3. nsm\_if\_up 解析type为NSM\_IF\_TYPE\_L3
4. \_nsm\_l3\_if\_up 获取ifp的ip地址，准备安装内核连接的路由
5. nsm\_rib\_add\_connected\_ipv4 将这个地址安装到nsm rib
6. nsm\_rib\_new 新建struct rib, 后续填入地址 nexthop
7. nsm\_rib\_add 查找same rib，由于新增，不会找到，然后将rib插入到nsm\_ptree\_node中
8. nsm\_rib\_process nexthop的type为NEXTHOP\_TYPE\_IFINDEX，在vrf->ifv查找到并检测状态为running，设置为NEXTHOP\_FLAG\_ACTIVE
9. nsm\_rib\_process\_add nsm\_rib\_install\_kernel 将这条路由安装到kernel。
10. nsm\_server\_recv\_route\_ipv4 然后会收到OSPFD的新路由请求
11. nsm\_rib\_add
12. nsm\_rib\_process
13. nsm\_rib\_process\_delete 对原来的OSPF路由3.3.3.3 via 10.10.10.1进行删除
14. nsm\_rib\_process\_add 对3.3.3.3 via 10.10.10.1 via 20.20.20.2 的OSPF路由进行添加

### **static路由的down up**

interface gige 5/0 vlan 300

ip address 30.30.30.1 255.255.255.0

3.3.3.3/32 [1/0] via 30.30.30.2, gige5/0 vlan300

Interface gige 5/0 vlan 300

Shutdown

No shutdown

一开始和前面一样，先删除IPI\_ROUTE\_CONNECT的路由，然后运行update timer

1. kernel\_if\_link\_evnt\_read 收到kernel的link event，解析到RTM\_NEWLINK type。
2. netlink\_new\_link 解析msg，通过名字找到struct interface,对比kernel状态后得知if down
3. nsm\_if\_down 解析type为NSM\_IF\_TYPE\_L3
4. \_nsm\_l3\_if\_down 获取ifp的ip地址，从内核删除连接的路由
5. nsm\_connected\_down\_ipv4 目标通过ifp地址，准备删除NSM的rib
6. nsm\_rib\_delete\_connected\_ipv4 将prefix(30.30.30.0)，并将ifindex，type为IPI\_ROUTE\_CONNECT传入下面函数进行删除
7. nsm\_rib\_delete\_by\_ifindex 通过prefix 在vrf->afi[afi].rib[SAFI\_UNICAST]中找到30.30.30.0的nsm\_ptree\_node, 在node中通过ifindex找到rib，先标记它。
8. nsm\_rib\_delete 用nsm\_rib\_delnode断开rib在链表中的连接
9. nsm\_rib\_process nsm\_nexthop\_active\_update识别30.30.30.0 nexthop type为NEXTHOP\_TYPE\_IFINDEX，在vrf->ifv中查找到并检测到if正在running，nexthop检测为NEXTHOP\_FLAG\_ACTIVE
10. nsm\_rib\_process\_delete 调用nsm\_rib\_uninstall\_kernel 调用netlink把这条本地路由删除
11. nsm\_rib\_update\_ipv4\_timer
12. nsm\_rib\_update\_vrf 检测到是 INTERFACE\_CHANGE 事件，会检测unicast tree中所有的node
13. nsm\_rib\_process 检测到static路由3.3.3.3时，检测nexthop active，nexthop type为NEXTHOP\_TYPE\_IFINDEX，not running，所以设置not active
14. nsm\_rib\_process\_delete 此static路由被删除

up也是通过update timer把static路由加回来

## Recursive

R1

Gige 5/0 ip 10.10.10.1

router ospf 1

network 10.10.10.0/24 area 0

R2

Gige 5/0 ip 10.10.10.2

router ospf 1

network 3.3.3.3/32 area 0

network 10.10.10.0/24 area 0

R1下

ip route 40.40.40.1/32 3.3.3.3

1. nsm\_static\_add
2. nsm\_static\_install
3. nsm\_nexthop\_active\_update
4. nsm\_nexthop\_active\_check
5. nsm\_nexthop\_active\_ipv4\_vrf 在这里和正常加直连static有所区别，在检查下一跳的时候，nexthop 类型为 NEXTHOP\_TYPE\_IPV4，此时会用gateway进行递归查找，找到IPI\_ROUTE\_OSPFF的rib,并且nexthop type为NEXTHOP\_TYPE\_IPV4\_IFINDEX，递归的找到下一跳，返回后设置nexthop为NEXTHOP\_FLAG\_ACTIVE
6. nsm\_rib\_process\_add
7. nsm\_nexthop\_active\_update 这里再次调用nsm\_nexthop\_active\_ipv4\_vrf，set=1，这次会把递归的下一跳信息保存到nexthop->rrn\_info，并且把查到的nsm\_ptree\_node 赋值给nexthop->rrn = rn，并设置nexthop为NEXTHOP\_FLAG\_RECURSIVE，nexthop->rrn 将会在安装时用到。
8. casa\_netlink\_route\_multipath 会检查rib flag是否是NEXTHOP\_FLAG\_RECURSIVE，是的话会安装其nexthop->rrn->info flags为RIB\_FLAG\_SELECTED的rib，然后再次调用casa\_netlink\_route\_multipath，casa\_netlink\_rhop\_exist\_in\_list检查找到的recursive rib的NEXTHOP\_FLAG\_FIB nexthop安装了没有。
9. nl\_route\_info\_add\_ipv4 prefix 40.40.40.1最后nexthop安装的其实是10.10.10.2

用timer update的原因是出于性能考虑，一次可以处理多个rib变化

### 配置static导致重新算

在上面的拓扑中，在R1下 ip route 3.3.3.3/32 10.10.10.2 static路由

1. nsm\_static\_add
2. nsm\_static\_install
3. nsm\_rib\_process 新增一条static路由，触发update timer
4. nsm\_rib\_update\_ipv4\_timer
5. nsm\_rib\_process
6. nsm\_nexthop\_active\_update 会先记录下40.40.40.1的 nh->rifindex
7. nsm\_nexthop\_active\_ipv4\_vrf 会递归查找到40.40.40.1的nexthop rrn与本次查询到的rn不一样，则把rifindex设置为0，则返回后，查询到rifindex不相等，则将rib设为RIB\_FLAG\_CHANGED
8. nsm\_rib\_process\_update 会检查rib是否设为RIB\_FLAG\_CHANGED，若是fib不是kernel类型的话代表nexthop的改动触发了路由的更新
9. nsm\_rib\_uninstall\_kernel 把原来rrn为OSPF的route先删除
10. nsm\_nexthop\_active\_update 更新40.40.40.1 nexthop的信息，这次set=1，会设置rrn
11. nsm\_rib\_install\_kernel 把新的recursive路由安装

## MPLS路由

### mpls code flow

R1：

lo 1.1.1.1

Gige 5/0 Vlan 200 ip 20.20.20.1

R2:

Lo 3.3.3.3

Gige 5/0 vlan 200 ip 20.20.20.2

Gige 5/0 vlan 300 ip 30.30.30.2

#### Router 加多一个 接口，看看 bng 怎么学习 ospf--> ldp 这个过程

R1上去加Gige 5/0 Vlan 300 ip 30.30.30.1

并且ospf 通告这个网段

1. nsm\_rib\_add 将新的路由的preifix 3.3.3.3 放到rib->vrf->nm 去查，看是否能查到ftn，结果有ftn 3.3.3.3，于是把rib设为RIB\_FLAG\_MPLS\_FWD
2. nsm\_rib\_process 安装OSPF路由3.3.3.3 via 20.20.20.2
3. nl\_route\_info\_add\_ipv4 查到info->ext\_flags是 RIB\_FLAG\_MPLS\_FWD，把info->ifindex = rib->vrf->leak\_ifindex(dp\_0\_253)，nexthop 地址清0。则kernel路由中看到下一跳不再是20.20.20.2而是

3.3.3.3 dev dp\_0\_253 proto 219 src 30.30.30.1 metric 20

3.3.3.3 dev dp\_0\_253 proto 219 src 20.20.20.1 metric 20

1. nsm\_server\_recv\_ftn\_ipv4 解析出ftn\_add\_data fad，检查其flags为NSM\_MSG\_FTN\_ADD。先用fad->oif\_ix去查找ifp，检查到ifp running，然后调用nsm\_mpls\_ftn\_add\_msg\_process
2. nsm\_mpls\_ftn\_add\_msg\_process 调用nsm\_gmpls\_set\_ftn\_gen\_data把data 安装进gen\_data
3. nsm\_gmpls\_ftn\_add\_msg\_process 检查协议族和下一跳地址的合法性，nsm\_gmpls\_ftn\_table\_lookup去寻找一个ftn table(NSM\_MPLS->ftn\_ix\_table4)
4. gmpls\_ftn\_add\_process
5. gmpls\_ftn\_new 新建一个ftn\_entry，并将之前的data保存到这个entry。
6. gmpls\_ftn\_row\_create 创建新的nhlfe且赋值，并把它加入NHLFE\_TABLE4
7. gmpls\_xc\_new 创建新的xc entry
8. gmpls\_xc\_nhlfe\_link 把nhlfe和xc entry link起来
9. gmpls\_ftn\_add
10. gmpls\_ftn\_entry\_add\_list
11. gmpls\_ftn\_entry\_add\_list\_primary 把ftn entry加入ftn\_list，并设置为FTN\_ENTRY\_FLAG\_SELECTED
12. gmpls\_ftn\_entry\_select\_process 判断到ftn->flags是FTN\_ENTRY\_FLAG\_SELECTED，要把它加到kernel
13. nsm\_gmpls\_ftn\_add\_check 检查，并删除已安装过的ftn
14. nsm\_mpls\_ftn4\_add\_to\_fwd 准备安装fec prefix为3.3.3.3的ftn，并解析出tunnel\_nhop，tunnel\_label，tunnel\_if\_ident等信息
15. ipi\_mpls\_ftn4\_entry\_add
16. pal\_ipi\_mpls\_ftn\_entry\_add 检查安装的prefix的rib是否已经安装过，由于OSPF阶段已经安装，跳过。
17. zebra\_remote\_route\_ftn4\_add

只有一个ospf的情况

R1：

lo 1.1.1.1

Gige 5/0 Vlan 200 ip 20.20.20.1

R2:

lo 3.3.3.3

Gige 5/0 vlan 200 ip 20.20.20.2

在R2运行ospf network 20.20.20.0/24 area 0

1. nsm\_server\_recv\_route\_ipv4
2. nsm\_rib\_add
3. nsm\_rib\_process 1 2 3是收到并安装的OSPF路由，3.3.3.3 via 20.20.20.2
4. nsm\_server\_recv\_ftn\_ipv4 收到一个新的FTN
5. gmpls\_ftn\_entry\_add\_list\_primary 把这个ftn\_entry加入到ftn\_list
6. ipi\_mpls\_ftn4\_entry\_add
7. pal\_ipi\_mpls\_ftn\_entry\_add 拿这个fec的prefix到p\_vrf->afi[AFI\_IP].rib[SAFI\_UNICAST]查，发现已经有这个路由rib，是type为OSPF的，且判断到这个rib不是RIB\_FLAG\_MPLS\_FWD。
8. nsm\_fea\_ipv4\_del 先把原来的rib在kernel中删除了
9. 给rib设置RIB\_FLAG\_MPLS\_FWD
10. nsm\_fea\_ipv4\_add 将rib安装回来
11. nl\_route\_info\_add\_ipv4 检查到RIB\_FLAG\_MPLS\_FWD 把nexthop addr 设置为0，设置leak\_ifindex（dp\_0\_253)设置prefsrc，然后进入发送队列，发送此路由到kernel

#### Router clear ldp session，看看 bng 怎么单独学习 ldp （换成在interface 下disable-ldp/enable-ldp ipv4）

R1：

lo 1.1.1.1

Gige 5/0 Vlan 200 ip 20.20.20.1

R2:

lo 3.3.3.3

Gige 5/0 vlan 200 ip 20.20.20.2

##### enable-ldp ipv4

在R1 gige5/0 vlan 200 下enable-ldp ipv4

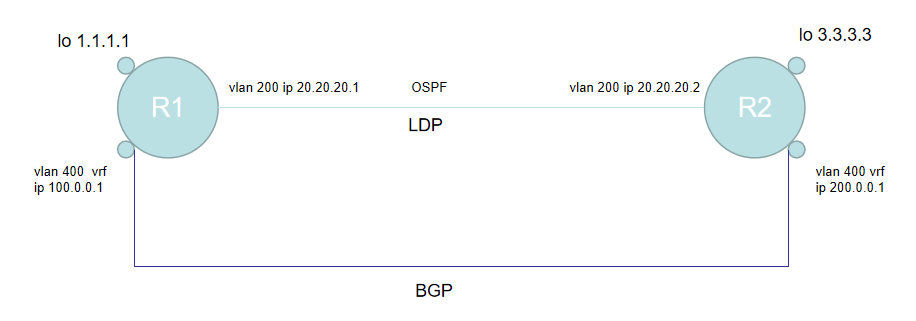
1. nsm\_server\_recv\_ftn\_ipv4
2. nsm\_mpls\_ftn\_add\_msg\_process
3. nsm\_gmpls\_ftn\_add\_msg\_process
4. gmpls\_ftn\_add\_process
5. gmpls\_ftn\_entry\_select\_process
6. nsm\_mpls\_ftn4\_add\_to\_fwd
7. ipi\_mpls\_ftn4\_entry\_add
8. pal\_ipi\_mpls\_ftn\_entry\_add 把fec\_addr(prefix)放到p\_vrf->afi[AFI\_IP].rib[SAFI\_UNICAST] 树中查找，找到rib，这个rib就是fec的下一跳，检测此RIB是否已经设置RIB\_FLAG\_MPLS\_FWD，若还没有，则先删除kernel中安装的，再设置RIB\_FLAG\_MPLS\_FWD，再安装。
9. nsm\_fea\_ipv4\_add
10. netlink\_route\_multipath
11. nl\_route\_info\_add\_ipv4 检查到RIB\_FLAG\_MPLS\_FWD 把nexthop addr 设置为0，设置leak\_ifindex（dp\_0\_253)设置prefsrc，然后进入发送队列，发送此路由到kernel

##### disable-ldp ipv4

在R1 gige5/0 vlan 200 下disable-ldp ipv4

1. nsm\_server\_recv\_ftn\_ipv4
2. nsm\_mpls\_ftn\_del\_msg\_process
3. nsm\_gmpls\_ftn\_del\_msg\_process
4. gmpls\_ftn\_entry\_remove\_list\_primary 把fec preifix为3.3.3.3的ftn，将它从ftn\_list中移除
5. gmpls\_ftn\_del\_from\_fwd 准备删除掉FTN的路由
6. ipi\_mpls\_ftn4\_entry\_del nsm\_rib\_lookup先查ftn的路由是否已经有了，检查到rib已经安装了，于是检查rib是不是被设置为RIB\_FLAG\_MPLS\_FWD。
7. nsm\_fea\_ipv4\_delete 检查到rib ext\_flags RIB\_FLAG\_MPLS\_FWD,先删除kernel中的路由
8. 再去除rib的RIB\_FLAG\_MPLS\_FWD
9. nsm\_fea\_ipv4\_add 将rib作为OSPF type再加回去

### Vpn code flow:



#### bgp 加直连，看看 bng 怎么学习

对端下redistribute connected

1. nsm\_server\_recv\_ftn\_ipv4
2. nsm\_mpls\_ftn\_add\_msg\_process
3. gmpls\_ftn\_entry\_add\_list\_primary 在这里把这个ftn保存到ftn tree中
4. nsm\_mpls\_ftn4\_add\_to\_fwd
5. pal\_ipi\_mpls\_ftn\_entry\_add 在此处打算将ftn的路由设置到路由表，但是由于bgp的rib还未安装，所以这里没有安装进kernel
6. nsm\_server\_recv\_route\_ipv4 收到这个type IPI\_ROUTE\_BGP，subtype IPI\_ROUTE\_BGP\_MPLS 就会去找vrf的gw\_vrf，这个gw\_vrf在后面查找递归的时候会用gw\_vrf==0 vrf==1
7. nsm\_rib\_add 查询这个路由的prefix是否在ftn的tree中已经有了，有了就设置RIB\_FLAG\_MPLS\_FWD,由于前面ftn加了，所以这里会设标志位
8. nsm\_rib\_process
9. nsm\_rib\_process\_add
10. nsm\_nexthop\_active\_ipv4\_vrf 这里会查200.0.0.0 via 3.3.3.3的nexthop,是递归，并且找到从vrf(0)

找到rrn为20.20.20.2,后续进行安装

#### clear ldp/clear ospf/ shutdown 5/0 vlan 100 看看分别是怎么删除 vpn 路由



clear ldp session \*

1. nsm\_server\_recv\_ftn\_ipv4
2. gmpls\_ftn\_entry\_remove\_list
3. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉fec 200.0.0.0节点ftn\_entry
4. gmpls\_ftn\_del\_from\_fwd
5. pal\_ipi\_mpls\_ftn\_entry\_del 把preifx 200.0.0.0放到vrftest的p\_vrf->afi[AFI\_IP].rib[SAFI\_UNICAST]中查找，因为可能这个prefix的nexthop是个递归，找到后用nsm\_rib\_process 进行rib的update，保证后面的删除动作或者添加可以正常。然后nsm\_rib\_lookup来检查看rib有没有RIB\_FLAG\_SELECTED，递归路由是否存在。如果没有，则会退出。路由正常存在，则检查其RIB\_FLAG\_MPLS\_FWD,然后用nsm\_fea\_ipv4\_delete 删除这个路由，去掉RIB\_FLAG\_MPLS\_FWD，nsm\_fea\_ipv4\_add把路由加回去，这时回到BGP。
6. nsm\_server\_recv\_ftn\_ipv4 删除prefix 3.3.3.3的路由(和上面步骤类似)
7. nsm\_server\_recv\_ftn\_ipv4
8. gmpls\_ftn\_entry\_add\_list\_primary 把fec(3.3.3.3) ftn entry加入ftn\_list，并设置为FTN\_ENTRY\_FLAG\_SELECTED
9. pal\_ipi\_mpls\_ftn\_entry\_add 给rib添加RIB\_FLAG\_MPLS\_FWD
10. nsm\_server\_recv\_ftn\_ipv4
11. gmpls\_ftn\_entry\_add\_list\_primary 把fec(200.0.0.0) ftn entry加入ftn\_list，并设置为FTN\_ENTRY\_FLAG\_SELECTED
12. pal\_ipi\_mpls\_ftn\_entry\_add 给rib添加RIB\_FLAG\_MPLS\_FWD

no network 20.20.20.0/24 area 0

clear ospf

1. nsm\_server\_recv\_ftn\_ipv4
2. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉这个节点ftn\_entry(fec 200.0.0.0)
3. pal\_ipi\_mpls\_ftn\_entry\_del 删除掉这个路由的RIB\_FLAG\_MPLS\_FWD，然后再加回去
4. nsm\_server\_recv\_ftn\_ipv4
5. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉这个节点ftn\_entry(fec 3.3.3.3)
6. pal\_ipi\_mpls\_ftn\_entry\_del 删除掉这个路由的RIB\_FLAG\_MPLS\_FWD，然后再加回去
7. nsm\_server\_recv\_route\_ipv4
8. nsm\_rib\_delete
9. nsm\_rib\_process 删除OSPF 3.3.3.3 via 20.20.20.2 的路由
10. nsm\_server\_recv\_route\_ipv4
11. nsm\_rib\_delete
12. nsm\_rib\_process 删除BGP 200.0.0.0 via 3.3.3.3 的路由

Shutdown

1. kernel\_if\_link\_evnt\_read 收到kernel netlink
2. \_nsm\_l3\_if\_down
3. nsm\_mpls\_rib\_if\_down\_process
4. nsm\_mpls\_ftn\_if\_down\_process 会先处理这个ifp的 ftn process
5. kernel\_if\_link\_evnt\_read 删除IPI\_ROUTE\_CONNECT路由事件
6. nsm\_connected\_down\_ipv4
7. nsm\_rib\_process\_delete 删除这个IPI\_ROUTE\_CONNECT路由 20.20.20.0
8. nsm\_server\_recv\_route\_ipv4
9. nsm\_rib\_delete
10. nsm\_rib\_process 删除OSPF 3.3.3.3 via 20.20.20.2 的路由
11. nsm\_server\_recv\_route\_ipv4
12. nsm\_rib\_delete
13. nsm\_rib\_process 删除BGP 200.0.0.0 via 3.3.3.3 的路由
14. nsm\_server\_recv\_ftn\_ipv4
15. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉这个节点ftn\_entry(fec 200.0.0.0)
16. pal\_ipi\_mpls\_ftn\_entry\_del 想要删除掉200.0.0.0 via 20.20.20.2 rib的RIB\_FLAG\_MPLS\_FWD，然后再加回去，但由于这个路由之前被删掉了，这里不动作
17. nsm\_server\_recv\_ftn\_ipv4
18. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉这个节点ftn\_entry(fec 3.3.3.3)
19. pal\_ipi\_mpls\_ftn\_entry\_del 想要删除掉3.3.3.3 via 20.20.20.2 rib的RIB\_FLAG\_MPLS\_FWD，然后再加回去，但由于这个路由之前被删掉了，这里不动作

clear bgp

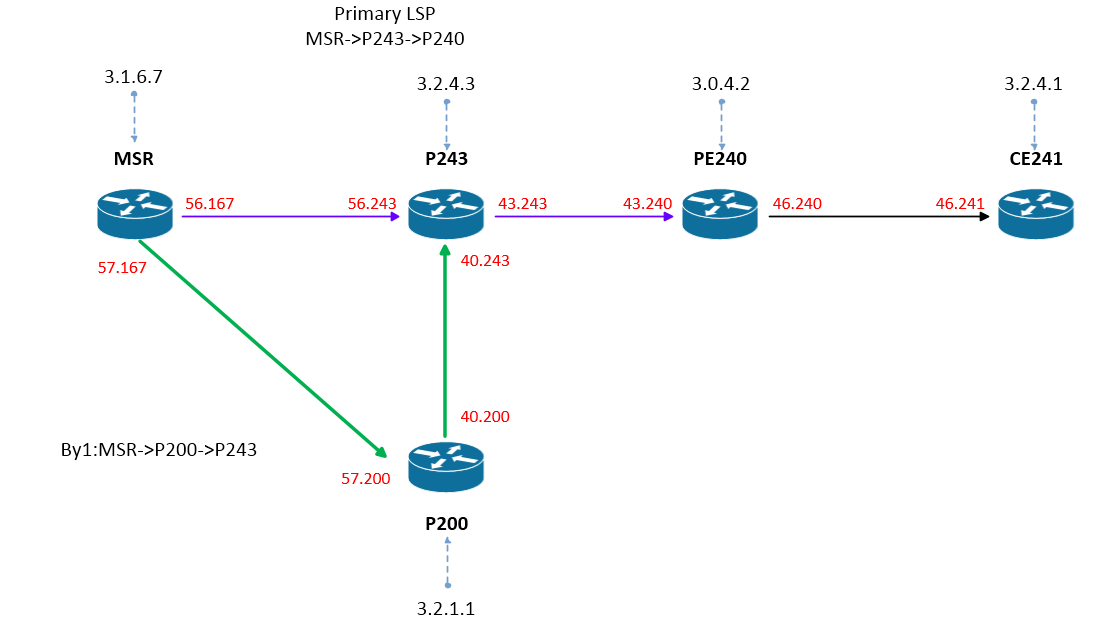
1. nsm\_server\_recv\_ftn\_ipv4
2. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉这个节点ftn\_entry(fec 200.0.0.0)
3. pal\_ipi\_mpls\_ftn\_entry\_del 删除掉这个路由的RIB\_FLAG\_MPLS\_FWD，然后再加回去（BGP）
4. nsm\_fea\_ipv4\_delete
5. UNSET\_FLAG(rib->ext\_flags, RIB\_FLAG\_MPLS\_FWD)
6. nsm\_fea\_ipv4\_add BGP
7. nsm\_server\_recv\_route\_ipv4 收到BGP的路由消息
8. nsm\_rib\_process
9. nsm\_rib\_process\_delete 对这个BGP的路由进行删除
10. nsm\_fea\_ipv4\_delete

Ecmp

加一条ospf ecmp,然后再clear ldp session

1. nsm\_server\_recv\_ftn\_ipv4
2. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉这个节点ftn\_entry(fec 3.3.3.3 ftn\_ix =1)
3. pal\_ipi\_mpls\_ftn\_entry\_del 要将prefix为 3.3.3.3的RIB\_FLAG\_MPLS\_FWD fib删除，里面有两个nexthop 20.20.20.2 30.30.30.2 都会删除。然后去掉RIB\_FLAG\_MPLS\_FWD再将这两个路由加回来，这时回到OSPF
4. nsm\_server\_recv\_ftn\_ipv4
5. gmpls\_ftn\_entry\_remove\_list\_primary 在ftn\_list中删除掉这个节点ftn\_entry(fec 3.3.3.3 ftn\_ix =2)
6. gmpls\_dep\_ftn\_update\_process 检查到fec 3.3.3.3 ftn\_ix =2是3.3.3.3 ptree\_node 中最后一个ftn了，则会对它lsp\_dep\_confirm 中的prefix(200.0.0.0)进行先删除(lsp\_dep\_confirm 保存了需要依赖这个lsp 3.3.3.3 才能工作的vrf)
7. gmpls\_ftn\_del\_from\_fwd
8. pal\_ipi\_mpls\_ftn\_entry\_del 要将prefix为 200.0.0.0 的RIB\_FLAG\_MPLS\_FWD fib删除，里面有两个nexthop 20.20.20.2 30.30.30.2 都会删除。然后去掉RIB\_FLAG\_MPLS\_FWD再将这两个路由加回来，这时回到BGP
9. gmpls\_ftn\_del\_from\_fwd 再继续删除 fec 3.3.3.3 ftn\_ix=2
10. pal\_ipi\_mpls\_ftn\_entry\_del 由于之前3已经把rib中的RIB\_FLAG\_MPLS\_FWD去掉，这里不会动作

### MPLS RSVP TE



#### tunnel 起来的时候，nsm 的流程

rsvp-trunk T1 ipv4

下

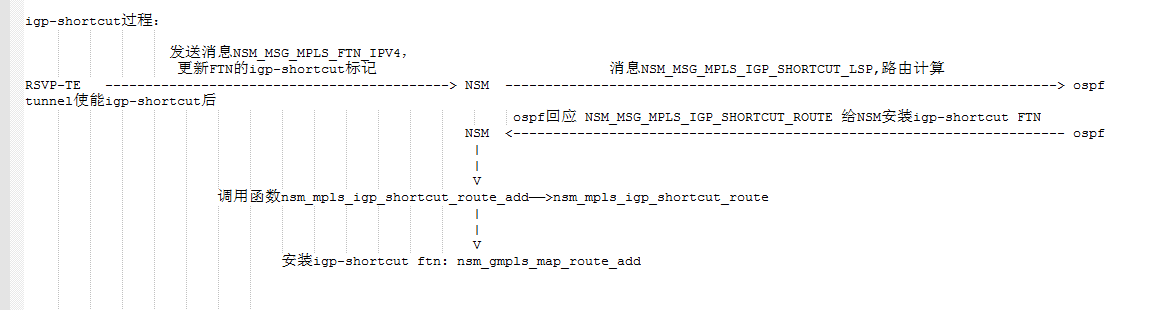
to 3.0.4.2

1. nsm\_server\_recv\_ftn\_ipv4 收到添加ftn的消息
2. gmpls\_ftn\_add\_process 创建新的ftn\_entry，设置ftn\_entry的值，保存tun\_id，pri\_fec\_prefix
3. gmpls\_ftn\_row\_create ftn\_entry建立nhlfe和xc\_entry，并配置他们的值
4. gmpls\_ftn\_add
5. gmpls\_ftn\_entry\_add\_list\_primary 将ftn\_entry(fec 3.0.4.2) 加入ftn\_list
6. gmpls\_ftn\_entry\_select\_process 处理被选中的ftn, 安装路由
7. nsm\_mpls\_ftn4\_add\_to\_fwd 将nhlfe的nexthop赋值给tunnel\_nhop, 由于ftn->owner.owner == MPLS\_RSVP，还会给mpls\_owner\_fwd赋值trunk\_id，lsp\_id，ingr，egr
8. nsm\_mpls\_ftn\_get\_frr\_entry 将trunk\_id，lsp\_id，ingr，egr 赋值给 te(zebra\_te\_tunnel\_info\_t)
9. pal\_ipi\_mpls\_ftn\_entry\_add 调用zebra\_remote\_route\_ftn4\_add(fec\_addr, \*prefixlen, nexthop\_addr, (vrf==GLOBAL\_TABLE?GLOBAL\_FTN\_ID:vrf), \*label\_id\_out, ftn?CHECK\_FLAG(ftn->flags, FTN\_ENTRY\_FLAG\_VPN\_DIRECT):0, &te); 把刚记录的tunnel\_nhop 发给DP
10. nsm\_fea\_ipv4\_delete(&p, rib); 删除原来的rib（prefix3.1.6.7 nexthop 56.234 OSPF）
11. SET\_FLAG(rib->ext\_flags, RIB\_FLAG\_MPLS\_FWD); 设置mpls flag
12. nsm\_fea\_ipv4\_add(&p, rib); 重新安装 rib （prefix3.1.6.7 nexthop 56.234 OSPF）

#### 一条路由引入到 te 之后 nsm 的流程 （IGP shortcut + map route）

接上面下 map-route 3.2.4.1/32，把通往3.2.4.1的包指向tunnel

1. nsm\_server\_recv\_ftn\_ipv4 收到添加ftn的消息
2. gmpls\_ftn\_add\_process 创建新的ftn\_entry，设置ftn\_entry的值，保存tun\_id，pri\_fec\_prefix
3. gmpls\_ftn\_row\_create ftn\_entry建立nhlfe和xc\_entry，并配置他们的值
4. nsm\_mpls\_ftn\_add\_msg\_process
5. nsm\_gmpls\_ftn\_add\_msg\_process
6. gmpls\_ftn\_add\_process
7. gmpls\_ftn\_entry\_add\_list\_primary 将ftn\_entry(fec 3.2.4.1 nhlfe 192.74.56.243) 加入ftn\_list(ftn\_list是ft->m\_table中以3.2.4.1为key的ptree\_node中的pn->info(类似rib list))
8. gmpls\_ftn\_entry\_select\_process 这个ftn的ftn\_type MPLS\_FTN\_RSVP\_MAPPED 得出dep\_type = CONFIRM\_RSVP\_MAPPED\_ROUTE，flag FTN\_ENTRY\_FLAG\_DEPENDENT。由于flags为FTN\_ENTRY\_FLAG\_DEPENDENT，所以要获取ftn（fec 3.2.4.1）所依赖的ftn去加confirm list，所以取 pri\_fec\_prefix(primary\_lsp fec 3.0.4.2)到gmpls\_lsp\_dep\_add
9. gmpls\_lsp\_dep\_add 通过primary lsp fec 查找到ptree\_node，然后用mpls\_confirm\_list\_add将（fec 3.2.4.1）加入到3.0.4.2的comfirm list
10. gmpls\_confirm\_data\_add\_to\_fwd 安装fec 3.2.4.1的路由
11. nsm\_mpls\_ftn4\_add\_to\_fwd
12. pal\_ipi\_mpls\_ftn\_entry\_add
13. zebra\_remote\_route\_ftn4\_add 将prefix 3.2.4.1 下一跳 pri\_fec\_prefix 3.0.4.2发给DP
14. nl\_route\_info\_add\_ipv4 安装kernel路由

Igp-shortcut

Ftn加入ftn list的时候，会对比

## Vxlan

### Configuration

vxlan

udp-port 4789

vtep-ip-global loopback 1

!

interface tunnel 1

tunnel name tty1

tunnel mode vxlan

tunnel source 45.45.45.1

tunnel destination 30.30.30.1

no shutdown

vxlan id 16777215

vni-name vni1

tunnel tty1

static-entry host-mac 0000.0000.1234 remote-vtep-ip 12.12.12.1

interface gige 5/0 vlan 100

no shutdown

map vxlan 16777215

### flow

vxlan

udp-port 4789

vtep-ip-global loopback 1

1. vxlan\_proc\_vxlan\_cmd 调用vxlan\_add\_all\_inst\_bcm，检查instance list然后安装，由于是刚开始配置，还未有instance
2. vxlan\_proc\_udp\_port\_cmd 将设置的udp 4789 作为VXLAN的通信port，但由于instance未建立，所以没有后续动作，如果已经有instance，那么将会把安装的instance删除，并把新的port设置后，重新安装instance
3. vxlan\_proc\_vtep\_ip\_global\_cmd 获取设置的loopback的地址作为vxlan的全局src ip，若没有设置src ip则将会用这个IP作为tunnel src ip

interface tunnel 1

tunnel name tty1

tunnel mode vxlan

tunnel source 45.45.45.1

tunnel destination 30.30.30.1

no shutdown

nsm\_interface.c

1. 创建tunnel的信息
2. vxlan\_proc\_tnl\_name 设置tunnel 的名字
3. vxlan\_proc\_tnl\_mode 设置tunnel 的模式为vxlan
4. vxlan\_proc\_tnl\_src 设置 src 地址
5. vxlan\_proc\_tnl\_dest 设置dest 地址
6. 启动tunnel

vxlan id 16777215

vni-name vni1

tunnel tty1

static-entry host-mac 0000.0000.1234 remote-vtep-ip 12.12.12.1

1. vxlan\_proc\_vxlan\_id\_cmd 新建vxlan instance 16777215，创建instance的vxlan\_inst\_t 节点，并将其加入instance 链表，然后nsm\_send\_vxlan\_inst\_to\_ffemgr(ZEBRA\_VXLAN\_INSTANCE\_ADD, vnid); 添加vxlan instance到ffe。
2. process\_vxlan\_request 将instance加入vxlan\_info\_instance\_tree（用vnid作key），然后发送至ffe
3. vxlan\_proc\_vni\_name\_cmd 给vxlan id 16777215设置vni 名字
4. vxlan\_proc\_bind\_tunnel\_name\_cmd 根据tunnel name 找到tunnel，然后检查instance的tunnel table看有没有绑定过同个tunnel。然后取出tunnel的src ip, dst ip,udp port,vnid 发送给（ZEBRA\_VXLAN\_INST\_TUNNEL\_ADD）ffe。
5. process\_vxlan\_request 将绑定以vnid和dst ip做key加入vxlan\_info\_tunnel\_tree，然后发给ffe
6. vxlan\_proc\_static\_mac\_entry\_cmd 配置远端的vtep 后面的host mac和ip使用这个instance，会进行相同mac配置的替换(目前看此处有bug，mac table赋值不对)，调用vxlan\_proc\_one\_mac\_bcm 类型ZEBRA\_VXLAN\_INST\_STATIC\_MAC\_ADD给ffe
7. process\_vxlan\_request 使用vnid mac dstip作为key保存到 vxlan\_info\_mac\_tree，然后发给ffe

interface gige 5/0 vlan 100

no shutdown

map vxlan 16777215

1. vxlan\_proc\_map\_vxlan\_cmd 获取ifname及vlan号，在map table中查找重复。然后把vxlan id,ifname,vlan发送给ffe (vxlan\_proc\_one\_map\_bcm, ZEBRA\_VXLAN\_INST\_VLAN\_MAPPING\_ADD)
2. process\_vxlan\_request 以vxlan id,ifname,vlan作为key 保存到vxlan\_info\_vlan\_tree，然后发送到ffe。

### 总结vxlan建立

1. 设置vxlan 全局参数
2. 建立tunnel，设定local和remote ip
3. 要将流量引入tunnel（什么样的流量要走tunnel？Vxlan id X的流量。那什么流量是属于vxlan id X？Host 为xxx的流量），且tunnel是为某个vxlan id服务，那么建立vxlan id，并绑定tunnel。然后设置static host的规则，则流量引入完成
4. 要为这个服务指定一个实际的接口，map vxlan到某个interface

## COA/PD

1. nsm\_server\_read\_msg
2. nsm\_parse\_sub\_route\_msg
3. nsm\_server\_recv\_sub\_route
4. nsm\_sub\_route\_coa\_pd
5. nsm\_rib\_add
6. nsm\_rib\_process

## POOL ROUTE

CPE GW

1. netlink\_newdel\_addr 从netlink读取到新的地址
2. netlink\_addr\_update 解析出地址等信息，解析出命令为RTM\_NEWADDR
3. nsm\_connected\_add\_ipv4 检查地址是否有重复，若有的话需要把kernel的地址移除，检测要添加的ifp是否running
4. nsm\_connected\_up\_ipv4
5. nsm\_rib\_add\_connected\_ipv4 添加这个路由rib->type为IPI\_ROUTE\_CONNECT
6. nsm\_rib\_add
7. casa\_ribnode\_add\_sort 路由进行按distance,metric排序
8. nsm\_rib\_process

触发上面的流程是因为dhcp给cpe-gw interface配置了ip，所以配置直连路由

Sub pool route

1. nsm\_parse\_sub\_route\_msg 收到sub route的消息，调用casa\_ipi\_nsm\_message\_decode\_sub\_route\_msg 解析出msg中的消息
2. nsm\_server\_recv\_sub\_route 解析sub\_type为IPI\_ROUTE\_BNG\_CPE\_PREFIX，构造prefix 等信息
3. nsm\_rib\_add rib->type为IPI\_ROUTE\_LTE
4. nsm\_rib\_process