上行\_MN端：

初始化（2个端口）:

1. 配置coa->发bu->接ba（两个端口）
2. 隧道1条：

ipip隧道

1. rule:

886: from 202.157.11.39 lookup 249

887: from 202.187.11.38 lookup 248

888: from all lookup 250

1. route table:

248路由表

default via 202.187.0.254 dev eth0

249路由表

default via 202.157.0.254 dev ath0

250路由表

default via tnl1

流程：

1. send
2. 系统默认调用 路由查询（其实没用）, 得到路由表250（tun1），
3. 调用xfrm lookup函数（配置启动xfrm后就能调用该函数），对数据包查询xfrm policy，因为一开始数据包没有源地址，数据包会获取tun1的设备的索引地址，查询成功后 template和state的比对等操作。生成一个dst链表。
4. 如果发生流切换，更新tmpl
5. 对数据包添加包头。包头里放的是索引地址（源地址），目的地址
6. 发包。

无NAT情况

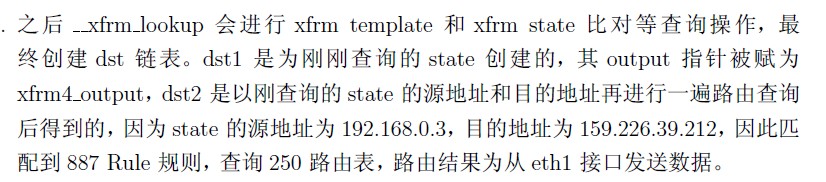
1. 可以仿照有NAT的情况，只需要把rule与route table进行设定，让源地址为COA2的包不直接发送，而是从tnl2进行发送

可行性：因为对于MN而言，在建立隧道之前，已经确认了自己是否处于NAT中，所以，可以针对有无NAT，对rule和route table进行特定的设置

存在的问题：中间的xfrm的查询等操作除了将源地址改为COA2与重新路由之外，其他xfrm的很多处理是无用的，看起来有一些累赘与多余

优点：与有NAT情况在流程上是对应起来的，不用作特殊处理，而且可以代码复用

参考：xfrm查询完后会再次进行路由查询



参数：

1. policy:

sel src: HOA dst: CNA dport proto(optional)

tmpl: src: COA1 dst: HA mode: XFRM\_MODE\_TUNNEL

更新后 src: COA2 dst: HA mode: XFRM\_MODE\_TUNNEL

1. state1: src:COA1 dst: HA

state2: src: COA2 dst: HA

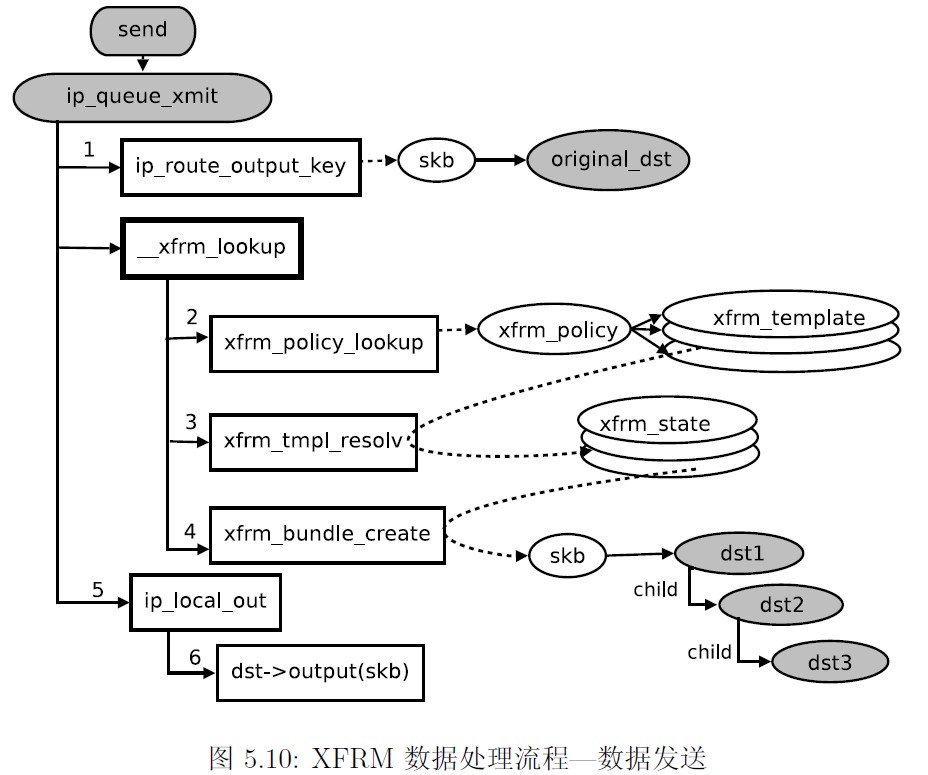
3. HA:10.21.5.74

FA:10.21.5.81

CN:10.21.5.144

4. ip地址配置（in the "cngictrl.h”）

**char** staticipaddr[IDNUMS][80]**=**{"192.168.0.2/24","192.168.1.2/24"};



Q:

/\* For the policy \*/

xfrm\_user\_tmpl

/\* For the state \*/

xfrm\_encap\_tmpl

<http://mkl-note.blogspot.com/2011/12/ipsec-related-resources.html>

//m：ld 17：51 13.5.2

思路：

1. 通过policy的sel匹配到流
2. 通过tmpl找到state
3. state的sel与flow进行匹配

HA端：

和mn类似