# LVS FULLNAT概要设计

By 吴佳明\_普空

## 1 背景

LVS当前应用主要采用DR和NAT模式，但这2种模式要求RealServer和LVS在同一个vlan中，导致部署成本过高；TUNNEL模式虽然可以跨vlan，但RealServer上需要部署ipip模块等，网络拓扑上需要连通外网，较复杂，不易运维。

为了解决上述问题，我们在LVS上添加了一种新的转发模式：FULLNAT，该模式和NAT模式的区别是：Packet IN时，除了做DNAT，还做SNAT（用户ip->内网ip），从而实现LVS-RealServer间可以跨vlan通讯，RealServer只需要连接到内网；

## 2目标

FULLNAT将作为一种新工作模式（同DR/NAT/TUNNEL），实现如下功能：

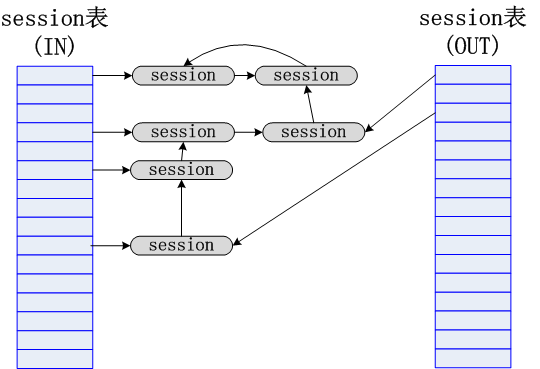
1. Packet IN时，目标ip更换为realserver ip，源ip更换为 内网local ip；
2. Packet OUT时，目标ip更换为client ip，源ip更换为vip；

注：Local ip 为一组内网ip地址；

性能要求，和NAT比，正常转发性能下降<10%；

## 3总体思路

1. 每个session维护9元组，caddr/vaddr/laddr/daddr，增加了local address;
2. 逻辑上采用2个session表：



创建in\_idx和out\_idx（struct ip\_vs\_conn\_idx）结构体变量，2个index指向同一session；

1. OUT2IN：in\_idx结构体变量，hash key为caddr/vaddr；
2. IN2OUT：out\_idx结构体变量，hash key为daddr/laddr；

## 4关键技术点

### 4.1 session管理-ip\_vs\_conn.c

FULL NAT的重点是session管理，其设计思路 参见 第3章-总体思路；

1. DR/NAT/TUNNEL下，laddr设置为caddr，以便查找session/nat转换时，一致处理；
2. FULL-NAT下，LOCAL\_IN HOOK点，找到session后，如何确认是IN2OUT 还是 OUT2IN的包？查找session时，输出dir参数，dir取自struct ip\_vs\_conn\_idx中的flag；
3. 1个session在2个hash桶中，查找session时，是否需要计算2个hash key？不需要，在查找sesson时，都用s\_addr/d\_addr hash；因为创建session已经规避了该问题。

### fullnat转换-ip\_vs\_xmit.c

1. 新增函数ip\_vs\_fnat\_xmit用于fullnat转换；
2. 新增函数tcp\_fnat\_in\_handler– 实现OUT2IN的fullnat
3. 新增函数tcp\_fnat\_out\_handler– 实现IN2OUT的fullnat

### TOA (address of tcp option)-插入client ip

为了保证应用透明性，通过tcp option传递client ip给RealServer；

在函数tcp\_fnat\_in\_handler中调用插入client ip的功能，因为

1. 该功能只有fullnat才会用
2. 和tcp协议相关

新增函数：tcp\_opt\_add\_toa，用于实现toa功能；

函数tcp\_opt\_insert\_cip实现可参考函数tcp\_options\_write，主要流程：

1. 如果skb数据长度 超过 设定的MSS，则异常报错；
2. 复制skb，增加长度；
3. Skb数据往后移动，tcp头部之后，空出TOA空间；
4. 插入CIP，包括 OPT CODE+LEN+DATA
5. 调整tcp头部长度参数，调整SKB参数；

### TCP OPT处理

参照函数tcp\_parse\_options；

#### 4.4.1过滤TIMESTAMP

FULLNAT方式下，如果timestamp开启，同时RealServer上开启tcp\_tw\_recycle，则对于NAT网关出来的用户，可能会出现服务端拒连接的问题；同时，采用local address会加剧该问题；

解决方法：过滤掉timestamp，即过滤掉Client->RS SYN包中的timestamp；

新增函数tcp\_opt\_remove\_timestamp，在函数tcp\_fnat\_in\_handle中被调用。

注意点：需重新计算校验和；

#### 4.4.2调整MSS

FULLNAT模式下，需要插入client ip到tcp option中，为了保证数据包大小不超过MTU，需要将RS->Client SYN\_ACK包中的MSS减小TCPOLEN\_ADDR（8字节）；

新增函数tcp\_opt\_adjust\_mss，在函数tcp\_fnat\_in\_handle中被调用。

注意点：需重新计算校验和；

### Local address管理

Local address以list方式维护在virtual server中，类似realserver；

1. ip维护：采用链表，RR轮转；
2. port维护：通过查找session表，判断端口是否可用；

不同工作模式，选择不同local address:

1. FULLNAT– 从virtual server的local address list中选取；
2. DR/NAT/TUNNEL – 采用client ip作为local address，从而实现不同转发模式下，查找session等的统一处理；

### 序列号转换

#### 4.6.1 正常流程

选择local address后，需要重新计算seq，以满足同一个TCP流中的SYN包序列号单向递增的要求；

1. Seq初始化

新增函数tcp\_in\_init\_seq，用于序列号初始化工作，在函数tcp\_fnat\_in\_handle中被调用；

序列号计算采用内核标准函数**secure\_tcp\_sequence\_number；**

注意：IPV4和IPV6序列号计算方式不同；

1. OUT2IN，转换seq

新增函数tcp\_in\_adjust\_seq，在函数tcp\_fnat\_in\_handle中被调用；

1. IN2OUT，转换ack\_seq

新增函数tcp\_out\_adjust\_seq，在函数tcp\_fnat\_out\_handle中被调用**；**

#### 4.6.2 syn包命中已有session，则重新计算seq

### ICMP包处理

ICMP包主要在 ip\_vs\_in\_icmp 和 ip\_vs\_out\_icmp 2个函数中处理；

### 4.8 RESET功能

Established状态的session超时释放后，client/rs可能认为相关socket还是活跃的，从而影响业务；

解决方法：对于Established状态的session，超时释放时，向client和rs发送reset包；

### 4.9控制接口

FULLNAT增加了local address 和 fullnat转发模式 2个配置；

1. Fullnat转发模式
   1. 该信息维护在svc->flags中，因此，需要在ip\_vs.h头文件中增加

#define IP\_VS\_CONN\_F\_FULLNAT 0x0005 /\* full nat \*/

* 1. 需要修改keepalived和ipvsadm程序

1. Local address
   1. 每个service都有自己的local address list
   2. 在sockopt/netlink中，增加

IP\_VS\_SO\_SET\_ADDLADDR

IP\_VS\_SO\_SET\_DELLADDR

IP\_VS\_SO\_GET\_LADDRS

* 1. 需要修改keepalived和ipvsadm程序

## 5 其它因素考虑

### 5.1 对于其它模块的影响

Template/persistent/ipv6/udp/icmp各个模块的影响，处理可能考虑不全；

### 5.2 测试考虑

1. 测试NAT/DR/TUNNEL 兼容性
2. 测试TCP/UDP/ICMP/AH/ESP 数据包
3. 测试IPV6
4. 测试Template/persistent