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：

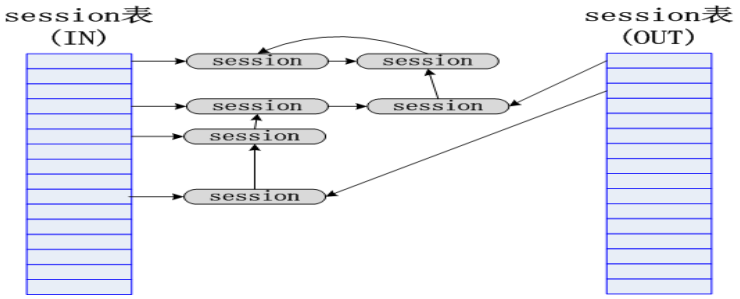
1. Packet IN 时，目标 ip 更换为 realserver ip，源 ip 更换为 内网 local ip；
2. Packet OUT 时，目标 ip 更换为 client ip，源 ip 更换为 vip；

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。

4.2 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

4.3 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；

4.4 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；

4.5 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；

4.6 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；

2. OUT2IN，转换 seq
   新增函数 tcp_in_adjust_seq，在函数 tcp_fnat_in_handle 中被调用；

3. 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

4.7 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
   a) 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 */
   b) 需要修改 keepalived 和 ipvsadm 程序
2. Local address
   a) 每个 service 都有自己的 local address list
   b) 在 sockopt/netlink 中，增加
      IP_VS_SO_SET_ADDLADDR
      IP_VS_SO_SET_DELLADDR
      IP_VS_SO_GET_LADDRS
   c) 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