# Mac acl代码分析

### 背景

mac acl目前是揉在ip acl之中的，虽然现在没有明显的问题，但后续如果再加功能将会越来越难以维护，故此需要将其剥离出来，单独形成一条流程。

### acl在rps的代码架构

rps层面，acl的代码作用就是配置下发到hsl，这里简要地说明一下配置方式和代码中用什么数据结构来存储acl的规则，然后下发给hsl。

#### 配置方式：

**mac access-list** **extended** *name*

**permit** { *source-address source-wildcard* | **any** } { *dest-address dest -wildcard* | **any** } [**ether-type** *ether-type-number*] [**out-vlan** *out-vlan-number*] [**out-8021p**  *out-8021p-number*] [**in-vlan** *in-vlan-number*] [**in-8021p** *out-8021p-number*] [ **time-range** *time-range-name* ] [ **sequence** *sequence-number* ]

**deny** { *source-address source-wildcard* | **any** } { *dest-address dest -wildcard* | **any** } [**ether-type** *ether-type-number*] [**out-vlan** *out-vlan-number*] [**out-8021p**  *out-8021p-number*] [**in-vlan** *in-vlan-number*] [**in-8021p** *out-8021p-number*] [ **time-range** *time-range-name* ] [ **sequence** *sequence-number* ]

**mac-access-group** *name* { **in** | **out** }

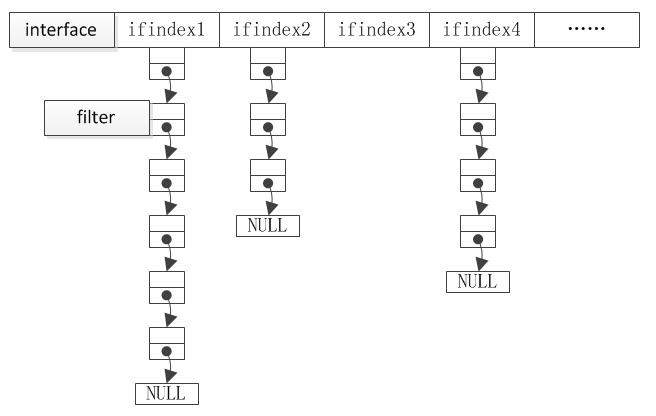
配置方式，总体来说分为三个步骤:

第一步：配置一条access-list，也就是acl group，取个响亮的名字

第二步：在access-list下，配置一或多条acl规则，也就是permit或者deny什么数据

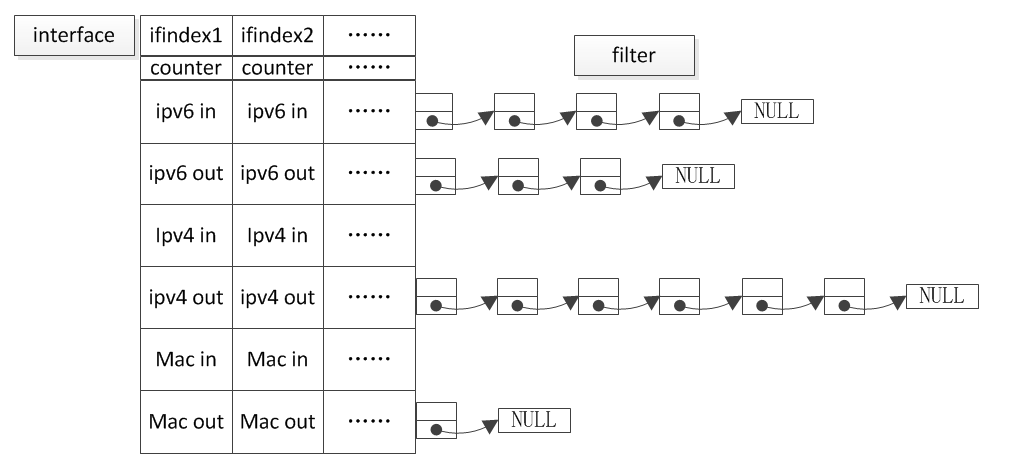
第三步：在接口下，将一条配置好的access-list绑定到接口中，并指定出入方向

从以上的配置步骤可以猜测，内存中acl的数据结构大概是基于interface（接口）的一个哈希链表，每个接口对应多个filter（acl规则）：



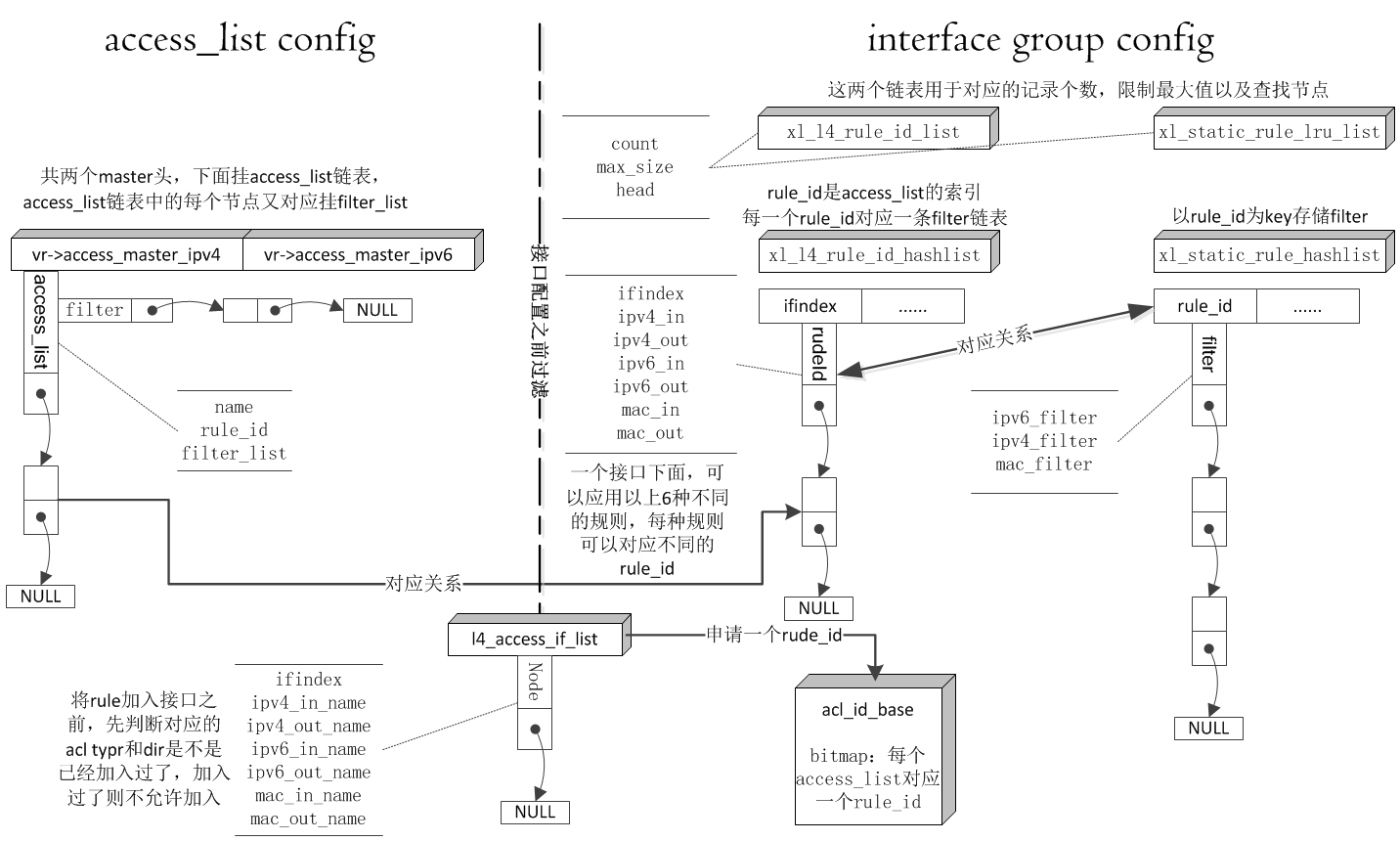
考虑到filter的计数，添加filter counter字段

一个接口下面，相同类别和方向的group只能配置一个，比如ip acl，in方向，配了in方向还可以配置out方向。另外还可以配置mac acl的in方向，和out方向。另外还有ipv6，所以一个接口最多配置6个group。所以需要一个结构体，里面包含这6种情况，对应每一种情况都要挂一条链表，如下图



#### 数据结构：

总体：



图中已经表示得很清楚了，不再赘述。最终的目的就是要形成右边接口对应acl group的数据结构，然后将此数据结构定时下发到sdk

附：

非接口模式的代码数据结构

