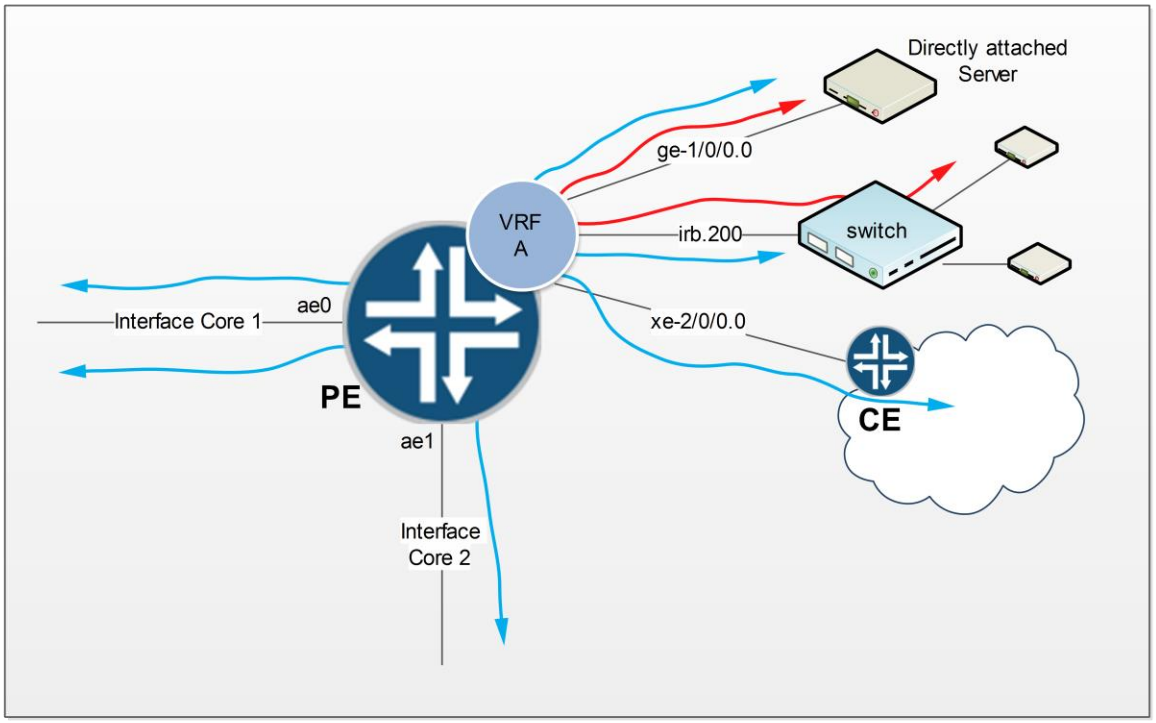
A PER VRF ACCOUNTING SOLUTION

http://www.junosandme.net

By David Roy

What is the aim?

Recently I have to find a solution to count input and output traffic in a per VRF basis. Said like that it seems that the solution was very easy: actually not really. Indeed, the typical VRF that I had was the following. It is depicted by the following figure:



In the figure above, we have a simple PE, with two core interfaces (MPLS) and several VRFs. Here only one is shown. The VRF A connects several types of interface:

- Classical PE/CE interfaces (xe-2/0/0). The CE and PE communicates together for instance with BGP. Data plane traffic is generated by servers/equipment behind the CE.
- Directly attached equipment on ge-1/0/0.0. There is no dynamic protocol between the server and the PE.
- Some equipment attached to a switch. The switch is connected to a bridge interface of the PE within the VRF. The Layer 3 routing interface is the irb.200 to reach the core.

Accounting requirements

The aim is to count only traffics that come from/to the core interfaces (i.e ae0, ae1) from/to the VRF interfaces (i.e. ge-1/0/0; irb.200; xe-2/0/0): these are the blue flows. All the intra-VRF traffics must not be counted: the red flow for instance.

The solution which answers to this problematic should be the most possible generic: it means does to depend on the interface’s name.

Which solutions?

I spent some times to find a good solution. I tried to do that with simple interface firewall filters or forwarding firewall filter in instance-shared mode but each time the exclusion of intra-VRF traffic was too complex or impossible. I also studied the LSI and VT interface solution without success. Finally I found a nice solution based on SCU (Source Class Usage) and DCU (Destination Class Usage).

Why I didn't use only the SCU feature? Actually there is a current limitation with SCU which is the following:

Classes cannot be mapped to directly connected prefixes configured on local interfaces.

In my setup, I have directly attached servers that directly generated traffic. With only SCU feature I was not able to count traffic, from example, generated by server attached to ge-1/0/0 and wishing to reach the core. A mix of SCU and DCU was therefore mandatory.

Some words about SCU/DCU

Here some info extracted from the Juniper documentation:

> Source class usage (SCU) counts packets sent to customers by performing lookups on the IP source address and the IP destination address. SCU makes it possible to track traffic originating from specific prefixes on the provider core and destined for specific prefixes on the customer edge. You must enable SCU accounting on both the inbound and outbound physical interfaces.
>
> Destination class usage (DCU) counts packets from customers by performing lookups of the IP destination address. DCU makes it possible to track traffic originating from the customer edge and destined for specific prefixes on the provider core router.

Packets that should be counted can be filtered by a specific forwarding export policy. You can define up to 126 SCU/DCU class per chassis.

The solution

Let's dive into the solution. First of all, you have to define your SCU/DCU forwarding export policy. Mine was very simple. I didn't want to filter traffic on based on prefix, community or anything else. I just wanted to count traffic coming from / to the core interfaces.

Here my policy:

```
policy-statement load-balancing-policy {
    term 0 {
        to instance master;
        then destination-class toCORE;
    }
    term 1 {
        from instance master;
        then source-class fromCORE;
    }
    term 2 {
        then {
            load-balance per-packet;
            accept;
        }
    }
}
```

The policy is applied like that:

```
routing-options {
    forwarding-table {
        export load-balancing-policy;
    }
}
```

As you observed I merged the load-balancing policy (term 2) with my SCU/DCU policy (term 0 and 1) in a single policy. Even if packet passes through term 0 and 1 it will finally reach term 2.

Term 0 matches all traffic which is destined to the "master" routing instance. In my topology only MPLS core interfaces ae0 and ae1 are part of the master instance. All the other interfaces are attached to VRFs, therefore to other routing instances.

The term 1 matches the traffic coming from the master routing instance- i.e from ae0 or ae1.

As you can see all traffic matching term 0 will be associated to the DCU class "toCore": this will be the case for traffic coming from interfaces of the VRF and destined to core interfaces. Intra-VRF traffic won't match this term because traffic will be destined to a "VRF" instance and not to the master instance.

The action on term 1 is the reverse path. Traffic matching term 1 will be associated to the SCU class "fromCore"

Let's move on. In Later 3 VPN environment, you need to enable vrf-table-label knob with the SCU option in your VRF.

```
routing-instances {
    VRF-A {
        vrf-table-label source-class-usage;
    }
}
```

Then configure SCU/DCU on each interface you want enable accounting. For core interface we configure SCU input (we also do the same for ae1)

```
interfaces {
    ae0 {
        unit 0 {
            family inet {
                accounting {
                    source-class-usage {
                        input;
                    }
                }
            }
            family mpls;
            family iso;
        }
}
```

For VRF's interface we configure SCU output and DCU (hereafter the example of irb.200 but this is same for other VRF's interfaces ge-1/0/0 and xe-2/0/0)

```
interfaces {
    irb {
        unit 200 {
            family inet {
                accounting {
                    source-class-usage {
                        output;
                    }
                    destination-class-usage;
                }
            }
        }
}
```

What does it mean exactly?

A packet entering in the router in ae0, where SCU Input is configured, will first pass to a "source-lookup. Then the result of the source lookup will try to match a term in our SCU policy. In our case it will match term 1 because traffic coming from ae0 is coming from master instance. Then a "destination lookup" will be performed to find if the forwarding interface has SCU output configured. We suppose that the MPLS traffic coming from ae0 has to reach server attached to irb.200. In our case SCU output is configured on irb.200. Only in this case the packet will be internally marked with de SCU "fromCore" class and will be counted on irb.200 when it will leave the router. Removing SCU output on irb.200 would stop the accounting.

For the reverse path: A packet coming from irb.200 will first perform a “destination lookup” and the forwarding next hop resulting of this lookup will try to match an entry in our policy because the DCU’s option is configured on ingress irb.200 interface. In our case the destination will be ae0, so the packet will match term 0 (to master instance) and will be marked internally with the DCU “toCore” class and will be counted on irb.200 interface.

Once committed you have now per direction “fromCore” and “toCore” counter per VRF’s interface. Here the example on irb.200 and ge-1/0/0.0

```
droydavi@sponge> show interfaces irb.200 statistics
 Flags: Sendbcast-pkt-to-re, DCU, SCU-out
                                       Packets                     Bytes
      Destination class           (packet-per-second)    (bits-per-second)
                           toCore          294137303          434734933834
                                  (             4000)  (          47297800)
                                       Packets                     Bytes
      Source class                (packet-per-second)    (bits-per-second)
                         fromCore           73037221          107949012638
                                  (              999)  (          11821472)
```

```
droydavi@sponge> show interfaces ge-1/0/0.0 statistics
 Flags: Sendbcast-pkt-to-re, DCU, SCU-out
                                       Packets                     Bytes
      Destination class           (packet-per-second)    (bits-per-second)
                           toCore          294137303          434734933834
                                  (              348)  (            429780)
                                       Packets                     Bytes
      Source class                (packet-per-second)    (bits-per-second)
                         fromCore           73037221          107949012638
                                  (              100)  (            121472)
```

Remember that intra-VRF traffic (i.e. ge-1/0/0 to xe-2/0/0) and intra-master-instance traffic (i.e. ae0 to ae1) will be not counted due to the fact they do not match any entry of the SCU/DCU policy + depending on where we configured SCU in, SCU out or DCU on the interfaces.

The solution was almost finished because here we have on each interface of each VRF two counters that show us the traffic coming from the core or destined to the core. But we don’t have the aggregated statistics for a given VRF (I mean the global traffic for a VRF of all VRF’s interfaces). Nevertheless, you can do that manually or via scripting something on your NMS, because SCU/DCU counters are available per interface via SNMP.

```
droydavi@sponge> show snmp mib walk jnxScuStatsBytes ascii
jnxScuStatsBytes.873.1."fromCore" = 196124217996
jnxScuStatsBytes.892.1."fromCore" = 107550163992
jnxScuStatsBytes.896.1."fromCore" = 0
jnxScuStatsBytes.901.1."fromCore" = 108043057778

droydavi@sponge> show snmp mib walk jnxDcuStatsBytes ascii
jnxDcuStatsBytes.873.1."toCore" = 1291472480344
jnxDcuStatsBytes.892.1."toCore" = 322679497726
jnxDcuStatsBytes.896.1."toCore" = 0
jnxDcuStatsBytes.901.1."toCore" = 435158135486
```

But could we do that directly on the router itself. Actually yes! We have just to create a dedicated firewall filter per VRF, like that:

```
firewall {
    family inet {
        filter FWFR-<VRF-NAME> {
            term in {
                from {
                    source-class fromCORE;
                }
                then {
                    count in;
                    accept;
                }
            }
            term out {
                from {
                    destination-class toCORE;
                }
                then {
                    count out;
                    accept;
                }
            }
            term other {
                then accept;
            }
        }
    }
}
```

And apply it like that on the given VRF:

```
routing-instances {
    VRF-A {
        vrf-table-label source-class-usage;
        forwarding-options {
            family inet {
                filter {
                    output FWFR-VRF-A;
                }
            }
        }
    }
}
```

With this solution you directly have 2 aggregated counters per VRF (in and out) which aggregate the fromCore and toCore counters of all interfaces part of a given VRF. Your NMS can pool directly these 2 aggregated counters if needed but counters are also available by cli: show firewall filter xxxx

David.