|  |  |
| --- | --- |
|  |  |

eVIP Elements and Traffic Handling

# Abstract

# *this document describes some elements (lbeagent, lberep, seselector, pnagent, lbeselector and repDB) on EVIP and the solution to handle traffics.*

Contents

[1 Abbreviations 3](#_Toc485818270)

[2 Overview 4](#_Toc485818271)

[3 SE selector 4](#_Toc485818272)

[4 LBE Agent 4](#_Toc485818273)

[5 LBE Rep 4](#_Toc485818274)

[6 PN Agent 6](#_Toc485818275)

[7 LBE Selector 6](#_Toc485818276)

[8 repDB 6](#_Toc485818277)

[9 Packet Handling on LBE 6](#_Toc485818278)

[10 Packet handling on PN 9](#_Toc485818279)

[Reference 10](#_Toc485818280)

# Abbreviations

PN: payload node.

IPVS: IP virtual server.

# Overview

LBE (Load Balancing Element) uses IPVS to distribute incoming connections over PNs depending on the configured distribution algorithm (WRR is used in SBG). On an LBE, the following eVIP subsystems/entities are involved.

* SE selector
* LBE Agent
* LBE Rep

# SE selector

The SE selector agent runs on LBE to forward an outgoing packet to a suitable SE with the help of HMARK.

The HMARK [2] module is activated by LBE agent when generating the IPTABLE rules. There is a rule jumping to chain “EVIP\_HMARK\_FORWARDING” resides in the PREROUTING chain of the mangle table.

When an outgoing packet traverses the LBE agent, the HMARK module assigns a fwmark for this connection, which can make sure that the packets return to the same SE as the incoming connections. The below values are used to calculate the fwmark in *sesel\_sd.c* :

*--hmark-mod: 256;*

*--hmark-tuple {src,dst};*

*--hmark-offset: 999 (v4); 998(v6);*

*--hmark-rnd: 0xc175a3b8*

They are also used to generate a iptables output rule:

***Chain EVIP\_HMARK\_FORWARDING (2 references)***

*pkts bytes target prot opt in out source destination*

*0 0 HMARK all -- \* \* 10.21.0.5 0.0.0.0/0 HMARK mod 256 + 0xc00 src-prefix 32 dst-prefix 32 rnd 0xc175a3b8*

***llb1lbe\_0\_15: ~# ip rule show***

*100: from all fwmark 0xc00/0xffffff00 lookup 4071*

***llb1lbe\_0\_15: ~# ip route show table 4071***

*default dev se scope link*

# LBE Agent

# LBE Rep

Load Balancer Element Replicator, replicates connection information to all repDBs on all payloads and also receive replicated connections from repDBs.

When an LBE fails, the saved/replicated information can be used to move the replicated connections to surviving LBEs. UDP is used to replicate connection information . TCP is used to inject connection information from repDB into kernel space.

## PVS Daemon

At startup, lberep creates two IPVS daemon: master daemon and backup daemaon [4][6] ,

Ipvsadm –start-deamon master –mcast-interface=lo –syncid 0

pvsadm –start-deamon backup –mcast-interface=lo –syncid 1

and multicast route is also created, “ip route add 224.0.0.81 dev lo”.

The lbeRep also creates two UDP sockets for receiving and sending connection information from/into IPVS daemon on “224.0.0.81:8848” when starting up.

IPVS master daemon replicates connection information to lbeRep though UDP multicast and lbeRep picks up those packets and inserts into a Queue according to the destination address. The connection information in the UDP message are separated with respect to their destination address and this makes the replication database distributed over the PNs.

## Queue System.

lbeRep uses Parlux Protocol E to connect to every repDB servers at the port 25105 [3]. Each repDB address has one queue on lbeRep to cache the packets that incoming from kernel space. The allocated memory for each queue is set to the MTU-size.

The queue system on the lberep is set up to handle sorting/sending of the connection information. PNID is used to sort the connections. lberep has received a mapping from eVIP-C to be able to translate the PNID to control traffic addresses. This translation is made at send time for each queue with API “convert\_pnid\_to\_addr”.

A queue is sent to the corresponding RepDB when one of these two things happens:

* The total size of connections in a queue sums up to the MTU size
* a timer is triggered.

The queue system has three purposes:

* First, to package/sort the connections going to different RepDB:s.
* Secondly, to make sure that connection information is replicated with a short enough interval.
* Lastly, packages that are supposed to be replicated should not exceed the MTU size of the interface

## Failover

During failover, a repDB firstly gets a signal from eVIPC,using [Protocol C](https://openalm.lmera.ericsson.se/plugins/mediawiki/wiki/evip/index.php/EvipPage097), that a **new LBE selector key map** is available, e.g.

“repa[12787]: Told to start keymap update with flags 3”

The repDB uses this map to decide which connection information that needs to be moved to a new LBE.

lbeRep receives packets including connection information from repDB and inject connection information into the backup Daemon though UDP. The Backup Daemon can update the IPVS connection automatically.

To receive packets from repDb, lbeRep also creates a TCP server on port 25161 .It introduce libev to monitor the IO event on TCP socket and read injection message from TCP packets. Then, send to backup daemon after parsing the message.

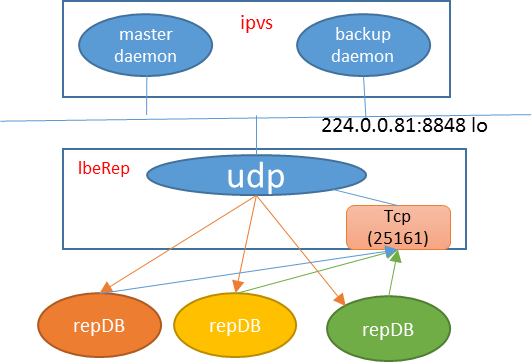


Figure 1 template && connection replication

# PN Agent

# LBE Selector

# repDB

# Packet Handling on LBE

* Tcp packet Handling

Load balancing TCP connections begins with when a TCP SYN packet is received for a virtual server, an iptables-rule redirect the packet with the help of a **fwmark**, the fwmark corresponds to a certain virtual server. This packet is sent to one of the real servers and the LVS/IPVS stores this connection and guarantees that all packets for the connection are sent to the same real server.

Once a connection is closed, the connection record is kept for a short period of time so that any remaining packets will be transmitted to the correct real server. After a long period of inactivity, the connection record is torn down.

* Udp Packet Handling

UDP handling on EVIP has Stateless UDP (distribution per packet) and Stateful UDP (SBG case, iptable rules with fwmark).

* ESP packet handling

IPSEC packets are always forward if it matches the destinatin VIP address and mainly the fake UDP port 500 connections we use to be able to handle ESP.

## packet route

* fwmark

Normally, The virtual server can be a donaim server or a fwmark working with iptables rules. On SBG, we use fwmark to route incoming packets:

*blade\_0\_11:homer# iptables -nvL -t mangle*

*Chain PREROUTING (policy ACCEPT 1379K packets, 57M bytes)*

*pkts bytes target prot opt in out source destination*

*1379K 57M EVIP\_HMARK\_FORWARDING all -- \* \* 0.0.0.0/0 0.0.0.0/0*

*0 0 DROP icmp -- \* \* 0.0.0.0/0 0.0.0.0/0 ctstate INVALID*

*1379K 57M EVIP\_CONNTRACK\_FORWARDING all -- \* \* 0.0.0.0/0 0.0.0.0/0*

*0 0 EVIP\_ICMP\_CT\_FWD icmp -- \* \* 0.0.0.0/0 0.0.0.0/0*

*979K 39M EVIP\_FP\_VIP\_ID\_1\_tcp tcp -- \* \* 0.0.0.0/0 11.0.0.1*

*0 0 EVIP\_FP\_VIP\_ID\_1\_udp udp -- \* \* 0.0.0.0/0 11.0.0.1*

*0 0 EVIP\_FP\_VIP\_ID\_1\_esp esp -- \* \* 0.0.0.0/0 11.0.0.1*

*Chain EVIP\_FP\_VIP\_ID\_1\_esp (1 references)*

*pkts bytes target prot opt in out source destination*

*0 0 MARK esp -- \* \* 0.0.0.0/0 11.0.0.1 MARK set 0x5b8d81*

*Chain EVIP\_FP\_VIP\_ID\_1\_tcp (1 references)*

*pkts bytes target prot opt in out source destination*

*399K 16M MARK tcp -- \* \* 0.0.0.0/0 11.0.0.1 tcp dpt:5060 MARK set* ***0x5b8d82***

*Chain EVIP\_FP\_VIP\_ID\_1\_udp (1 references)*

*pkts bytes target prot opt in out source destination*

*0 0 MARK udp -- \* \* 0.0.0.0/0 11.0.0.1 udp dpt:5060 MARK set 0x5b8d82*

*0 0 MARK udp -- \* \* 0.0.0.0/0 11.0.0.1 udp dpt:500 MARK set 0x5b8d82*

*Chain EVIP\_ICMP\_CT\_FWD (1 references)*

*target prot opt source destination*

*ACCEPT all anywhere 3001::5/128 connmark match ! 0x0*

*EVIP\_ICMP\_FORWARDING ipv6-icmp anywhere anywhere [goto]*

Chain EVIP\_ICMP\_FORWARDING (1 references)

target prot opt source destination

MARK all -- anywhere *11.0.0.1* MARK set 0x1870e

ACCEPT all -- anywhere *11.0.0.1* mark match 0x1870e

* IPVS

the fwmark is used by IPVS to route the packets to realserver through a virtual server.

llb1lbe\_0\_11:~# ipvsadm -L

IP Virtual Server version 1.2.1 (size=8388608)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

FWM 6000001 IPv6 wrr persistent 370000

-> [fc00::6e]:0 Route 6300 0 0

FWM 6000002 wrr persistent 370000

-> 169.0.0.110:0 Route 6300 0 0

* Ip rule and ip route

lbeAgent creates the ip rules and ip routes to the PNs.

llb1lbe\_0\_11:~# ip rule ls

100: from all to 169.0.0.110 lookup 100110

100: from all fwmark 0x1870e lookup 100110 //the rules for conntrack steering traffic (reply traffic from a server)

100: from all to 169.0.0.150 lookup 100150

100: from all fwmark 0x5b8d82 lookup 6000002

llb1lbe\_0\_11:~# ip route show table 100110

default dev e\_tun\_1 scope link

llb1lbe\_0\_11:~# ip route show table 100150

default dev e\_tun\_1 scope link

## paired system

Ipvs has no active-standby concept. The real server cannot be changed during active-standby failover. When real server starts, only one PN of the pair is registered in IPVS, the **fake address** of PN with the lower id disregarding whether it is active or standby:

EVIP(config)# show pn-pairs

o = standalone, a = active, s = standby

150 s:110

110 a:150

llb1lbe\_0\_11:~# ipvsadm -S

-A -f 6000001 -6 -s wrr -p 370000

-a -f 6000001 -6 -r [fc00::6e]:0 -g -w 6300

-A -f 6000002 -s wrr -p 370000

-a -f 6000002 -r 169.0.0.110:0 -g -w 6300

llb1lbe\_0\_11:~# ip -6 tunnel

ip6tnl0: ipv6/ipv6 remote :: local :: encaplimit 0 hoplimit 0 tclass 0x00 flowlabel 0x00000 (flowinfo 0x00000000)

e\_tun\_1: any/ipv6 remote fe80::200:ff:fe01:10f local fc00::200 dev evip\_macvlan0 encaplimit 4 hoplimit 64 tclass 0x00 flowlabel 0x00000 (flowinfo 0x00000000)

e\_tun\_2: any/ipv6 remote fe80::200:ff:fe01:10c local fc00::200 dev evip\_macvlan0 encaplimit 4 hoplimit 64 tclass 0x00 flowlabel 0x00000 (flowinfo 0x00000000)

se: any/ipv6 remote fc00::300 local fc00::200 dev evip\_macvlan0 encaplimit 4 hoplimit 64 tclass 0x00 flowlabel 0x00000 (flowinfo 0x00000000)

Although here has two different fake address of 11 and 15, but they have same tunnel ip address, is fe80::200:ff:fe01:10f , blade\_0\_11 active.

Once the active failover, the ip route table is also changed to peer:

llb1lbe\_0\_11:~# ip route show table 100110

default dev e\_tun\_2 scope link

For **incoming ICMP traffic**, EVIP\_ICMP\_FORWARDING chain decides where to forward packets, using marks we have already seen for conntrack:

alb1lbe\_1 ~ # iptables -t mangle -L EVIP\_ICMP\_FORWARDING

target prot opt source destination

MARK all -- anywhere *11.0.0.1* MARK set 0x1870e

ACCEPT all -- anywhere 11.0.0.1 mark match 0x1870e

# Packet handling on PN

Reference

[1] <https://openalm.lmera.ericsson.se/plugins/mediawiki/wiki/evip/index.php/EvipPage030>

[2] <http://ipset.netfilter.org/iptables-extensions.man.html>

[3] <https://openalm.lmera.ericsson.se/plugins/mediawiki/wiki/evip/index.php/EvipPage099>

[4] <http://www.linuxvirtualserver.org/docs/sync.html>

[5] <https://openalm.lmera.ericsson.se/plugins/mediawiki/wiki/evip/index.php/EvipPage017>

[6] <http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/LVS-HOWTO.server_state_sync_demon.html>