Linux Virtual Server High Availability using VRRPv2

Alexandre Cassen
Linux Virtual Server OpenSource Project
Paris, France
acassen@linux-vs.org, http://www.LinuxVirtualServer.org/~acassen/

Abstract

This paper describes one implementation of the VRRPv2 protocol. We have done enhancements of the original protocol for use with Linux Virtual Server. The goal of this document is to present the software design of the VRRPv2 implementation for use with LVS. We will focus on the implementation, so attempted audience is network software designer, experimented system administrators and all hacking person looking for VRRPv2 implementation internals. Finally the goal of this document is a real world example on how our design can be used in conjunction with Linux Virtual Server to provide virtual services shared by multiple LVS directors.

1. Introduction

Loadbalancing is a good solution for service virtualization. When you design a loadbalanced topology you must take a special care of :

- Realserver availability using healthchecking.
- Loadbalancer availability using failover protocol.

Realserver availability provide a global High Available virtual service. This problematic is mainly handled using healthcheck software. But using a loadbalancer director you introduce a Single Point Of Failure for the virtual service. So loadbalancer high availability must be provided. In this paper we will not present healthchecker, we will specially focus on the second point. We will trait this topic using a protocol especially designed to handle gateway virtualization/failover called VRRPv2. This protocol will give us the ability to define a "Virtual Loadbalancer". This "Virtual Loadbalancer" will be compounded by multiple "Real Loadbalancer" that can be active at a time and monitoring each other.

2. VRRPv2 Overview

"VRRP specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers on a LAN. The VRRP router controlling the IP address(es) associated with a virtual router is called the Master, and forwards packets sent to these IP addresses. The election process provides dynamic fail over in the forwarding responsibility should the Master become unavailable. This allows any of the virtual router IP addresses on the LAN to be used as the default first hop router by end-hosts. The advantage gained from using VRRP is a higher availability default path without requiring configuration of dynamic routing or router discovery protocols on every end-host." [rfc2338].

3. System Architecture Overview

Our global logical architecture is illustrated in the following figure :

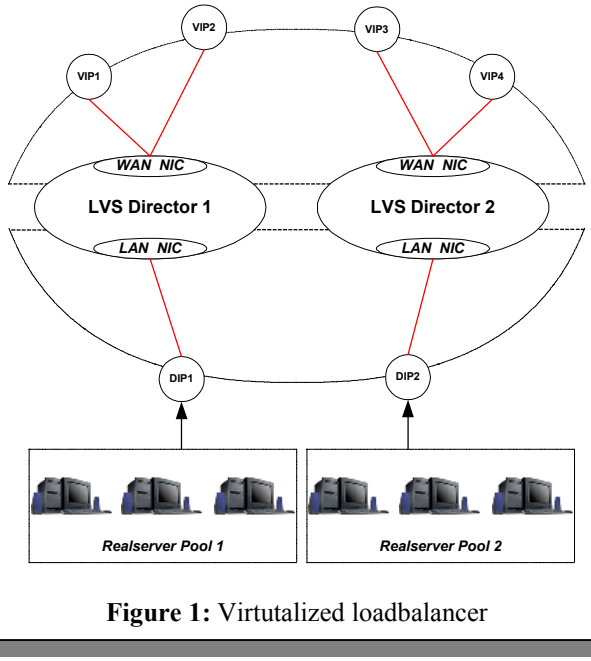

Figure 1: Virtutalized loadbalancer

As described into the rfc2338, VRRPv2 protocol manipulate floating IP addresses roaming between routers.

3.1 Definitions

Our VRRPv2 implementation uses the following notions [rfc2338.1.2] :

- ***VRRP Instance*** : A thread manipulating VRRPv2 specific set of ip addresses. A VRRP Instance may backup one or more VRRP Instance. In our figure 1 we are dealing with 4 VRRP Instances. One owning (VIP1,VIP2), one owning (VIP3,VIP4), one owning (DIP1) and one owning (DIP2). It may participate in one or more virtual routers.
- ***IP Address owner*** : The VRRP Instance that has the IP address(es) as real interface address(es). This is the VRRP Instance that, when up, will respond to packets addressed to one of these IP address(es) for ICMP, TCP connections, …

- ***MASTER state*** : VRRP Instance state when it is assuming the responsibility of forwarding packets sent to the IP address(es) associated with the VRRP Instance. This state is illustrated on figure 1 by red line.
- ***BACKUP state*** : VRRP Instance state responsible of packets forwarding when the current VRRP Instance MASTER state fails.
- ***Real Loadbalancer*** : A LVS director running one or many VRRP Instances.
- ***Virtual Loadbalancer*** : A set of Real Loadbalancer.
- ***Synchronized Instance*** : VRRP Instance with which we want to be synchronized. This provide VRRP Instance monitoring.
- ***Advertisement*** : The name of a simple VRRPv2 packet sent to a set of VRRP Instances in MASTER state.

3.2 System workflow

We want to design a full redundant architecture with outgoing traffic loadbalancing. This is a very common use of such a protocol. When setting up LVS director high availability (virtualized loadbalancer) we want the backup director to handle traffic too. That way LVS directors will be redundant and active at a time. This architecture is illustrated on figure 1. We define half of the host to route through DIP1 and the other half to route through DIP2 [rfc2338.4.2]. This consideration introduce the synchronization handling.

In our case study, Realserver Pool 2 belong to virtual IP addresses VIP3 & VIP4. So remote client access those VIPs to access Realserver Pool 2 service. But Realserver Pool 2 route its outgoing traffic through DIP2, so DIP2 & VIP3/VIP4 must be active on the same director to preserve routing path. This finally mean that if LVS director 2 LAN interface fails, WAN interface IP address(es) owned must be set on the redundant LVS director 1 WAN interface.

Considering VRRP configuration, each LVS directors must have the knowledge of the whole LVS redundant topology. This mean that both LVS directors run 4 VRRP Instances. 2 VRRP Instances in MASTER state and the others in BACKUP state. MASTER/BACKUP state are symmetric on the both LVS directors VRRP configuration. On LVS director 1, the VRRP Instances configuration looks like :

- ***VI_1*** : In MASTER state - owning VRRP VIP1 & VIP2.
- ***VI_2*** : In MASTER state - owning VRRP DIP1.
- ***VI_3*** : In BACKUP state – backuping VRRP VIP3 & VIP4.
- ***VI_4*** : In BACKUP state – backuping VRRP DIP2.

On LVS director 2 we have the symmetric configuration where VI_1 & VI_2 are in BACKUP state and VI_3 & VI_4 are in MASTER state. If one VRRP Instance in MASTER state fails, to preserve routing path, the VRRP transition state must follow the workfow :

- VRRP DIP2 fails :

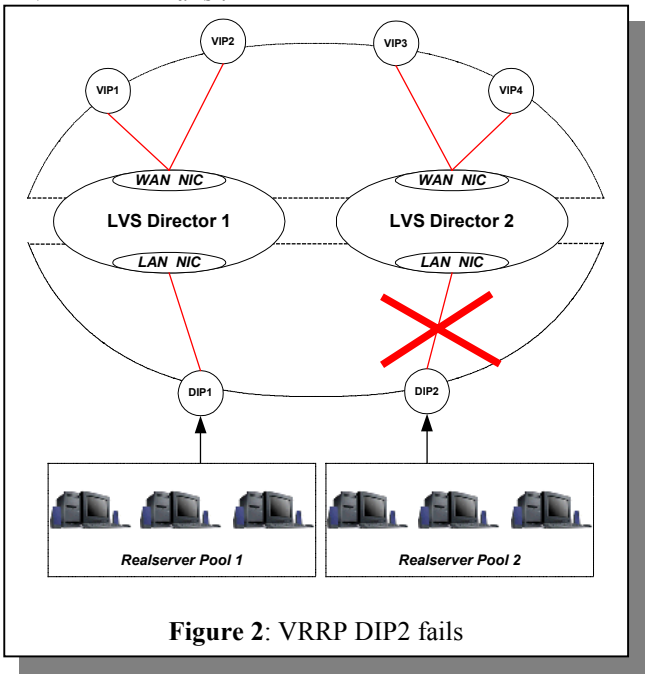

Figure 2: VRRP DIP2 fails

Due to an undefined reason, DIP2 is not available. Immediately, symmetric VRRP Instance, in BACKUP state, running on LVS director 1 stop receiving VRRP MASTER advertisements. It deduce that DIP2 is unreachable so the BACKUP state Instance must transit to MASTER state.

- VRRP Instance VI_4 on LVS director 1 transition to MASTER state :

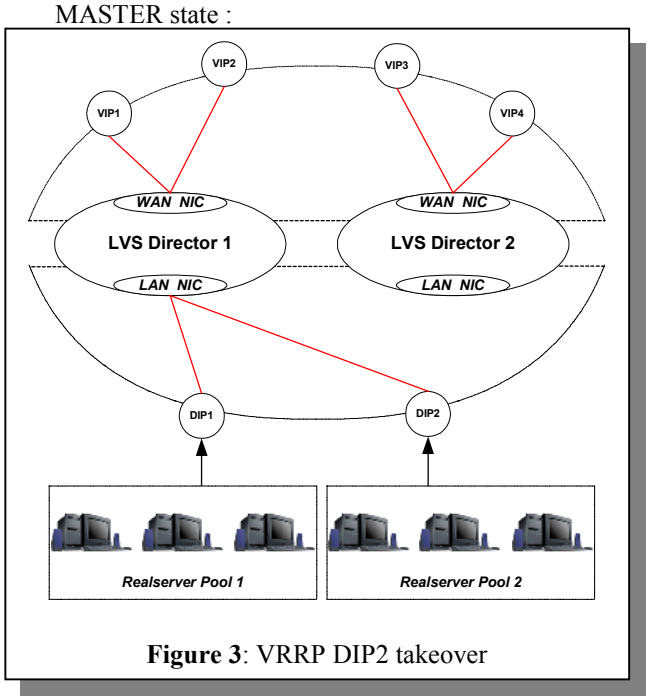

Figure 3: VRRP DIP2 takeover

LVS director 1 VI_4 stop receiving VRRP advertisements from LVS director 2 VI_4. The DIP2 automatically takeover on LVS director 1.

- VRRP Instance VI_3 on LVS director 1 synchronization to MASTER state :

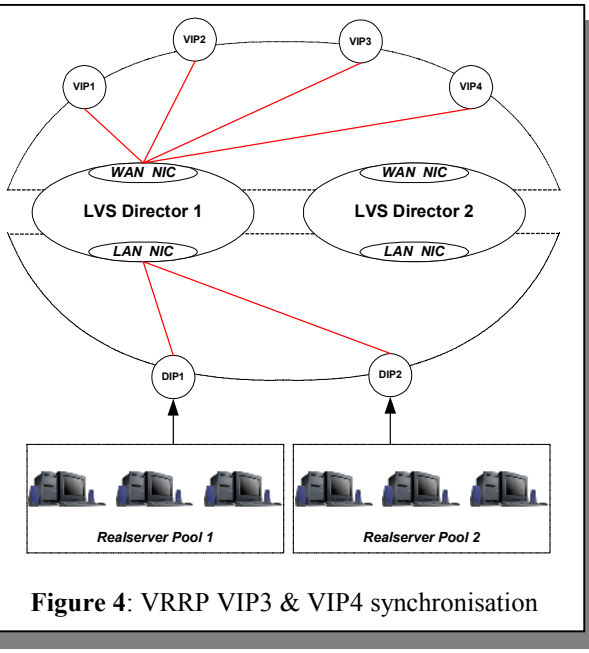

Figure 4: VRRP VIP3 & VIP4 synchronisation

Finally VIP3 & VIP4 are synchronized on LVS director 1 to preserve routing path. If LVS director 2 LAN interface is shut, this is a finite state fails for LVS. So others MASTER VRRP Instances need to be synchronized.

The VRRP MASTER Instances synchronization convergence is a derivation of the VRRP protocol. This mean that the IP takeover convergence is extremely depending on the VRRP timers handling.

In this fail state where LVS director 2 is depreciated, all traffic is handled by LVS director 1. This state must be considered as a takeover transition state. This mean that if LVS director 2 LAN interface is becoming available, all the VRRP Instances will transit to their initial VRRP state. So the stable state is the initial state, takeover state is completely temporary because it is all time waiting to transit to its initial state.

We have just describe this VRRP Instance synchronization state as it is not in the VRRP [rfc2338]. All the other state transition will not be presented in this document since it is efficiently described into the VRRP [rfc2338].

4. Implementation issues

The VRRPv2 implementation is articulated around 3 software components :

- VRRP Socket Pool
- VRRP Packet Dispatcher
- VRRP Instance Synchronization

VRRP implements its own scheduling framework to handle fd timeout and packet dispatching. So the 2 first point are referring to the VRRP scheduling framework.

4.1 VRRP Socket Pool

This part of the VRRP scheduling framework is only used during the VRRP bootstrap. It register the VRRP Instance to a global I/O multiplexing framework. The result of this step is the creation of a sharing read fd thread for future I/O dispatching. The design is illustrated in the Figure 5 :

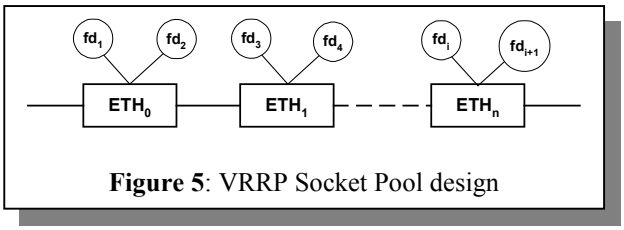

Figure 5: VRRP Socket Pool design

VRRP is an interface specific protocol. This mean that a VRRP Instance is bound to a specific interface for inbound & outbound traffic. We can figure out the socket pool importance here. As described into VRRP [rfc2338.5.3.6], VRRP is a layer4 protocol which can run in conjunction of IPSEC-AH. The VRRP advertisements can be either IPSEC-AH packet or direct VRRP packet protocol. So we need to deal with 2 different protocols at a time on the same physical interface (ethernet interface for us). This highlight that many VRRP Instance using the same protocol (IPSEC-AH or VRRP) share a single fd.

The figure 5 show our internal VRRP socket pool implementation. Each NIC own a maximum of 2 fds (one for VRRP, the other for IPSEC-AH). All the VRRP Instances are (de)multiplexed through this fds. So our design can handle 2*n (de)multiplexing points.

4.2 VRRP Packet Dispatcher

The second part of the global VRRP scheduling framework is the VRRP Packet Dispatcher. It provide an asynchronous threaded framework to handle incoming and outgoing VRRP packets. During the VRRP bootstrap, a VRRP Socket Pool is created. Finally the VRRP Socket Pool register a VRRP Packet Dispatcher per VRRP Socket Pool fds. Then each fds is handled asynchronously using a global software I/O multiplexer. This global workflow is illustrated in the figure 6 :

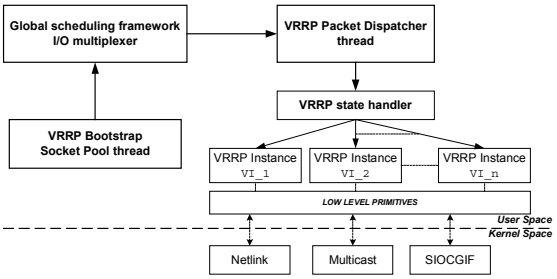

Figure 6: VRRP Packet Dispatcher design

A VRRP Packet Dispatcher can handle many VRRP Instances sharing the same fd. The central part of the packet dispatcher is the state handler responsible of VRRP state transition and synchronization.

4.3 VRRP Instance Synchronization

The last software component is the Instance Synchronization. The global view of this part is illustrated in the figure 4. The implementation follow the workflow :

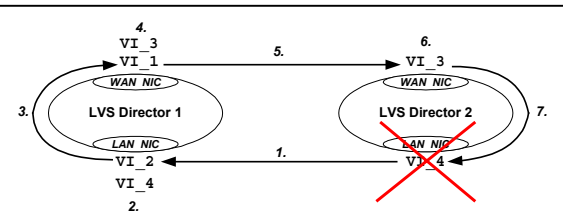

Figure 7: VRRP Instance Synchronization

On figure 7, we indicate VRRP Instance in MASTER state in initial state. To explain efficiently how Instance Synchronization works we take the sample illustrated in the figure 7. In this configuration we assume that (VI_1 ,VI_2) and (VI_3,VI_4) must be sync. So if one MASTER state instance transit to another state, the synchronized instance must transit to the same state.

1. LAN NIC on LVS director 2 fails, so VI_4 stop sending MASTER state advertisements.

2. VI_4 in BACKUP state timeout on receiving MASTER advertisements. So VI_4 transition to MASTER state on LVS director 1 and start sending advert (with IPSEC-AH sequence synchronization if using IPSEC-AH).

3. Our axiom imply that (VI_3,VI_4) must be in the same state at a time.

4. VI_3 in BACKUP state transit to MASTER state by sending the higher VRRP advertisement to force election (The VRRP owner advertisement).

5. All BACKUP and MASTER VRRP Instances received this owner adver.

6. VI_3 in MASTER state receive this advertisement and transit to BACKUP state.

7. Here is the synchronization end event.

If LAN NIC on LVS director 2 becomes alive all the VRRP Instances transit to their initial VRRP state. With the step 7 we can highlight a possible flapping Instance state due to a side effect. Axiom specify that (VI_3,VI4) must have the same state at a time. So during step 6, VI_3 transit to BACKUP state. So VI_4 must transit to BACKUP state too. But if VI_4 transit to BACKUP state due to a network failure, VI_4 will never receive MASTER state advertisements. So VI_4 on LVS director 2 can assume that MASTER is down and then transit to MASTER state it self ! And so on, axiom specify (VI_3,VI_4) must be sync. So VI_3 on LVS director 2 transit to MASTER state and start sending MASTER advert. Symmetrically, VI_3 receive remote MASTER advertisements and become BACKUP since its priority is minor than initial MASTER (VI_3 on LVS director 2). So on, axiom (VI_3,VI_4) must be sync… So VI_4 on LVS director 1 become BACKUP. But VI_4 on LVS director 1 timeout receiving remote MASTER advert so transit to MASTER state…. And so on… we repeat this infinitely so we have a synchronization loop causing a flapping roaming IP… We can call this phenomena a "synchronization circuit" (like the famous NOKIA one). To handle this possible synchronization infinite loop we can fine the axiom as follow :

I. Predicates

```
(1) init_state(VI_3)LVS2 <> init_state(VI_3)LVS1

(2) (1)=> if init_state(VI_3)LVS2 = MASTER
         then init_state(VI_3)LVS1 = BACKUP

(3) Prio(VI)MASTER > Prio(VI)BACKUP

(2),(3)=> Prio(VI_3)LVS2 > Prio(VI_3)LVS1

(5) (state(VI_3)=state(VI_4))LVS1
    <> (state(VI_3)=state(VI_4))LVS2

(6) state(VI_3)LVS2 = BACKUP
    & init_state(VI_3)LVS2 = MASTER
    => state(VI_4)LVS2 = FAULT

(7) (3)=> if state(VI_4)LVS2=DOWN
         then state(VI_4)LVS1=MASTER

(8) state(VI_3)LVS2=MASTER
    & state(VI_4)LVS2=FAULT
    => state(VI_4)LVS2=MASTER
```

II. Finite states handling

```
Finite state of figure 4 :
(5)&(7) =>
  if state(VI_3)LVS1=MASTER
  then state(VI_3)LVS2=BACKUP
(6) => state(VI_4)LVS2=FAULT
```

In the FAULT state the VRRP Instance continue sending advertisements. So when it become

```
alive it continue sending advertisements with-
out eventual IPSEC-AH sequences synchroniza-
tions.

Finite state initial state (after a takeover):
LAN NIC of LVS director 2 become alive :
(3) => state(VI_4)LVS1=BACKUP
(5) => state(VI-3)LVS1=BACKUP
    => state(VI_3)LVS2=MASTER
(8) => state(VI_4)LVS2=MASTER

So we are safely back to the initial state.
```

5. Conclusion & future work

In this paper we have introduced a solution for LVS directors failover using the VRRP protocol. This protocol was especially designed for this purpose and provide an acceptable security level using IPSEC-AH authentication (data integrity & anti-replay). The use of this code in conjunction of a healthchecking stack can provide a good solution for transparent directors & realservers failover. This is the goal of the keepalived project : http://keepalived.sourceforge.net : provide a all in one tool to provide realservers & directors failover.

Currently code has been tested with 2 LVS directors and 4 VRRP Instances. The code need to been audited with many VRRP Instances handling at a time. The VRRP Instance timer need to be fined during the instance synchronization to limit VRRP protocol convergence. Currently we do not handle VRRP VMAC since linux kernel doesn't permit to deal with more than one MAC address per physical interface. We use gratuitous ARP to update remote router/hosts arp caches (which can produce a TTL expiration during takeover). Need to spend more time into the kernel source to evaluate work… If you have inputs on this topics fill free to send it to me (it will save me time ☺)

References

[1] RFC2338, Virtual Router Redundancy Protocol, http://www.ietf.org/rfc/rfc2338.txt

[2] RFC2281, Cisco Hot Standby Router Protocol, http://www.ietf.org/rfc/rfc2281.txt

[3] Using HSRP for Fault-Tolerant IP Routing, http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs009.htm

[4] Internet DRAFT, Security Framework for Explicit Multicast, draft-paridaens-xcast-sec-framework-01.txt

[5] RFC2401, Security Architecture for the Internet Protocol, http://www.ietf.org/rfc/rfc2401.txt

[6] RFC2402, IP Authentication Header, http://www.ietf.org/rfc/rfc2402.txt

[7] RFC2104, HMAC: Keyed-Hashing for Message Authentication, http://www.ietf.org/rfc/rfc2104.txt

[8] Gowri Dhandapani, Anupama Sundaresan, Netlink Sockets – Overview, http://qos.ittc.ukans.edu/netlink/html/index.html

[9] W. Zhang, Linux Virtual Server OpenSource project, http://www.linuxvirtualserver.org