[**理解 OpenStack 高可用（HA）（2）：Neutron L3 Agent HA 之 虚拟路由冗余协议（VRRP）**](http://www.cnblogs.com/sammyliu/p/4692081.html)

本系列会分析OpenStack 的高可用性（HA）概念和解决方案：

（1）[OpenStack 高可用方案概述](http://www.cnblogs.com/sammyliu/p/4741967.html%20)

（2）[Neutron L3 Agent HA - VRRP （虚拟路由冗余协议）](http://www.cnblogs.com/sammyliu/p/4692081.html%20)

（3）[Neutron L3 Agent HA - DVR （分布式虚机路由器）](http://www.cnblogs.com/sammyliu/p/4713562.html%20)

（4）[Pacemaker 和 OpenStack Resource Agent （RA）](http://www.cnblogs.com/sammyliu/p/5025362.html%20)

（5）[RabbitMQ HA](http://www.cnblogs.com/sammyliu/p/4730517.html)

（6）[MySQL HA](http://www.cnblogs.com/sammyliu/p/4995868.html%20)

**1. 基础知识**

**1.1 虚拟路由冗余协议 - VRRP**

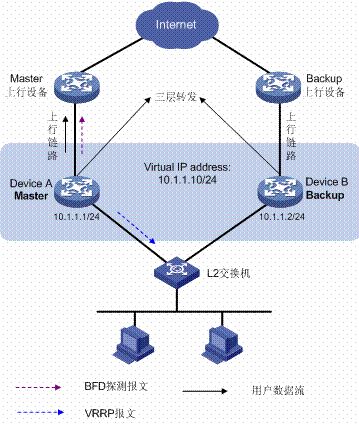
**1.1.1 概念**

    路由器是整个网络的核心。一个网络内的所有主机往往都设置一条缺省路由，这样，主机发出的目的地址不在本网段的报文将被通过缺省路由发往路由器，从而实现了主机与外部网络的通信。在通常只使用单路由器来承担缺省路由的情况下，当该路由器坏掉后，本网段内所有以它为缺省路由下一跳的主机将断掉与外部的通信。可见，在使用单路由器的情况下，如果路由器发生致命性的故障，将导致本地网络的瘫痪，如果是骨干路由器，影响的范围将更大，所造成的损失也是难以估计的。因此，对路由器采用热备份是提高网络可靠性的必然选择。

    VRRP（Virtual Router Redundancy Protocol，虚拟路由冗余协议）是一种可以解决这种问题的容错协议，其协议文本[在这里](https://www.ietf.org/rfc/rfc3768.txt)。 它保证当主机的下一跳路由器出现故障时，由另一台路由器来代替出现故障的路由器进行工作，从而保持网络通信的连续性和可靠性。一组实现了该协议的路由器向局域网内的主句提供具有高可用性（HA）的虚拟路由器。

**1.1.2 VRRP 怎么解决问题**

   VRRP 将一组物理路由器组织成一个虚拟路由器，称之为一个备份组。 这个虚拟的路由器拥有自己的 IP 地址，局域网内的机器使用该 IP 地址作为其缺省路由器的地址。



以上图为例，

* Device A 和 B 组成一个 VRRP 组，它的虚拟 IP（VIP） 为 10.1.1.10/24。其中，通过选举机制，A 是 Master Router，B 是 Backup Router。一个 VRRP 组内可以由多个设备，但是只有一个是 Master 设备。
* Device A 和 B 可以由自己的 IP 地址，VIP 可以和其中的某 IP 相同，也可以不同。
* 当前，Router A 作为 Master router 向局域网内的机器提供路由服务。
* 当前，Router B 作为 Backup router。它的任务是周期性地接受 A 发出的心跳。在规定的时间段内，如果都没有收到 A 发出的心跳，则启动一个选举过程，重新选出 Master。
* 局域网内的机器将虚拟路由器当作默认网关，它们仅仅知道这个虚拟路由器的IP 地址 10.1.1.10，而并不知道具体的 Master 路由器的 IP 地址以及 Backup 路由器的IP 地址。它们将自己的缺省路由下一跳地址设置为10.1.1.10。于是，网络内的主机就通过这个虚拟的路由器来与其它网络进行通信。如果 Master 路由器坏掉，Backup 路由器将会通过选举策略选出一个新的 Master 路由器，继续向网络内的主机提供路由服务。从而实现网络内的主机不间断地与外部网络进行通信。

可以看出：

* VRRP 是一种路由器选择协议，它可以把一个虚拟路由器的责任动态分配到 VRRP 组内的多个路由器中的一台。控制虚拟路由器 IP 地址的 VRRP 路由器称为主路由器，它负责转发数据包到这些虚拟 IP 地址。一旦主路由器不可用，这种选择过程就提供了动态的故障转移机制，这就虚拟允许路由器的 IP 地址可以作为终端主机的默认第一跳路由器。使用 VRRP 的好处是有更高的默认路径的可用性而无需在每个终端主机上配置动态路由或路由发现协议。
* VRRP 包封装在 IP 包中发送，用于 VRRP 组内设备的互相通信比如保持心跳等。它只定义了一种报文即 VRRP 报文，这是一种组播报文，由 Master 路由器定时发出来通告组内的路由器它的存在。
* VRRP 中定义了三种状态模型，初始状态 Initialize，活动状态 Master 和备份状态 Backup，其中只有活动状态的交换机可以为到虚拟IP地址的的转发请求提供服务。

几个概念：

* VIP：VRRP 使用两种类型的 VIP：一种 VIP 参与 VRRP 通告，使用配置项 virtual\_ipaddress 指定；另一种 VIP 不参与 VRRP 通告，使用配置项 virtual\_ipaddress\_excluded 指定。
* 优先级（Priority）：虚拟路由器中VRRP设备的优先级。虚拟路由器根据优先级选举出Master设备和Backup设备。
* 抢占模式：在抢占模式下，如果 Backup 设备的优先级比当前 Master 设备的优先级高，则主动将自己切换成 Master。
* 非抢占模式：在非抢占模式下，只要 Master 设备没有出现故障，Backup 设备即使随后被配置了更高的优先级也不会成为Master设备。
* VRRP协议报文：封装在IP报文中，发送到分配给 VRRP 的 IP 组播地址。在IP报文头中，源地址为发送报文接口的主 IP 地址（不是虚拟IP地址），目的地址是224.0.0.18，TTL是255，协议号是112。目前，VRRP协议包括两个版本：VRRPv2和VRRPv3，VRRPv2仅适用于IPv4网路，VRRPv3适用于IPv4和IPv6两种网络。
* VRRP 节点三种状态：初始状态（Initialize）、活动状态（Master）、备份状态（Backup）。其中，只有处于Master状态的设备才可以转发那些发送到虚拟IP地址的报文。

Master 路由器的选举过程：

VRRP根据优先级来确定虚拟路由器中每台路由器的角色（Master 路由器或 Backup 路由器）。优先级越高，则越有可能成为 Master 路由器。  
初始创建的 VRRP 交换机工作在 Initialize 状态，收到接口 Up 的消息后，若此路由器的优先级小于255，则会先切换至 Backup 状态，待 Master\_Down\_Interval 定时器超时后再切换至 Master 状态。首先切换至 Master 状态的 VRRP 路由器通过 VRRP 通告报文的交互获知虚拟交换机中其他成员的优先级，进行Master的选举：

* + 如果 VRRP 报文中 Master 路由器的优先级高于或等于自己的优先级，则 Backup 路由器保持 Backup 状态；
  + 如果 VRRP 报文中 Master 路由器的优先级低于自己的优先级，采用抢占方式的 Backup 交换机将切换至 Master 状态，采用非抢占方式的Backup 路由器仍保持Backup状态。
  + 如果多个 VRRP 路由器同时切换到 Master 状态，通过 VRRP 通告报文的交互进行协商后，优先级较低的 VRRP 路由器将切换成 Backup 状态，优先级最高的 VRRP 路由器成为最终的 Master设备；优先级相同时，VRRP 路由器上 VRRP 备份组所在接口主 IP 地址较大的成为 Master 设备。
  + 如果创建的VRRP 路由器为IP地址拥有者，收到接口 Up 的消息后，将会直接切换至Master状态。

Master交换机状态的通告：

* Master 路由器周期性地发送 VRRP 通告报文，在 VRRP 备份组中公布其配置信息（优先级等）和工作状况。Backup 路由器通过接收到 VRRP 报文的情况来判断Master 路由器是否工作正常。
  + 当 Master 路由器主动放弃 Master 地位（如 Master 路由器退出备份组）时，会发送优先级为0的通告报文，用来使 Backup 路由器快速切换成 Master 路由器，而不用等到 Master\_Down\_Interval 定时器超时。
  + 当 Master 路由器发生网络故障而不能发送通告报文的时候，Backup 路由器并不能立即知道其工作状况。等到 Master\_Down\_Interval 定时器超时后，才会认为 Master 路由器无法正常工作，从而将状态切换为 Master。

目前，主流的硬件路由器设备都实现了该协议： [这篇文章](http://www1.brocade.com/downloads/documents/html_product_manuals/NOS_411_AG_04/GUID-B645813D-E68A-424E-A1FD-9E5E7560A08B.html) 介绍了 Broadcom 路由器上 VRRP 的配置，[这篇文章](http://blog.chinaunix.net/attachment/attach/20/10/41/20201041201f60ca9386690a5c0fe987bb225dba8f.pdf) 介绍了一个 Cisco 路由器 VRRP 的配置例子，[这篇文章](http://www.h3c.com.cn/Products___Technology/Technology/Dependability/Other_technology/Technology_book/200802/335873_30003_0.htm) 是华为的 VRRP 白皮书。

它的优势：

* 操作简单：它不需要改变组网情况，也不需要在主机上做任何配置，只需要在相关路由器上配置极少的几条命令，就能实现下一跳网关的备份，并且不会给主机带来任何负担。和其他方法比较起来，VRRP更加能够满足用户的需求。
* 简化网络管理：在具有多播或广播能力的局域网（如以太网）中，借助 VRRP 能在某台设备出现故障时仍然提供高可靠的缺省链路，有效避免单一链路发生故障后网络中断的问题，而无需修改动态路由协议、路由发现协议等配置信息，也无需修改主机的默认网关配置。
* 适应性强：VRRP报文封装在IP报文中，支持各种上层协议。
* 网络开销小：VRRP只定义了一种报文——VRRP通告报文，并且只有处于Master状态的路由器可以发送VRRP报文。

其他参考文章：[（1）](http://zhangzhenzz.blog.51cto.com/5117763/965212)[（2）](http://techtutsonline.com/virtual-router-redundancy-protocol/)[（3）](http://bbs.nanjimao.com/thread-790-1-1.html)。更多内容，参见[协议文本](https://www.ietf.org/rfc/rfc3768.txt)，以及[VRRP 技术白皮书 - Huawei](https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0CCIQFjABahUKEwjLkNev7YvHAhVMWSwKHc5cCrs&url=%68%74%74%70%3a%2f%2f%65%2e%68%75%61%77%65%69%2e%63%6f%6d%2f%63%6e%2f%6d%61%72%6b%65%74%69%6e%67%2d%6d%61%74%65%72%69%61%6c%2f%63%6e%2f%70%72%6f%64%75%63%74%73%2f%65%6e%74%65%72%70%72%69%73%65%5f%6e%65%74%77%6f%72%6b%2f%73%77%69%74%63%68%65%73%2f%73%39%37%30%30%2f%68%77%5f%32%30%31%30%35%37&ei=TdC-VYu2K8yysQHOuanYCw&usg=AFQjCNHM9Y9opc-2_Xgzr6mqIRVJLauLhg&bvm=bv.99261572,d.bGg)。

**1.2 Keepalived**

**1.2.2 概念**

[Keepalived](http://www.keepalived.org/)是一个用 C 语言编写的路由软件，其主要目的是向 Linux 系统和基础设施提供简单可靠的工具来实现负载均衡（Load balancing）和高可用（HA）。其中，

* 负载均衡框架基于  Linux Virtual Server (IPVS) 内核模块来提供网络四层的负载均衡，它实现了一组检查器（Checkers）来根据它们的健康状态动态地和自适应地维护和管理被均衡的服务器池。
* 高可用（HA）是使用 VRRP 来实现的，Keepalived 是 VRRP 的一个非常好的开源实现。它是一个基于 VRRP 协议来实现的 WEB 服务高可用方案，可以利用其来避免单点故障。一个 WEB 服务至少会由两台台物理服务器运行 Keepalived，一台为主服务器（MASTER），一台为备份服务器（BACKUP），但是对外表现为一个虚拟 IP，主服务器会发送特定的消息给备份服务器，当备份服务器收不到这个消息的时候，即主服务器宕机的时候，备份服务器就会接管虚拟 IP，继续提供服务，从而保证了高可用性。

**1.2.2 软件架构和功能**

（本部分内容来自 Keepalived 官网）

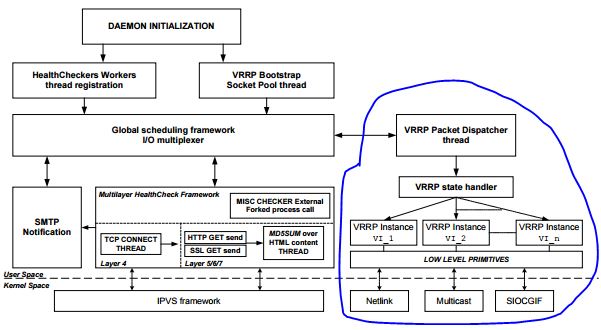
Keepalived 使用三个进程，其中 Watchdog 是控制进程，VRRP Stack and Checkers 是它的两个子进程。

* Watchdog 通过心跳机制来确保子进程处于运行状态。
* Checkers：负责真实服务器的健康检测，用于负载均衡。
* VRRP Stack：实现 VRRP 协议，提供 HA。

主要模块：

* System call : 用于启动用户添加的脚本。它可以被 Checkers 和 VRRP Stack 使用。它向 VRRP 框架提供了一种能力，即在 VRRP 协议状态变化时可以运行用户指定的脚本。
* SMTP : 使得 Keepalive 能够使用邮件来通知管理员。
* NETLINK Interface: 用于 VRRP。Keepalive 通过它来设置或者删除 VIP。

 VRRP Stack 模块：



VRRP Stack 是独立于 LVS的，它可以被单独使用。它主要提供以下功能：

* Failover：实现 VRRP协议的核心功能，提供故障切换能力。
* VRRP Instance synchronization：VRRP Group （组）之间的同步，比如保证两个 VRRP 组直接保持同样状态。
* VRRP 广播：Master 实例通知 Backup 实例。
* System call： 在 VRRP 状态变换时调用外部的脚本或者程序。

它包括以下组件：

* Netlink：提供 VIP 操作。
* Multicast：用于发送 VRRP 协议广播。
* SYSLOG：所有 daemon 的通知消息都会通过该组件写入日志。

用户可以使用配置文件 /etc/keepalived/keepalived.conf 来配置 Keepalived。本部分主要讲讲 keepalived 的概念，使用会在下文谈到。更多内容，参见[官网](http://www.keepalived.org/)。

**2. Neutron Juno 版本中 VRRP 的实现**

    在 [OpenStack HA （1）概述](http://www.cnblogs.com/sammyliu/p/4741967.html) 一文中谈到，Neutron L3 Agent 在 Juno 版本中添加了两种 HA 机制的实现：VRRP 机制（[blueprint](https://blueprints.launchpad.net/neutron/+spec/l3-high-availability)， [Spec](http://specs.openstack.org/openstack/neutron-specs/specs/juno/l3-high-availability.html)，[Wiki](https://wiki.openstack.org/wiki/Neutron/L3_High_Availability_VRRP)）和 DVR 机制。VRRP 机制就是借助实现 VRRP 协议的软件，来保证 Neutron L3 Agent 的高可用性。这需要多方面的实现：

* 创建 HA Router 时，创建多个 router 实例部署到不同物理服务器上的 L3 Agent 中。这个实现需要修改 Neutron schedulers。
* 借助实现 VRRP 的软件，保证多个 L3 Agent 的 HA，即其中一个是 Master，其他是 Backup。由 Master 向虚机提供路由服务。在 Master 故障时，某个 Standby 被选举为新的 Master，接替之前的 Master。目前的 Neutron VRRP 实现使用的是 Keepalived。
* 路由器故障切换时保持已建立的连接。Neutron 会使用 conntrack 来实现该功能。

**2.1 创建 HA Router**

neutron server 上：

（1）类似普通的 router，首先执行 DB 操作在 DB 中创建一个 router。

（2）检查 router 所在的 tenant 中是否存在 HA Network，如果不存在的话，则创建名称为 “HA network tenant”+ tenant\_id 的 network，以及该 network 中的名字为 “HA subnet tenant” +  tenant\_id， cidr 为 cfg.CONF.l3\_ha\_net\_cidr 的 subnet。其余的参数皆为默认或者null。

[复制代码](javascript:void(0);)

s1@controller:~$ neutron net-show bc5501ed-56ee-4d32-974b-02279fb35c32

+---------------------------+----------------------------------------------------+

| Field | Value |

+---------------------------+----------------------------------------------------+

| admin\_state\_up | True |

| id | bc5501ed-56ee-4d32-974b-02279fb35c32 |

| name | HA network tenant 74c8ada23a3449f888d9e19b76d13aab |

| provider:network\_type | gre |

| provider:physical\_network | |

| provider:segmentation\_id | 1 |

| router:external | False |

| shared | False |

| status | ACTIVE |

| subnets | 2ce1ff9d-6889-41cb-8d6b-18e7ccd67a7b |

| tenant\_id | |

+---------------------------+----------------------------------------------------+

s1@controller:~$ neutron subnet-show 2ce1ff9d-6889-41cb-8d6b-18e7ccd67a7b

+-------------------+------------------------------------------------------+

| Field | Value |

+-------------------+------------------------------------------------------+

| allocation\_pools | {"start": "169.254.192.1", "end": "169.254.255.254"} |

| cidr | 169.254.192.0/18 |

| dns\_nameservers | |

| enable\_dhcp | False |

| gateway\_ip | |

| host\_routes | |

| id | 2ce1ff9d-6889-41cb-8d6b-18e7ccd67a7b |

| ip\_version | 4 |

| ipv6\_address\_mode | |

| ipv6\_ra\_mode | |

| name | HA subnet tenant 74c8ada23a3449f888d9e19b76d13aab |

| network\_id | bc5501ed-56ee-4d32-974b-02279fb35c32 |

| tenant\_id | |

+-------------------+------------------------------------------------------+

[复制代码](javascript:void(0);)

注意，当 tenant 最后一个 HA Router 被删除的时候，该 HA network/subnet 不会被自动删除，这里有个 [bug](https://bugs.launchpad.net/neutron/+bug/1367157)。看起来还没有被 fixed，目前只能手动删除。

（3）分配一个可用的 VRRP ID，该 ID 为 1 - 255 之间。

（4）创建与部署的 router 实例相同数量的 port 作为 HA Interfaces，每个 L3 Agent 上使用一个。

[复制代码](javascript:void(0);)

{'tenant\_id': '',

'network\_id': network\_id,

'fixed\_ips': attributes.ATTR\_NOT\_SPECIFIED,

'mac\_address': attributes.ATTR\_NOT\_SPECIFIED,

'admin\_state\_up': True,

'device\_id': router\_id,

'device\_owner': constants.DEVICE\_OWNER\_ROUTER\_HA\_INTF,

'name': constants.HA\_PORT\_NAME % tenant\_id}})

[复制代码](javascript:void(0);)

比如需要部署的 router 的实例数目为2 的情况下，会分配 2 个port，分别用于各 router network namespace 中 keepalived 之间的通信：

s1@controller:~$ neutron port-list | grep 169.254.192  
| 20894a79-b668-44d9-bf42-ef6bc2981960 | HA port tenant 74c8ada23a3449f888d9e19b76d13aab | fa:16:3e:09:29:d4 | {"subnet\_id": "2ce1ff9d-6889-41cb-8d6b-18e7ccd67a7b", "ip\_address": "169.254.192.2"} |  
| a8a3a49f-c175-4ea6-9342-240ef646e99c | HA port tenant 74c8ada23a3449f888d9e19b76d13aab | fa:16:3e:84:24:7f | {"subnet\_id": "2ce1ff9d-6889-41cb-8d6b-18e7ccd67a7b", "ip\_address": "169.254.192.1"} |

（5）schedule router。具体参见 2.1.2 章节.

（6）向待部署 router 实例的每个 L3 Agent 发送 RPC Cast “routers\_updated” 消息

每个 L3 Agent 上：

 （1）循环运行的 routers\_updated 方法将待处理的每个 router 加入到内部 queue 中，等待被 L3 Agent 启动的循环线程处理

（2）处理过程可以参考我[这篇文章](http://www.cnblogs.com/sammyliu/p/4636091.html) 的 3.1 部分

（3）在完成常规的 router 添加操作后，调用 process\_ha\_router\_added 执行 HA 配置部分

（3.1）将 HA Port plug 到 OVS 的 br-int 上，并设置 interface 的三层属性，比如 IP， gateway， route 等。

Bridge br-int

Port "ha-20894a79-b6"

tag: 7

Interface "ha-20894a79-b6"

type: internal

（3.2）准备 keepalived

（3.2.1）生成一个 keepalived 实例。该实例的初始状态为 Backup，非抢占式，VRRP 通告间隔2秒，优先级为 50。

instance = keepalived.KeepalivedInstance('BACKUP', interface\_name, ri.ha\_vr\_id, ha\_port\_cidr, nopreempt=True, advert\_int=self.conf.ha\_vrrp\_advert\_int, priority=ri.ha\_priority)

（3.2.2）通过上面第三步分配的 VRRP ID，定位到 VRRP Group；然后将 3.2.1 中的 keepalived instance 加入到 该 group 。

（3.2.3）将 group 和 instance 保存到 keepalived config 中。

（4）调用 \_add\_keepalived\_notifiers 方法来添加 neutron 需要的 keepalived notifiers。具体见 2.1.3 章节。

（5）调用 process\_router 方法继续处理该 route。对于HA Router，会首次启动新的或者重启已有的 keepalived 进程。

（5.1）生成 keepalived 配置。配置文件的位置可以由配置项 vrrp\_confs  指定，其默认值为 “$state\_path/vrrp”。还可以使用配置项 ha\_vrrp\_auth\_type （默认 'PASS'）  和 ha\_vrrp\_auth\_password （默认 none）指定 VRRP 组播的认证方法，以及 使用配置项 ha\_vrrp\_advert\_int （默认值 2） 指定 VRRP 广播发送的时间间隔。

[复制代码](javascript:void(0);)

vrrp\_sync\_group VG\_1 { # VRRP ID 为 1

group {

VR\_1

}

notify\_master "/var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_master.sh"

notify\_backup "/var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_backup.sh"

notify\_fault "/var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_fault.sh"

}

vrrp\_instance VR\_1 {

state BACKUP

interface ha-20894a79-b6 # 实例绑定的网卡，因为在配置虚拟 IP 的时候必须是在已有的网卡上添加的

virtual\_router\_id 1 # VRRP ID,每个 tenant HA Router 一个 ID

priority 50 # 该 HA Router instance 的 优先级，根据这个选举 Master

nopreempt # 使用非抢占模式

advert\_int 2 # VRRP 广播发送间隔，单位秒

track\_interface { # 跟踪接口，设置额外的监控

ha-20894a79-b6

}

virtual\_ipaddress { # 该 VRRP 的 VIP，格式为 HA Network CIDR 最后一位被 VID 替换，该例子中，CIDR 为 “169.254.0.1/24”， VID 为 1

169.254.0.1/24 dev ha-20894a79-b6

}

}

[复制代码](javascript:void(0);)

keepalived 各种配置说明：

|  |  |  |
| --- | --- | --- |
| **配置** | **说明** | **neutron 用法** |
| notify master/backup/fault | 表示当该 VRRP 节点切换到 master/backup/fault 状态时，要执行的脚本 | 见本文 2.1.3 章节 |
| state | state 指定 instance(Initial) 的初始状态，就是说在配置好后，这台服务器的初始状态就是这里指定的，但这里指定的不算，还是得要通过竞选通过优先级来确定，里如果这里设置为master，但如若他的优先级不及另外一台，那么这台在发送通告时，会发送自己的优先级，另外一台发现优先级不如自己的高，那么他会就回抢占为master | neutron 将所有 keeplived 实例的初始状态都设置为 BACKUP，而且都使用默认优先级 50。这样实际上，一般来说，哪个 L3 Agent 先起来，它的 keepalived 实例就是  Master 了。 |
| interface | VRRP 实例绑定的网卡，在此网卡上添加 VIP 作为 secondary IP | HA Interface |
| dont track primary | 忽略 VRRP 的 interface 错误 | 不使用 |
| track interface | 跟踪接口，设置额外的监控，里面任意一块网卡出现问题，都会进入故障(FAULT)状态，例如，用nginx做均衡器的时候，内网必须正常工作，如果内网出问题了，这个均衡器也就无法运作了，所以必须对内外网同时做健康检查 | neutron 只跟踪 HA Interface，其实它还是可以跟踪 network namespace 的其他 interface，比如 qr，qg 等，是吧？ |
| mcast src ip | 发送多播数据包时的源IP地址，这里注意了，这里实际上就是在那个地址上发送VRRP通告，这个非常重要，一定要选择稳定的网卡端口来发送，这里相当于heartbeat的心跳端口，如果没有设置那么就用默认的绑定的网卡的IP，也就是interface指定的IP地址 | 不设置，默认使用 HA Interface IP |
| garp master delay | 在切换到master状态后，延迟进行免费的ARP(gratuitous ARP)请求 | 未指定 |
| virtual router id | 这里设置VRID，这里非常重要，相同的 VRID 为一个组，他将决定多播的MAC地址 | 创建 HA Router时从 neutron DB 分配，从 1 开始 |
| priority 100 | 设置本节点的优先级，优先级高的为master | 默认为 50 |
| advert int | 检查间隔，默认为1秒 | 配置项 ha\_vrrp\_advert\_int （默认值 2） |
| virtual ipaddress | 这里设置的就是VIP，也就是虚拟 IP 地址，他随着 state 的变化而增加或者删除，当 state 为master的时候就添加，当 state 为 backup 的时候删除，这里主要是有优先级来决定的，和 state 设置的值没有多大关系，这里可以设置多个 IP 地址。[这篇文章](http://serverfault.com/questions/334197/keepalived-for-more-than-20-virtual-addresses) 谈到其数目限制为 20，超过该数目的 IP 可以放到 virtual\_ipaddress\_excluded。 | VIP。格式为 HA Network CIDR 最后一位被 VID 替换， 该例子中，CIDR 为 “169.254.0.1/24”， VID 为 1。 因此，每个 HA Router 的 VIP 是固定的。 |
| virtual routes | 原理和 virtual ipaddress 一样，只不过这里是增加和删除路由 | 不使用 |
| authentication  auth type  auth pass | 认证方式，类型，密码 | 配置项 ha\_vrrp\_auth\_type （默认 'PASS'）  和 ha\_vrrp\_auth\_password （默认 none） |
| nopreempt | 设置不抢占，这里只能设置在state为backup的节点上，而且这个节点的优先级必须别另外的高 | 设置，使用非抢占模式。这样，只要 Master 没坏，即使 Backup 的优先级更高，也不会触发选举 |
| preempt delay | 抢占延迟 | 不使用 |
| virtual\_ipaddress\_excluded | 需要 keepalived 维护但是不会放在VRRP包中传输的IP。也就是说当某个router被选举为master的时候其会的将qg口设置对应的IP。而在变为 backup 时，会将相应的 IP 删除。 | VIP。Neutron 将 浮动 IP，internal/external port IP 等保存在这里。这样，当一个 node 变为 master 的时候，这些 IP 地址会生效；当一个 node 变为 backup 的时候，这些 IP 地址会被删除。 |
| virtual\_routes | 在某个router设置为 master 的时候其会的设置对应的 route。相反的，如果一个router变成了backup，那么上面这些操作会反过来做一遍。 | 每个 external port 对应一条路由，在 VRRP 状态变化时增加或者删除 |

（[资料来源](http://bbs.nanjimao.com/thread-845-1-1.html)。Neutron 所使用的 keepalived config 模板在 [这里](https://wiki.openstack.org/wiki/Neutron/L3_High_Availability_VRRP)）

（5.2）在 route network namespace 运行命令 'keepalived -P -f config\_path -p pid\_file -r pid\_file-vrrp' 启动 keepalived 进程。

root 31510 1 0 10:42 ? 00:00:00 keepalived -P -f /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/keepalived.conf -p /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb.pid -r /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb.pid-vrrp

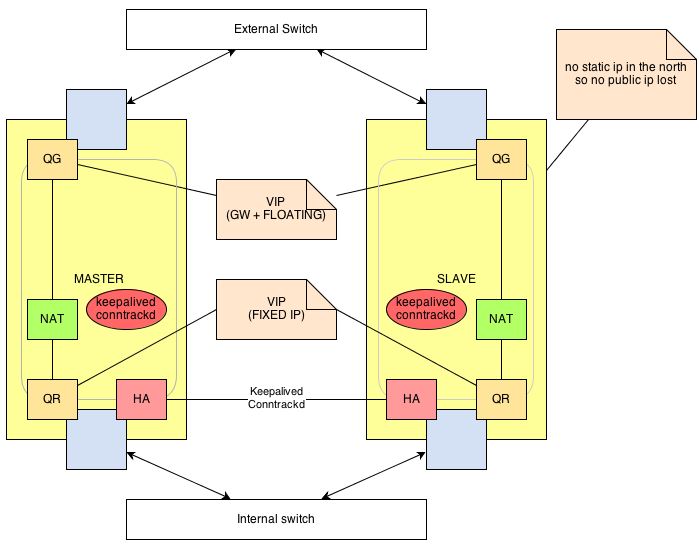
root 31512 31510 0 10:42 ? 00:00:00 keepalived -P -f /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/keepalived.conf -p /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb.pid -r /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb.pid-vrrp

根据上面基础知识部分的介绍，第一个进程（进程号 31510）是 Watchdog 进程，第二个进程（进程号 31512）是 VRRP Stack 进程。

（6） Neutron 中，keepalived 进程启动后，初始状态都是 BACKUP。在它指定时间没收到 Master 的广播后，它自己切换至 Master 状态，然后通过 VRRP 通告报文的交互获知虚拟路由器组中其他成员的优先级，进行 Master 的选举：

* 如果 VRRP 报文中 Master 路由器的优先级高于或等于自己的优先级，则 Backup 交换机保持 Backup 状态。显然，Neutron 中的 keepalived 都是同样的优先级，它们会各自保持自己的 BACKUP 状态。
* 如果 VRRP 报文中 Master 路由器的优先级低于自己的优先级，因为 neutron 中 keepalived 采用非抢占模式，Backup交换机仍保持 Backup 状态。

可见，neutron 中，哪个 L3 Agent namespace 中的 keepalived 进程先启动并先发出 VRRP 通告，则它会成为 Master，直到它 down 掉才会由别的节点接替。在状态变化时，会调用指定的脚本。详细见 2.1.3 部分。状态变化完成后，会达到下面这种效果（Juno 中还没有实现 conntrack）：



（7）启动后状态转换过程

[复制代码](javascript:void(0);)

Aug 1 18:42:32 network Keepalived[31509]: Starting Keepalived v1.2.7 (08/14,2013) #进程启动

Aug 1 18:42:32 network Keepalived[31510]: Starting VRRP child process, pid=31512 # VRRP Stack 进程启动

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Interface queue is empty

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Registering Kernel netlink reflector

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Registering Kernel netlink command channel

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Registering gratuitous ARP shared channel

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Initializing ipvs 2.6

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Opening file '/var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/keepalived.conf'. #导入配置文件

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Configuration is using : 64796 Bytes

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Using LinkWatch kernel netlink reflector...

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: VRRP\_Instance(VR\_1) Entering BACKUP STATE #根据初始配置，首先进入 BACKUP 状态

Aug 1 18:42:32 network Keepalived\_vrrp[31512]: Opening script file /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_backup.sh #脚本调用

Aug 1 18:42:39 network Keepalived\_vrrp[31512]: VRRP\_Instance(VR\_1) Transition to MASTER STATE #第一次选举后进入 Master 状态

Aug 1 18:42:39 network Keepalived\_vrrp[31512]: VRRP\_Group(VG\_1) Syncing instances to MASTER state

Aug 1 18:42:39 network Keepalived\_vrrp[31512]: Opening script file /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_master.sh #脚本调用

Aug 1 18:42:41 network Keepalived\_vrrp[31512]: VRRP\_Instance(VR\_1) Entering MASTER STATE

[复制代码](javascript:void(0);)

（8）VRRP 心跳

下面的 log 能看出来，Master 每隔两秒钟，向 组播地址 224.0.0.18/01:00:5e:00:00:12 发出长度为 54 bytes 的一个心跳：

ip netns exec qrouter-04f6e792-3b79-4b8f-a577-2ad38d33a2bb tcpdump -envi ha-20894a79-b6 -vvv

11:18:10.969378 fa:16:3e:09:29:d4 > 01:00:5e:00:00:12, ethertype IPv4 (0x0800), length 54: (tos 0xc0, ttl 255, id 1066, offset 0, flags [none], proto VRRP (112), length 40)

169.254.192.2 > 224.0.0.18: vrrp 169.254.192.2 > 224.0.0.18: VRRPv2, Advertisement, vrid 1, prio 50, authtype none, intvl 2s, length 20, addrs: 169.254.0.1

11:18:12.971101 fa:16:3e:09:29:d4 > 01:00:5e:00:00:12, ethertype IPv4 (0x0800), length 54: (tos 0xc0, ttl 255, id 1067, offset 0, flags [none], proto VRRP (112), length 40)

169.254.192.2 > 224.0.0.18: vrrp 169.254.192.2 > 224.0.0.18: VRRPv2, Advertisement, vrid 1, prio 50, authtype none, intvl 2s, length 20, addrs: 169.254.0.1

**2.2 HA Router 调度（scheduling）**

Neutron L3 Agent 的调度包括三个层面：

* 普通（Legacy）L3 Agent 调度：在存在多个 L3 Agent 的时候，通过随机或者最小 router 数算法找到一个 L3 Agent，将该 Router 部署在上面。
* 分布式（DVR）L3 Agent 调度：将 Router 分布到所有计算节点和网络节点。
* 高可用（HA） L3 Agent 调度：见下文

    关于 HA L3 Agent 的调度，这部分的实现代码在 [这里](https://review.openstack.org/#/c/66347/)。新的代码修改了已有的 scheduler 算法，使得它们可以支持 HA Router，但是目前不能同时支持 DVR 和 HA Router。主要代码在 /n eutron/scheduler/l3\_agent\_scheduler.py。该实现在 Neutron server 配置文件中添加了如下的配置项：

neutron.conf:

l3\_ha = True #由管理做的全局配置，创建的所有 router 均为 HA Router

max\_l3\_agents\_per\_router = 4 #每个 HA router 最多被部署到若干个 L3 Agent 上

min\_l3\_agents\_per\_router = 2 #每个 HA Router 最少被部署到若干个 L3 Agent 上

该文件：

（1）检查可用的 L3 Agent 数目（candidates）是否超过 conf.min\_l3\_agents\_per\_router 。

（2）确定部署的 router 数目，取 min（candidates，conf.max\_l3\_agents\_per\_router ）。

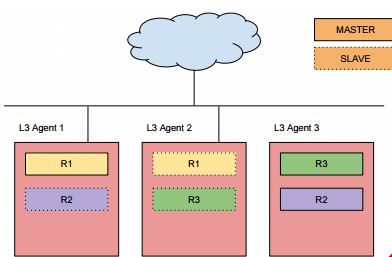
（3）实现两个 scheduler 类：ChanceScheduler - 随机地在可用的 L3 Agent 上部署 Router；LeastRoutersScheduler - 将 router 部署到所管理的 router 数目最少的 L3 Agent 上。它们根据不同的算法选出待部署 router 实例的 L3 Agent。

（4）bind\_router 函数会执行 db 操作，将 router 和 scheduler 选出的 L3 Agent 绑定在一起。比如：

2015-08-01 10:42:21.488 19430 DEBUG neutron.scheduler.l3\_agent\_scheduler [req-827c863f-cb01-4650-a26d-0f176ef84026 None] HA Router 04f6e792-3b79-4b8f-a577-2ad38d33a2bb is scheduled to L3 agent 04c360d0-3066-4f04-9af2-d4ef8586ad2b) bind\_ha\_router\_to\_agents /usr/lib/python2.7/dist-packages/neutron/scheduler/l3\_agent\_scheduler.py:330

2015-08-01 10:42:21.547 19430 DEBUG neutron.scheduler.l3\_agent\_scheduler [req-827c863f-cb01-4650-a26d-0f176ef84026 None] Router 04f6e792-3b79-4b8f-a577-2ad38d33a2bb is scheduled to L3 agent 4705d27c-5008-4828-b619-bbb2114188ba bind\_router /usr/lib/python2.7/dist-packages/neutron/scheduler/l3\_agent\_scheduler.py:226

当确定的部署数量为2 时，会是下面的情形：



**2.3 Keepalived notifier script （ VRRP 状态变化时的通知脚本）**

Neutron 的 KeepalivedNotifierMixin 类实现了 keepalived 在 VRRP 状态变化时的通知方法。Neturon 添加了 VRRP 状态变为 master，backup 和 fault 状态时的 notifier。

* 变更为 master 状态时的 notifier script：

#!/usr/bin/env bash

neutron-ns-metadata-proxy --pid\_file=/var/lib/neutron/external/pids/04f6e792-3b79-4b8f-a577-2ad38d33a2bb.pid --metadata\_proxy\_socket=/var/lib/neutron/metadata\_proxy --router\_id=04f6e792-3b79-4b8f-a577-2ad38d33a2bb --state\_path=/var/lib/neutron --metadata\_port=9697 --debug --verbose --log-file=neutron-ns-metadata-proxy-04f6e792-3b79-4b8f-a577-2ad38d33a2bb.log --log-dir=/var/log/neutron

echo -n master > /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/state

第一个命令，在 router network namespace 中启动 neutron-ns-metadata-proxy，这样使用该 router 的虚机就可以通过该 proxy 访问 metadata service 了。关于 metadata service， proxy 和 agent，可以参考文章 [（1）](http://blog.csdn.net/lynn_kong/article/details/9115033)[（2）](http://www.pubyun.com/blog/openstack/%E4%BB%80%E4%B9%88%E6%98%AFopenstack%E7%9A%84-metadata/)[（3）](http://techbackground.blogspot.com/2013/06/metadata-via-quantum-router.html)。

第二个命令，设置 state 文件内容为 “master”。 Keepalived 不支持直接查询 VRRP 状态，因此，只能通过 state 变化时被调用的脚本文件来保存其状态。

* 变更为 backup 状态的 notifier script：

#!/usr/bin/env bash

kill -9 $(cat /var/lib/neutron/external/pids/04f6e792-3b79-4b8f-a577-2ad38d33a2bb.pid)

echo -n backup > /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/state

第一个命令，在 router network namespace 中杀掉 neutron-ns-metadata-proxy 进程，这样使用该 router 的虚机就无法通过该 proxy 访问 metadata service 了。

第二个命令，在 state 文件中记录 VRRP state 为 “backup”。

#!/usr/bin/env bash

kill -9 $(cat /var/lib/neutron/external/pids/04f6e792-3b79-4b8f-a577-2ad38d33a2bb.pid)

echo -n fault > /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/state

因此，第一次 VRRP 选举完成后，即有一个 Master router 被选举出来。这时候，各节点上不同的 script 会执行不同的动作：

* Master router 的节点上：启动 route namespace 对应的 neutron-ns-metadata-proxy 进程，保存 “master” 至 state 文件，设置 VIP（包括配置文件中 virtual ipaddress 指定的 IP 地址和 virtual\_ipaddress\_excluded 部分的 IP 地址）和 virtual\_routes 部分的 route。
* Slave router 的节点上：杀掉 route namespace 对应的 neutron-ns-metadata-proxy 进程，保存 “backup” 至 state 文件，清除 VIP 和 virtual\_routes 部分的 route。

**2.4 依次添加 gateway，subnet，虚机 和 浮动 IP**

    这些操作都是对已有 HA router 的更新。 当 HA Router 的信息发生改变的时候，执行 DB 操作后，Neutron Server 会通知部署有 router 实例的 L3 Agent，然后它会去获取到更新后的信息，重新生成新的 keepalived 配置文件，然后通知 keepalived 去使用新的配置文件。

**2.4.1 添加 gateway**

    keepalived 的配置文件的 virtual\_ipaddress\_excluded 部分会增加 gateway 的 interface IP：192.168.1.113/24 dev qg-f19f97bc-e5 作为一个 VIP。这个 VIP 地址会在 Master node 被设置，在 Backup node 上不会被设置。

master：

36: qg-f19f97bc-e5: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:14:be:06 brd ff:ff:ff:ff:ff:ff

inet 192.168.1.113/24 scope global qg-f19f97bc-e5

valid\_lft forever preferred\_lft forever

slave：

61: qg-f19f97bc-e5: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:14:be:06 brd ff:ff:ff:ff:ff:ff

**2.4.2 添加 subnet interface**

    同样地，keepalived 的配置文件的 virtual\_ipaddress\_excluded 部分会增加 subnet的 gateway IP：10.0.10.1/24 dev qr-73641a84-72 做为一个新的 VIP。这个 VIP 会在 Master node 被设置，在 Backup node 上不会被设置。

master：

37: qr-73641a84-72: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:86:dd:53 brd ff:ff:ff:ff:ff:ff

inet 10.0.10.1/24 scope global qr-73641a84-72

valid\_lft forever preferred\_lft forever

bakckup：

62: qr-73641a84-72: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:86:dd:53 brd ff:ff:ff:ff:ff:ff

**2.4.3 添加一个虚机到 10.0.10.1/24 网络**

    虚机的默认网关为 10.0.10.1，这正是 Master node 上的 qr-73641a84-72 interface 上设置的 VIP 。因此，虚机发送到不是本机所在网络的网络包都会经过本 interface 进入 Master 路由器。

**2.4.4 向该 IP 添加一个 浮动 IP 192.168.1.112**

    同样地，keepalived 的配置文件的 virtual\_ipaddress\_excluded 部分会增加该 浮动 IP 和它所在的 external port：192.168.1.112/32 dev qg-f19f97bc-e5 作为一个新的 VIP。这个 VIP 会在 Master node 被设置，在 Backup node 上不会被设置。

该 VIP 被加到了 Master router 的 external port 上：

36: qg-f19f97bc-e5: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:14:be:06 brd ff:ff:ff:ff:ff:ff

inet 192.168.1.113/24 scope global qg-f19f97bc-e5

valid\_lft forever preferred\_lft forever

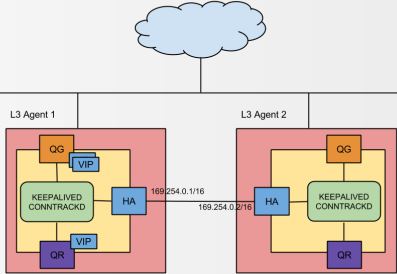
inet 192.168.1.112/32 scope global qg-f19f97bc-e5

valid\_lft forever preferred\_lft forever

同样，在 Slave router 的 exernal port 上没有添加。

但是，每一次 router 实例都更新的时候，Master router 和 Backup router 的 iptables 规则保持了一致的更新。 从以上过程可以看出，在某一时刻，只有 Master router 的 interface 在接受虚机发来的网络包，其它 VRRP router 只是在 standby。

至此，Master 和 Backup L3 Agent namespace 的效果为：



**2.5 HA Router 故障切换**

  非抢占模式下，会触发 keepalived  VRRP failover 的情况包括：

* 当 slave 节点在规定的时间内没有收到 master 的心跳：比如 一个 L3 Agent 上的 keepalived 进程死了，或者 HA Device down 了，或者 HA network down 了等。除此以外的其它情况，都需要使用 workaround 来触发故障切换：比如使用 peacemaker 监视 L3 Agent，一旦发现它 down 了，立刻将 HA Device down 掉。
* 当 master 节点上的 externa gateway 在规定的时间内无法访问时：这里有个 [bug](https://bugs.launchpad.net/neutron/+bug/1365461)报出来了，到目前为止还没有被解决。

    为了验证 failover，把 master node 上的  HA Device down 掉。

ip netns exec qrouter-04f6e792-3b79-4b8f-a577-2ad38d33a2bb ifconfig ha-20894a79-b6 down

这时候，该节点的 VRRP 状态变为 fault。

Aug 1 15:26:58 network Keepalived\_vrrp[4415]: Kernel is reporting: interface ha-20894a79-b6 DOWN

Aug 1 15:26:58 network Keepalived\_vrrp[4415]: Kernel is reporting: interface ha-20894a79-b6 DOWN

Aug 1 15:26:58 network Keepalived\_vrrp[4415]: VRRP\_Instance(VR\_1) Entering FAULT STATE

Aug 1 15:26:58 network Keepalived\_vrrp[4415]: VRRP\_Instance(VR\_1) Now in FAULT state

Aug 1 15:26:58 network Keepalived\_vrrp[4415]: VRRP\_Group(VG\_1) Syncing instances to FAULT state

Aug 1 15:26:58 network Keepalived\_vrrp[4415]: Opening script file /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_fault.sh

    原 backup node 上的 keepalived 因为在规定时间内收不到 VRRP 通告，主动发起选举过程，将自己变为 Master 状态，触发 notify\_master.sh 被调用。这时候，该节点上的 router namespace 的 qr，gg 和 ha interface 的 VIP 全部被配置了，而且 route 规则也增加了。它已经接替原 master 节点向虚机提供路由服务。

[复制代码](javascript:void(0);)

60: ha-a8a3a49f-c1: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:84:24:7f brd ff:ff:ff:ff:ff:ff

inet 169.254.192.1/18 brd 169.254.255.255 scope global ha-a8a3a49f-c1

valid\_lft forever preferred\_lft forever

inet 169.254.0.1/24 scope global ha-a8a3a49f-c1

valid\_lft forever preferred\_lft forever

inet6 fe80::f816:3eff:fe84:247f/64 scope link

valid\_lft forever preferred\_lft forever

61: qg-f19f97bc-e5: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:14:be:06 brd ff:ff:ff:ff:ff:ff

inet 192.168.1.112/32 scope global qg-f19f97bc-e5

valid\_lft forever preferred\_lft forever

inet 192.168.1.113/24 scope global qg-f19f97bc-e5

valid\_lft forever preferred\_lft forever

inet6 fe80::f816:3eff:fe14:be06/64 scope link

valid\_lft forever preferred\_lft forever

62: qr-73641a84-72: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UNKNOWN group default

link/ether fa:16:3e:86:dd:53 brd ff:ff:ff:ff:ff:ff

inet 10.0.10.1/24 scope global qr-73641a84-72

valid\_lft forever preferred\_lft forever

inet6 fe80::f816:3eff:fe86:dd53/64 scope link

valid\_lft forever preferred\_lft forever

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

root@network:/home/s1# ip netns exec qrouter-04f6e792-3b79-4b8f-a577-2ad38d33a2bb route -n

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

0.0.0.0 192.168.1.1 0.0.0.0 UG 0 0 0 qg-f19f97bc-e5

10.0.10.0 0.0.0.0 255.255.255.0 U 0 0 0 qr-73641a84-72

169.254.0.0 0.0.0.0 255.255.255.0 U 0 0 0 ha-20894a79-b6

169.254.192.0 0.0.0.0 255.255.192.0 U 0 0 0 ha-20894a79-b6

192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 qg-f19f97bc-e5

[复制代码](javascript:void(0);)

从外网 ping 虚机看， failover 过程中，网络有若干秒的中断，而且有建立起来的连接断了：

64 bytes from 192.168.1.112: icmp\_seq=41 ttl=63 time=2.53 ms  
64 bytes from 192.168.1.112: icmp\_seq=42 ttl=63 time=2.23 ms  
64 bytes from 192.168.1.112: icmp\_seq=50 ttl=63 time=5.35 ms  
64 bytes from 192.168.1.112: icmp\_seq=51 ttl=63 time=2.11 ms

这个问题会在后续版本会使用 conntrackd 来解决。

在 HA Device 重新 UP 后，VRRP 状态从  fault 变为 Backup 并一直保持该状态：

Aug 1 15:39:37 network Keepalived\_vrrp[4415]: Kernel is reporting: Group(VG\_1) UP

Aug 1 15:39:37 network Keepalived\_vrrp[4415]: VRRP\_Group(VG\_1) Leaving FAULT state

Aug 1 15:39:37 network Keepalived\_vrrp[4415]: VRRP\_Instance(VR\_1) Entering BACKUP STATE

Aug 1 15:39:37 network Keepalived\_vrrp[4415]: VRRP\_Group(VG\_1) Syncing instances to BACKUP state

Aug 1 15:39:37 network Keepalived\_vrrp[4415]: Opening script file /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_backup.sh

所有 port 的 VIP 和 route 规则都被删除了。

再将 master namespace 上的 HA Device down 掉，backup namespace 中的 keepalive 重新变为 Master：

Aug 1 15:41:58 network Keepalived\_vrrp[4415]: VRRP\_Instance(VR\_1) Transition to MASTER STATE

Aug 1 15:41:58 network Keepalived\_vrrp[4415]: VRRP\_Group(VG\_1) Syncing instances to MASTER state

Aug 1 15:41:58 network Keepalived\_vrrp[4415]: Opening script file /var/lib/neutron/ha\_confs/04f6e792-3b79-4b8f-a577-2ad38d33a2bb/notify\_master.sh

Aug 1 15:42:00 network Keepalived\_vrrp[4415]: VRRP\_Instance(VR\_1) Entering MASTER STATE

所有 port 的 VIP 和 route 规则都被设置了。

**3. Juno VRRP 小结和后续版本的改进**

**3.1 Juno VRRP 实现小结**

1. Juno VRRP 使用 Keepalived，在每个 HA Route 的实例的 namespace 中启动一个 keepalived 进程，它采用非抢占式模式。
2. keepalived 进程只监控 HA Device，在其不可用时触发选举机制。
3. neutron 依赖 keepalived 设置或者清除 VIP （包括 VRRP VIP 和浮动 IP，internal/external port IP）
4. neutron 依赖 keepalived 设置或者清除 virtual\_routes 部分的 route 规则
5. neutron 在 keepalived VRRP 状态变化时保存 VRRP 的状态到本地文件
6. 所有 HA Route 实例的 namespace 中，只有 master namespace 提供 port 上的 IP 实际向网络内的虚机提供路由服务。neutron 依赖 keepalived 在 failover 时发送无用 ARP 来更新指定网络内机器的 ARP 表。
7. Juno 版本中的实现还不完善，比如只能监控 HA Device，不能监控别的 port，包括 internal 和 external 端口，以及 external link 等；还未实现切换时保留 TCP 连接的状态。

**3.2 后续版本中的持续改进**

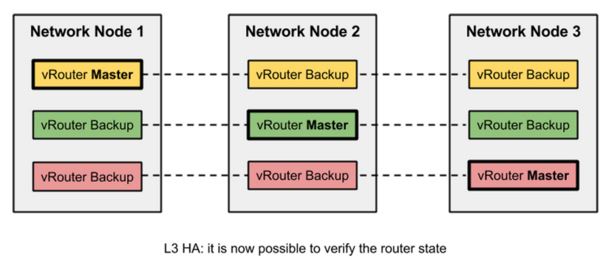
    Juno 版本中增加 VRRP 的实现实现后，有很多地方需要改进，有大量的 blueprint 被开出来，但是大都进展缓慢。比较有意思的有几个：

1. 这个 [blueprint](https://review.openstack.org/#/c/80332/)增加 conntracked 来支持 keepalived failover 后保存已建立的 TCP 连接。目前还没完成。
2. 通过监视  HA Device 的  ip 来监视 keepalived 节点的状态
3. 这个 [blueprint](https://blueprints.launchpad.net/neutron/+spec/report-ha-router-master)增加 neutron API 来 report keepalived master。已合并到 M 版本。
4. 这个 [blueprint](https://blueprints.launchpad.net/neutron/+spec/dvr-support-ha) 同时支持 DVR 和 HR，但是还未开工。
5. 这个 [bug](https://bugs.launchpad.net/neutron/+bug/1365461)打算增加 checker 来监视 external gateway。未完成。

这也能看出，Neutron 在基础部分（L2，L3）的实现比较扎实，但是在高级功能的实现上，包括 VRRP，DVR，VPN/FW/LBaas 等，还是不够扎实，想应用到生成环境中还需要做大量的工作。

**4. Kilo 版本更新 （//2015/11/27 更新）**

Neutron Kilo 版本中，新增了 *neutron l3-agent-list-hosting-router <router\_id>*命令来查看 master l3-agent 在哪个网络节点上。



root@hkg023:~# neutron l3-agent-list-hosting-router 03cb51be-3558-4253-8062-db177679e141

+--------------------------------------+-------------------+----------------+-------+----------+

| id | host | admin\_state\_up | alive | ha\_state |

+--------------------------------------+-------------------+----------------+-------+----------+

| 39b13aea-45a9-4336-afb4-c555b488e92d | hkg02kvm | True | :-) | |

+--------------------------------------+-------------------+----------------+-------+----------+

参考资料：

<https://wiki.openstack.org/wiki/Neutron/L3_High_Availability_VRRP>

<https://review.openstack.org/#/q/topic:bp/l3-high-availability,n,z>

<http://bbs.nanjimao.com/thread-845-1-1.html>

<http://bingotree.cn/?p=704>

<http://www.keepalived.org/documentation.html>

分类: [Neutron](http://www.cnblogs.com/sammyliu/category/677824.html),[OpenStack](http://www.cnblogs.com/sammyliu/category/636967.html),[高可用（HA）](http://www.cnblogs.com/sammyliu/category/724243.html)