[**Neutron 理解（5）：Neutron 是如何向 Nova 虚机分配固定IP地址的 （How Neutron Allocates Fixed IPs to Nova Instance）**](http://www.cnblogs.com/sammyliu/p/4419195.html)

学习 Neutron 系列文章：

（1）[Neutron 所实现的虚拟化网络](http://www.cnblogs.com/sammyliu/p/4622563.html%20)

（2）[Neutron OpenvSwitch + VLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4626419.html%20)

（3）[Neutron OpenvSwitch + GRE/VxLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4627230.html%20)

（4）[Neutron OVS OpenFlow 流表 和 L2 Population](http://www.cnblogs.com/sammyliu/p/4633814.html%20)

（5）[Neutron DHCP Agent](http://www.cnblogs.com/sammyliu/p/4419195.html)

（6）[Neutron L3 Agent](http://www.cnblogs.com/sammyliu/p/4636091.html%20)

（7）[Neutron LBaas](http://www.cnblogs.com/sammyliu/p/4656176.html%20)

（8）[Neutron Security Group](http://www.cnblogs.com/sammyliu/p/4658746.html%20)

（9）[Neutron FWaas 和 Nova Security Group](http://www.cnblogs.com/sammyliu/p/4675991.html)

（10）[Neutron VPNaas](http://www.cnblogs.com/sammyliu/p/4677386.html%20)

（11）[Neutron DVR](http://www.cnblogs.com/sammyliu/p/4713562.html)

（12）[Neutron VRRP](http://www.cnblogs.com/sammyliu/p/4692081.html)

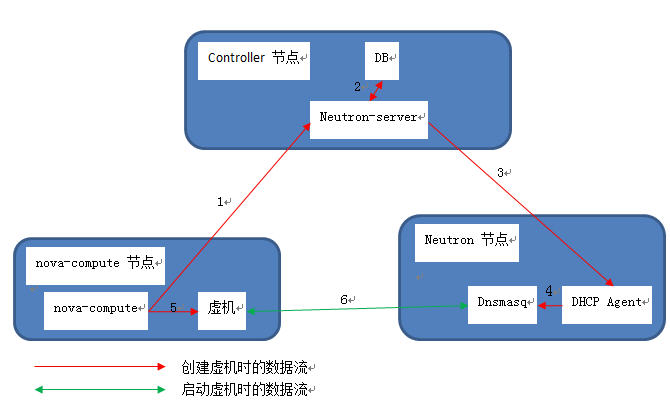
（13）[High Availability （HA）](http://www.cnblogs.com/sammyliu/p/4741967.html)

Nova 虚机获取固定IP （Fixed IP）主要分为两个步骤：

（1）在创建虚机过程中，Neutron 随机生成 MAC 和 从配置数据中分配一个固定IP 地址，并保存到 Dnsmasq 的 hosts 文件中，让 Dnsmasq 做好准备。

（2）虚机在启动时向 Dnsmasq 获取 IP 地址

本文将分析该过程。整个过程涉及不同节点的几个模块：



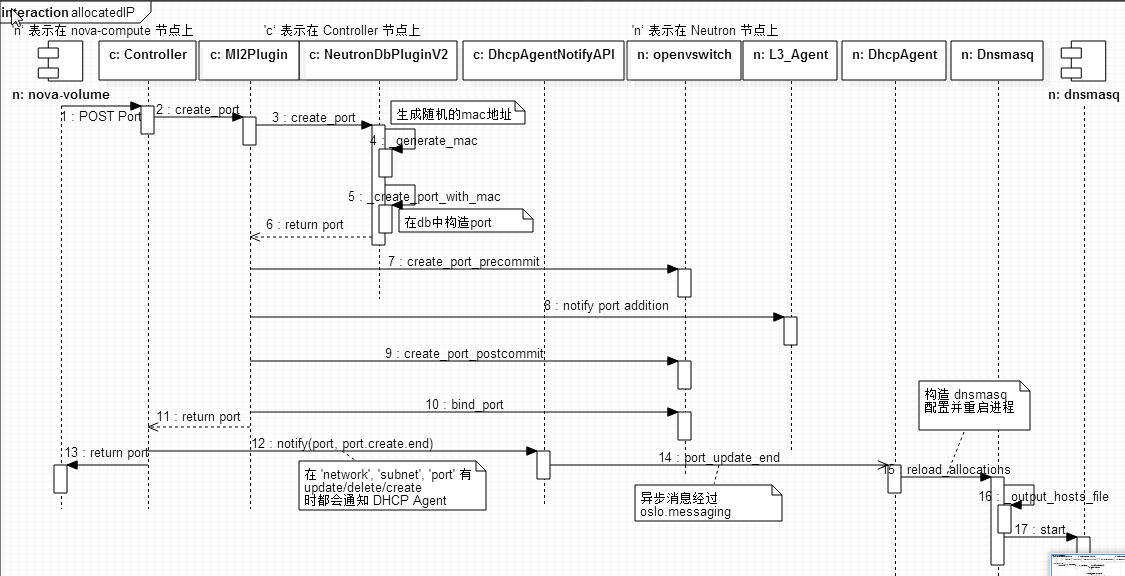
下面将具体分析该过程。

**1. 创建虚机时的数据流**

Nova-compute 在创建虚机时，需要 Neutron 所做的主要事情之一就是分配一个 MAC 和 一个或者多个固定 IP 地址，该过程从 Nova-compute 向 Neutron 申请 Port 开始：

REQ: curl -i http://controller:9696/v2.0/ports.json -X POST -H "X-Auth-Token: ...=" -H "User-Agent: python-neutronclient" -d '{"port": {"binding:host\_id": "compute1", "admin\_state\_up": true, "network\_id": "0a4cd030-d951-401a-8202-937b788bea43", "tenant\_id": "43f66bb82e684bbe9eb9ef6892bd7fd6", "device\_owner": "compute:nova", "security\_groups": ["8c0dc337-0a6d-4ad7-94bf-a400ee32b2ac"], "device\_id": "8671c14e-9ee4-4338-bcc5-8a5f0ea6e1d5"}}'

Controller 节点上的 Neutron Server 接到该请求后，会开始下面的过程：



步骤 2 ~ 6：Neutron Server 生成 MAC 和 IP。 其中 MAC 是由任意数组成的；Fixed IP 是从保存在数据库中的管理员配置的网络和地址数据生成的。

步骤 7 ~ 10： 调用 L3 Agent 和 OVS 执行一些操作。

步骤 12 ~ 14：通过 AMQP 的 cast 消息给 Neutron 节点上的 DHCP Agent，告诉它 Port 创建结束以及 新分配的 Port 的数据。

步骤 13：返回Port 给 nova-compute，数据示例如下：

{"port": {"status": "DOWN", "binding:host\_id": "compute1", "allowed\_address\_pairs": [], "extra\_dhcp\_opts": [], "device\_owner": "compute:nova", "binding:profile": {}, "fixed\_ips": [{"subnet\_id": "5598bdf9-2de4-4a4e-9054-2070102e0f1f", "ip\_address": "10.0.0.132"}], "id": "a9ab8ebf-a0b4-4599-867c-95518f069a10", "security\_groups": ["8c0dc337-0a6d-4ad7-94bf-a400ee32b2ac"], "device\_id": "8671c14e-9ee4-4338-bcc5-8a5f0ea6e1d5", "name": "", "admin\_state\_up": true, "network\_id": "0a4cd030-d951-401a-8202-937b788bea43", "tenant\_id": "43f66bb82e684bbe9eb9ef6892bd7fd6", "binding:vif\_details": {"port\_filter": true, "ovs\_hybrid\_plug": true}, "binding:vnic\_type": "normal", "binding:vif\_type": "ovs", "mac\_address": "fa:16:3e:f2:e0:23"}

步骤 15：Neturon 节点上的 DHCP Agent 根据接收到的 Port 创建完成通知，重新生成 Dnsmasq 的 hosts 文件，然后让 Dnsmasq 重新加载该文件。该 hosts 文件的示例如下：

root@network:/home/s1# cat /var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/host  
fa:16:3e:f2:e0:23,host-10-0-0-132.openstacklocal,10.0.0.132 #刚创建的 Port 对应的项，包括 MAC 地址、host FQDN、IP 地址

fa:16:3e:9d:e9:11,host-10-0-0-135.openstacklocal,10.0.0.135fa:16:3e:45:14:34,host-10-0-0-1.openstacklocal,10.0.0.1

fa:16:3e:9d:e9:17,host-10-0-0-141.openstacklocal,10.0.0.141

fa:16:3e:9d:e9:18,host-10-0-0-142.openstacklocal,10.0.0.142

Nova 拿到 Port 的数据后，会写入虚机的 libvirt.xml 文件。下面的虚机配了两块 NIC，所有它有两个 interface 块：

[复制代码](javascript:void(0);)

<devices>

<disk type="file" device="disk">

<driver name="qemu" type="qcow2" cache="none"/>

<source file="/var/lib/nova/instances/199f9e78-96aa-4d31-b250-6626b88723f5/disk"/>

<target bus="virtio" dev="vda"/>

</disk>

<interface type="bridge">

<mac address="fa:16:3e:0f:13:7e"/>

<model type="virtio"/>

<driver name="qemu"/>

<source bridge="qbrf3aeba46-04"/>

<target dev="tapf3aeba46-04"/>

</interface>

<interface type="bridge">

<mac address="fa:16:3e:a2:47:46"/>

<model type="virtio"/>

<driver name="qemu"/>

<source bridge="qbr04ab89a9-10"/>

<target dev="tap04ab89a9-10"/>

</interface>

</devices>

[复制代码](javascript:void(0);)

关于该过程，详细的代码说明可以看 [这里](http://www.cnblogs.com/feisky/p/3848889.html)。

**2. Neutron 中的网络概念和 DHCP Agent**

**2.1 Neutron 网络概念**

* **Network**：一个具有 tenant 隔离性的虚拟 2 层网络，它使用一个虚拟交换机。除了用于与外网连接的网络外，其它的 network 都只属于创建它的 tenant。 虚机在创建时必须选择至少一个 network，也可以选择使用多块网卡和多个 network 连接。
* **Subnet**：表示一个 IPv4 或者 IPv6 地址区间，该区间内的地址可以被分配给虚机。 一个 Subnet 可以使用一个 DHCP Server 用于分配指定区间内的 IP 地址给虚机。
* **Port**：表示给定 network 上的一个虚拟交换端口，一个 Port 和虚机的一个网卡直接连接。
* **Router**: 在 subnet IP 段之间转发 IP 包，使用一个 external gate 和外网连接，以及连接若干个 subnet。

**关系：**

（1）tenant ---- 1:n ----- network ------- 1：n ------- subnet （一个 tenant 可以拥有多个 network，一个 network 可以包含多个 subnet）

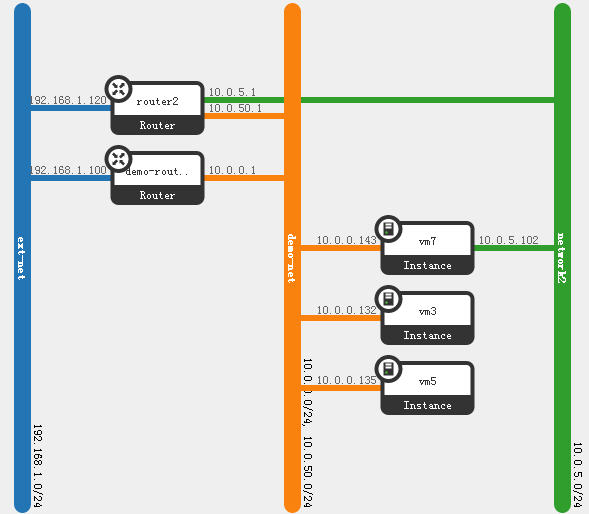
（2）network ------- 1: n ------- port ------ 1:1 --- subnet（一个network 可以有多个 port， 每个 port 连接一个 subnet）（若创建虚机时指定的是 net-id，那么虚机将随机地从该 network 包含的 subnet 中分配 IP）

（3）VM ----- 1 : n ---- NIC ----- 1:1 --- port（一个 VM 可以有多个 NIC，每个 NIC 连接一个 port）（可以在创建虚机时指定一个或者多个 port）

（4）Tenant  ----- 1 : n ---- Router ----- 1 : n ------ subnet/ext-network （一个 tenant 可以拥有多个 router，每个 router 在 Neutron network 节点上使用一个 Linux network namespace，其 ID 就是 neutron router-list 得到的 router 的 ID； 一个 router 连接一个通向外网的 gateway 和多个该 tenant 的 subnet）

（5）network ---- 1 : 1 ---- Dnamasq  ----- 1: n ----- subnet （一个 network 有一个 Dnsmasq 进程，该进程为多个启动了 DHCP 的 subnet 服务，分配它们拥有的 IP 给虚机）

示例：



综上：

Neutron 向 OpenStack 提供一个虚拟的大二层网络。在使用 Open vSwitch 时，它类似于一个跨 Neutron 节点和 Nova-compute 节点的虚拟交换机。Neutron 中的 network 是这个大二层网络中使用 VLAN Tag 分割的子网，而 Subnet 是 network 中一段 IP 地址的集合，Port 是 OVS 交换机上的一个端口（上图中步骤7、9、10就是在 OVS 上配置 Port），它连接虚机的 NIC，使用来自 subnet 的 IP 地址。network 具有租户隔离性，它包含一个自己的 DHCP Agent 来将它所包含的 subnet 的 IP 分给虚机的NIC；它还拥有一个自己的 L3 Agent 来提供 subnet 之间的 IP 包路由以及 subnet 和 external network 之间 IP 包的路由。可以参考我的[另一篇文章](http://www.cnblogs.com/sammyliu/p/4204190.html)中的示意图。

**2.2 Neutron DHCP Agent**

该 Agent 用户提供 DHCP 服务。它：

（1）在其所在的节点上的 /var/lib/neutron/dhcp/ 目录下，为每个 **network** 创建了一个子目录，目录名即 network id；每个子目录下，包含若干文件，比如 host 文件。比如：

root@network:/var/lib/neutron/dhcp# ls

0a4cd030-d951-401a-8202-937b788bea43 d04a0a06-7206-4d05-9432-3443843bc199

c359d42b-2a75-4764-bba9-be76b45f64d8 d24963da-5221-481e-adf5-fe033d6e0b4e

（2）为每个 **network** 启动一个 Dnsmasq 进程，该进程为该 network 中启动了 DHCP 的 **subnet** 提供服务。每一个 subnet 对应于一个 dhcp-range, 从 tag0 开始，一次 tag1，tag2....

nobody 7087 1 0 22:57 ? 00:00:00 dnsmasq --no-hosts --no-resolv --strict-order --bind-interfaces --interface=tap6356d532-32 --except-interface=lo --pid-file=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/pid --dhcp-hostsfile=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/host --addn-hosts=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/addn\_hosts --dhcp-optsfile=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/opts --leasefile-ro --dhcp-range=set:tag0,10.0.0.0,static,86400s --dhcp-range=set:tag1,10.0.50.0,static,86400s --dhcp-lease-max=512 --conf-file=/etc/neutron/dnsmasq.conf --domain=openstacklocal

关于 set::tag<#>，官方的解释是 The optional **set:<tag>**sets an alphanumeric label which marks this network so that dhcp options may be specified on a per-network basis. When it is prefixed with 'tag:' instead, then its meaning changes from setting a tag to matching it. Only one tag may be set, but more than one tag may be matched. 不过还不理解它的意思.....

（3）在 Port 或者 Subnet 有变化时，该 Dnsmasq 进程会重新读取新的配置，包括新的 host 文件，新的 dhcp-range 等。

**3. 关于 Dnsmasq**

**3.1 Dnsmasq 进程**

Dnsmasq 是被 Neutron 用来提供 DHCP 和 DNS 服务的一个开源程序。它提供 DNS 缓存和 DHCP 服务功能。作为域名解析服务器(DNS)，dnsmasq可以通过缓存 DNS 请求来提高对访问过的网址的连接速度。作为DHCP 服务器，[dnsmasq](https://www.archlinux.org/packages/?name=dnsmasq) 可以为局域网电脑提供内网ip地址和路由。DNS和DHCP两个功能可以同时或分别单独实现。dnsmasq轻量且易配置，适用于主机较少的网络。这里是它的 [man page](http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html)。这里是[中文wiki page](https://wiki.archlinux.org/index.php/Dnsmasq_%28%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87%29)。

Neutron 为每个 network 启动一个 Dnsmasq 进程，比如：

[复制代码](javascript:void(0);)

root@network:/home/s1# ps -ef | grep dnsmasq

nobody 1198 1 0 18:51 ? 00:00:00 dnsmasq --no-hosts --no-resolv --strict-order --bind-interfaces --interface=tap8dfd0bd8-45 --except-interface=lo --pid-file=/var/lib/neutron/dhcp/d04a0a06-7206-4d05-9432-3443843bc199/pid --dhcp-hostsfile=/var/lib/neutron/dhcp/d04a0a06-7206-4d05-9432-3443843bc199/host --addn-hosts=/var/lib/neutron/dhcp/d04a0a06-7206-4d05-9432-3443843bc199/addn\_hosts --dhcp-optsfile=/var/lib/neutron/dhcp/d04a0a06-7206-4d05-9432-3443843bc199/opts --leasefile-ro --dhcp-range=set:tag0,10.0.11.0,static,86400s --dhcp-lease-max=256 --conf-file=/etc/neutron/dnsmasq.conf --domain=openstacklocal

nobody 1201 1 0 18:51 ? 00:00:00 dnsmasq --no-hosts --no-resolv --strict-order --bind-interfaces --interface=tap6356d532-32 --except-interface=lo --pid-file=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/pid --dhcp-hostsfile=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/host --addn-hosts=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/addn\_hosts --dhcp-optsfile=/var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/opts --leasefile-ro --dhcp-range=set:tag0,10.0.0.0,static,86400s --dhcp-lease-max=256 --conf-file=/etc/neutron/dnsmasq.conf --domain=openstacklocal

[复制代码](javascript:void(0);)

用到的几个参数的意义如下：

--no-hosts:  Don't read the hostnames in /etc/hosts.  
--bind-interfaces:  Setting this option also enables multiple instances of dnsmasq which provide DHCP service to run in the same machine.  
--interface: Listen only on the specified interface(s). 在指定的 interface 上监听 DHCP 请求。  
--dhcp-hostsfile: Read DHCP host information from the specified file. The advantage of storing DHCP host information in this file is that it can be changed without re-starting dnsmasq: the file will be re-read when dnsmasq receives SIGHUP. DHCP host 文件，dnsmasq 收到 SIGHUP 后会重新读入。没看到 Neutron code 发出该信息，而似乎每次都重启dnsmasq 进程。

--addn-hosts: Additional hosts file. The file will be re-read when dnsmasq receives SIGHUP.

--dhcp-optsfile: Read DHCP option information from the specified file when dnsmasq receiving SIGHUP.

--dhcp-range: Enable the DHCP server. This option may be repeated, with different addresses, to enable DHCP service to more than one network. dnsmasq 默认关闭DHCP功能，该选项开启该功能。每个开启了 DHCP 的 subnet 拥有一个该项。  
--dhcp-lease-max: Limits dnsmasq to the specified maximum number of DHCP leases to prevent DoS attacks from hosts.  
--domain: Specifies DNS domains for the DHCP server.

这些数据皆由数据库中数据计算得出。

**3.2 Dnsmasq log**

dnsmasq 默认地将日志写到 /var/log/syslog 中。可以在 Neutron 节点上做如下配置，使得它使用别的log 文件以便调试：

（1）创建文件  /etc/neutron/dnsmasq.conf，在其中添加

log-facility = /var/log/neutron/dnsmasq.log

log-dhcp

（2）添加下面行到 /etc/neutron/dhcp\_agent.ini:

dnsmasq\_config\_file = /etc/neutron/dnsmasq.conf

（3）运行下面命令重启 Neutron DHCP agent：

sudo service neutron-dhcp-agent restart

**4. 虚机启动时向 Dnsmasq 申请固定 IP**

经过以上步骤，Dnsmasq 准备好相应虚机的IP 申请请求了，它：准备好了 host 文件，里面有每个虚机的 MAC 地址 和 IP 对照表；绑定了 interface，可以收到请求；启动好了进程，可以为指定的 subnet 服务。

获取 IP 的过程如下：

（1）虚机 VM\_1 开机，发出 DHCPDISCOVER 广播，该广播消息在整个 network 中都可以被收到。

（2）广播到达 tap6356d532-32，Dnsmasq 在它上面监听。它检查其 host 文件，发现有对应项，它以  DHCPOFFER 消息将 IP 和 gateway IP 发回到 VM\_1。如果有其他DHCP Server的话，它们也可能会发回IP 地址。

（3）VM\_1 发回 DHCPREQUEST 消息确认只接受第二步发的 DHCPOFFER，别的 DHCP Server 给的 IP 可以自行收回了。

（4）Dnsmasq 发回确认消息 DHCPACK，整个过程结束。

参考创建一个虚机在 dnsmasq 的日志的整个过程：

[复制代码](javascript:void(0);)

Apr 11 23:42:14 dnsmasq[7087]: read /var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/addn\_hosts - 14 addresses #读取更新后的文件

Apr 11 23:42:14 dnsmasq-dhcp[7087]: read /var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/host

Apr 11 23:42:14 dnsmasq-dhcp[7087]: read /var/lib/neutron/dhcp/0a4cd030-d951-401a-8202-937b788bea43/opts

Apr 11 23:42:15 dnsmasq[7092]: read /var/lib/neutron/dhcp/c359d42b-2a75-4764-bba9-be76b45f64d8/addn\_hosts - 3 addresses #读取更新后的文件

Apr 11 23:42:15 dnsmasq-dhcp[7092]: read /var/lib/neutron/dhcp/c359d42b-2a75-4764-bba9-be76b45f64d8/host

Apr 11 23:42:15 dnsmasq-dhcp[7092]: read /var/lib/neutron/dhcp/c359d42b-2a75-4764-bba9-be76b45f64d8/opts

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 available DHCP subnet: 10.0.50.0/255.255.255.0 #第一个 subnet

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 available DHCP subnet: 10.0.0.0/255.255.255.0 #第二个 subnet

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 vendor class: udhcp 1.20.1

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 DHCPDISCOVER(tap6356d532-32) fa:16:3e:0f:13:7e #收到虚机的 DHCPDISCOVER 消息

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 tags: tag0, known, tap6356d532-32 #从 10.0.0.0/255.255.255.0 中分配地址 （为什么只是 tag0 呢？）

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 DHCPOFFER(tap6356d532-32) 10.0.0.143 fa:16:3e:0f:13:7e #发回IP地址

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 requested options: 1:netmask, 3:router, 6:dns-server, 12:hostname,

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 requested options: 15:domain-name, 28:broadcast, 42:ntp-server,

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 requested options: 121:classless-static-route

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 next server: 10.0.0.116

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 1 option: 53 message-type 2

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 54 server-identifier 10.0.0.116

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 51 lease-time 1d

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 58 T1 12h

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 59 T2 21h

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 1 netmask 255.255.255.0

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 28 broadcast 10.0.0.255

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 14 option: 15 domain-name openstacklocal

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 15 option: 12 hostname host-10-0-0-143

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 3 router 10.0.0.1

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 6 dns-server 192.168.1.1

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 available DHCP subnet: 10.0.50.0/255.255.255.0

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 available DHCP subnet: 10.0.0.0/255.255.255.0

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 vendor class: udhcp 1.20.1

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 DHCPREQUEST(tap6356d532-32) 10.0.0.143 fa:16:3e:0f:13:7e #确认虚机只接受该IP。别的 DHCP Service 可以收回它们废品的 IP 地址了。

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 tags: tag0, known, tap6356d532-32

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 DHCPACK(tap6356d532-32) 10.0.0.143 fa:16:3e:0f:13:7e host-10-0-0-143 #Dnsmasq 确认该 OFFER，以及发回其它info，整个过程结束。

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 requested options: 1:netmask, 3:router, 6:dns-server, 12:hostname,

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 requested options: 15:domain-name, 28:broadcast, 42:ntp-server,

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 requested options: 121:classless-static-route

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 next server: 10.0.0.116

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 1 option: 53 message-type 5

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 54 server-identifier 10.0.0.116

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 51 lease-time 1d

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 58 T1 12h

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 59 T2 21h

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 1 netmask 255.255.255.0

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 28 broadcast 10.0.0.255

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 14 option: 15 domain-name openstacklocal

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 15 option: 12 hostname host-10-0-0-143

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 3 router 10.0.0.1

Apr 11 23:42:41 dnsmasq-dhcp[7087]: 116908156 sent size: 4 option: 6 dns-server 192.168.1.1

[复制代码](javascript:void(0);)

**5. DHCP Agent 的性能和可靠性**

一方面，DHCP Agent 在一个 network 内使用一个 Dnsmasq 提供 DHCP 服务，在 network 很大的时候，可能会有性能问题，比如虚机无法及时获取IP 地址，这时候就需要做性能优化的工作了。而且它依赖于 Neutron Server 发来的 notification，在 AMQP 遇到性能问题的时候，它由可能无法及时收到通知，导致无法提供服务。Marintas 有篇文章谈这个问题 <https://www.mirantis.com/blog/improving-dhcp-performance-openstack/>。

另一方面，DHCP Agent 可能因为一些不可预测的原因不能提供服务，那会造成虚机就无法获取IP 地址。DHCP 协议天然就支持多个使用多个 DHCP Server。这一块，以后在 OpenStack 高可靠性（HA）分析部分会进行分析。

分类: [Neutron](http://www.cnblogs.com/sammyliu/category/677824.html),[Nova](http://www.cnblogs.com/sammyliu/category/669013.html)