



Linux虚拟网络设备之bridge(桥)

赞 8

收藏 33

网络 linux wuyangchun 2017年05月20日发布

5.8k 次浏览

继前两篇介绍了[tun/tap](#)和[veth](#)之后，本篇将介绍Linux下常用的一种虚拟网络设备，那就是bridge(桥)。

本篇将通过实际的例子来一步一步解释bridge是如何工作的。

什么是bridge?

首先，bridge是一个虚拟网络设备，所以具有网络设备的特征，可以配置IP、MAC地址等；其次，bridge是一个虚拟交换机，和物理交换机有类似的功能。

对于普通的网络设备来说，只有两端，从一端进来的数据会从另一端出去，如物理网卡从外面网络中收到的数据会转发给内核协议栈，而从协议栈过来的数据会转发到外面的物理网络中。

而bridge不同，bridge有多个端口，数据可以从任何端口进来，进来之后从哪个口出去和物理交换机的原理差不多，要看mac地址。

创建bridge

我们先用iproute2创建一个bridge:

```
dev@debian:~$ sudo ip link add name br0 type bridge
dev@debian:~$ sudo ip link set br0 up
```

当刚创建一个bridge时，它是一个独立的网络设备，只有一个端口连着协议栈，其它的端口啥都没连，这样的bridge没有任何实际功能，如下图所示：



首页



问答



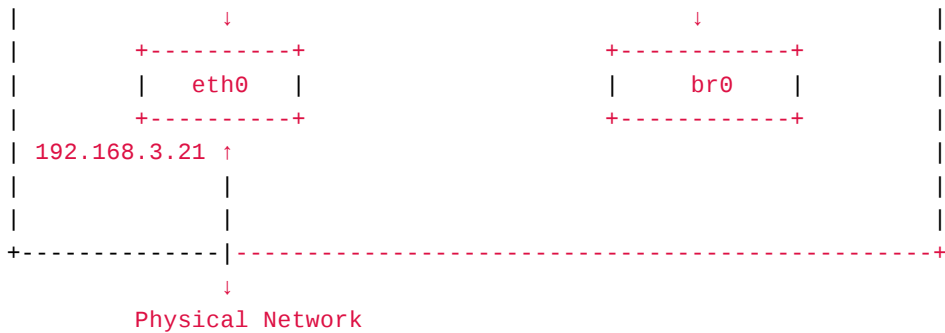
专栏



讲堂



更多



这里假设eth0是我们的物理网卡，IP地址是192.168.3.21，网关是192.168.3.1

将bridge和veth设备相连

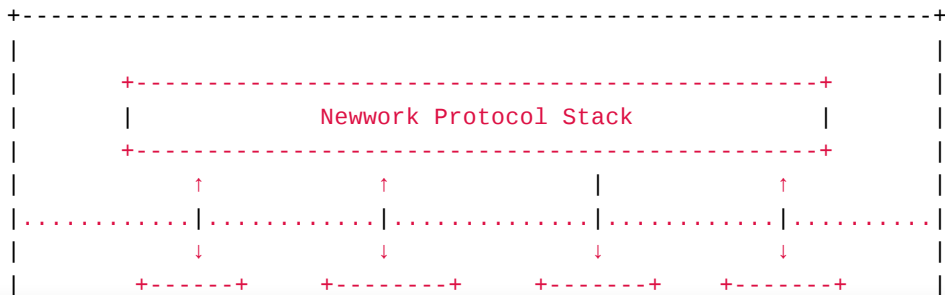
创建一对veth设备，并配置上IP

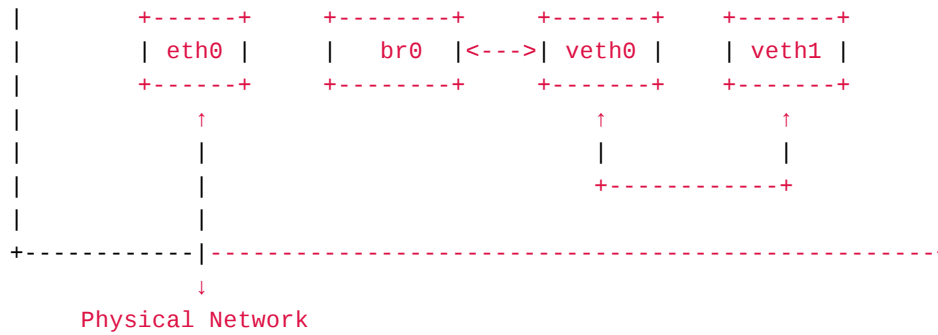
```
dev@debian:~$ sudo ip link add veth0 type veth peer name veth1
dev@debian:~$ sudo ip addr add 192.168.3.101/24 dev veth0
dev@debian:~$ sudo ip addr add 192.168.3.102/24 dev veth1
dev@debian:~$ sudo ip link set veth0 up
dev@debian:~$ sudo ip link set veth1 up
```

将veth0连上br0

```
dev@debian:~$ sudo ip link set dev veth0 master br0
#通过bridge link命令可以看到br0上连接了哪些设备
dev@debian:~$ sudo bridge link
6: veth0 state UP : <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state forwardir
```

这时候，网络就变成了这个样子:





这里为了画图方便，省略了IP地址前面的192.168，比如3.21就表示192.168.3.21

br0和veth0相连之后，发生了几个变化：

- br0和veth0之间连接起来了，并且是双向的通道
- 协议栈和veth0之间变成了单通道，协议栈能发数据给veth0，但veth0从外面收到的数据不会转发给协议栈
- br0的mac地址变成了veth0的mac地址

相当于bridge在veth0和协议栈之间插了一脚，在veth0上面做了点小动作，将veth0本来要转发给协议栈的数据给拦截了，全部转发给bridge了，同时bridge也可以向veth0发数据。

下面来检验一下是不是这样的：

通过veth0 ping veth1失败：

```
dev@debian:~$ ping -c 1 -I veth0 192.168.3.102
PING 192.168.2.1 (192.168.2.1) from 192.168.2.11 veth0: 56(84) bytes of data.
From 192.168.2.11 icmp_seq=1 Destination Host Unreachable

--- 192.168.2.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

为什么veth0加入了bridge之后，就ping不通veth2了呢？先抓包看看：

```
#由于veth0的arp缓存里面没有veth1的mac地址，所以ping之前先发arp请求
#从veth1上抓包来看，veth1收到了arp请求，并且返回了应答
dev@debian:~$ sudo tcpdump -n -i veth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth1, link-type EN10MB (Ethernet), capture size 262144 bytes
21:43:48.353509 ARP, Request who-has 192.168.3.102 tell 192.168.3.101, length 28
21:43:48.353518 ARP, Reply 192.168.3.102 is-at 26:58:a2:57:37:e9, length 28
```



首页



问答



专栏



讲堂



更多

```
dev@debian:~$ sudo tcpdump -n -i veth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth0, link-type EN10MB (Ethernet), capture size 262144 bytes
21:44:09.775392 ARP, Request who-has 192.168.3.102 tell 192.168.3.101, length 28
21:44:09.775400 ARP, Reply 192.168.3.102 is-at 26:58:a2:57:37:e9, length 28

#再看br0上的数据包，发现只有应答
dev@debian:~$ sudo tcpdump -n -i br0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on br0, link-type EN10MB (Ethernet), capture size 262144 bytes
21:45:48.225459 ARP, Reply 192.168.3.102 is-at 26:58:a2:57:37:e9, length 28
```

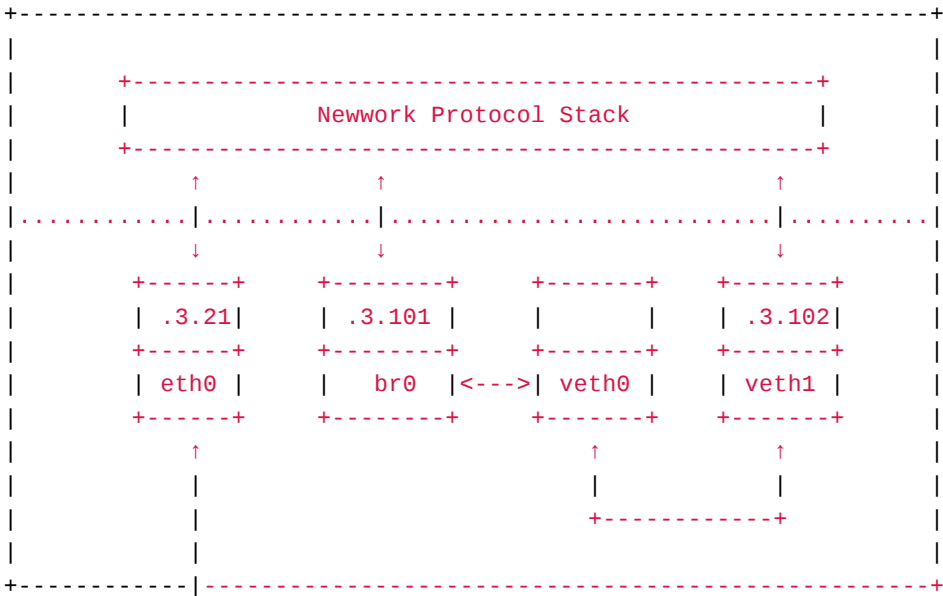
从上面的抓包可以看出，去和回来的流程都没有问题，问题就出在veth0收到应答包后没有给协议栈，而是给了br0，于是协议栈得不到veth1的mac地址，从而通信失败。

给bridge配上IP

通过上面的分析可以看出，给veth0配置IP没有意义，因为就算协议栈传数据包给veth0，应答包也回不来。这里我们就将veth0的IP让给bridge。

```
dev@debian:~$ sudo ip addr del 192.168.3.101/24 dev veth0
dev@debian:~$ sudo ip addr add 192.168.3.101/24 dev br0
```

于是网络变成了这样子：



其实veth0和协议栈之间还是有联系的，但由于veth0没有配置IP，所以协议栈在路由的时候不会将数据包发给veth0，就算强制要求数据包通过veth0发送出去，但由于veth0从另一端收到的数据包只会给br0，所以协议栈还是没法收到相应的arp应答包，导致通信失败。

这里为了表达更直观，将协议栈和veth0之间的联系去掉了，veth0相当于一根网线。

再通过br0 ping一下veth1，结果成功

```
dev@debian:~$ ping -c 1 -I br0 192.168.3.102
PING 192.168.3.102 (192.168.3.102) from 192.168.3.101 br0: 56(84) bytes of data.
64 bytes from 192.168.3.102: icmp_seq=1 ttl=64 time=0.121 ms

--- 192.168.3.102 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.121/0.121/0.121/0.000 ms
```

但ping网关还是失败，因为这个bridge上只有两个网络设备，分别是192.168.3.101和192.168.3.102，br0不知道192.168.3.1在哪。

```
dev@debian:~$ ping -c 1 -I br0 192.168.3.1
PING 192.168.3.1 (192.168.3.1) from 192.168.3.101 br0: 56(84) bytes of data.
From 192.168.3.101 icmp_seq=1 Destination Host Unreachable

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

将物理网卡添加到bridge

将eth0添加到br0上：

```
dev@debian:~$ sudo ip link set dev eth0 master br0
dev@debian:~$ sudo bridge link
2: eth0 state UP : <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state forwarding
6: veth0 state UP : <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 state forwardir
```

br0根本不区分接入进来的是物理设备还是虚拟设备，对它来说都一样的，都是网络设备，所以当eth0加入br0之后，落得和上面veth0一样的下场，从外面网络收到的数据包将无条件的转发给br0，自己变成了一根网线。

这时通过eth0来ping网关失败，但由于br0通过eth0这根网线连上了外面的物理交换机，所以连在br0上的设备都能ping通网关，这里连上的设备就是veth1和br0自己，veth1是通过veth0这根网线连上去的，而br0可以理解为自己有一根直连的网线。



首页



问答



专栏



讲堂



更多

```
#通过eth0来ping网关失败
dev@debian:~$ ping -c 1 -I eth0 192.168.3.1
PING 192.168.3.1 (192.168.3.1) from 192.168.3.21 eth0: 56(84) bytes of data.
From 192.168.3.21 icmp_seq=1 Destination Host Unreachable

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

#通过br0来ping网关成功
dev@debian:~$ ping -c 1 -I br0 192.168.3.1
PING 192.168.3.1 (192.168.3.1) from 192.168.3.101 br0: 56(84) bytes of data.
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=27.5 ms

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 27.518/27.518/27.518/0.000 ms

#通过veth1来ping网关成功
dev@debian:~$ ping -c 1 -I veth1 192.168.3.1
PING 192.168.3.1 (192.168.3.1) from 192.168.3.102 veth1: 56(84) bytes of data.
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=68.8 ms

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

由于eth0已经变成了和网线差不多的功能，所以在eth0上配置IP已经没有什么意义了，并且还会影响协议栈的路由选择，比如如果上面ping的时候不指定网卡的话，协议栈有可能优先选择eth0，导致ping不通，所以这里需要将eth0上的IP去掉。

```
#在本人的测试机器上，由于eth0上有IP，
#访问192.168.3.0/24网段时，会优先选择eth0
dev@debian:~$ sudo route -v
Kernel IP routing table

```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
default	192.168.3.1	0.0.0.0	UG	0	0	0	eth0
link-local	*	255.255.0.0	U	1000	0	0	eth0
192.168.3.0	*	255.255.255.0	U	0	0	0	eth0
192.168.3.0	*	255.255.255.0	U	0	0	0	veth1
192.168.3.0	*	255.255.255.0	U	0	0	0	br0

```

#由于eth0已连接入了br0，所有它收到的数据包都会转发给br0，
#于是协议栈收不到arp应答包，导致ping失败
dev@debian:~$ ping -c 1 192.168.3.1
PING 192.168.3.1 (192.168.3.1) 56(84) bytes of data.
From 192.168.3.21 icmp_seq=1 Destination Host Unreachable

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```



首页



问答



专栏



讲堂



更多

```
dev@debian:~$ sudo ip addr del 192.168.3.21/24 dev eth0
```

```
#再ping一次, 成功
dev@debian:~$ ping -c 1 192.168.3.1
PING 192.168.3.1 (192.168.3.1) 56(84) bytes of data:
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=3.91 ms

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.916/3.916/3.916/0.000 ms
```

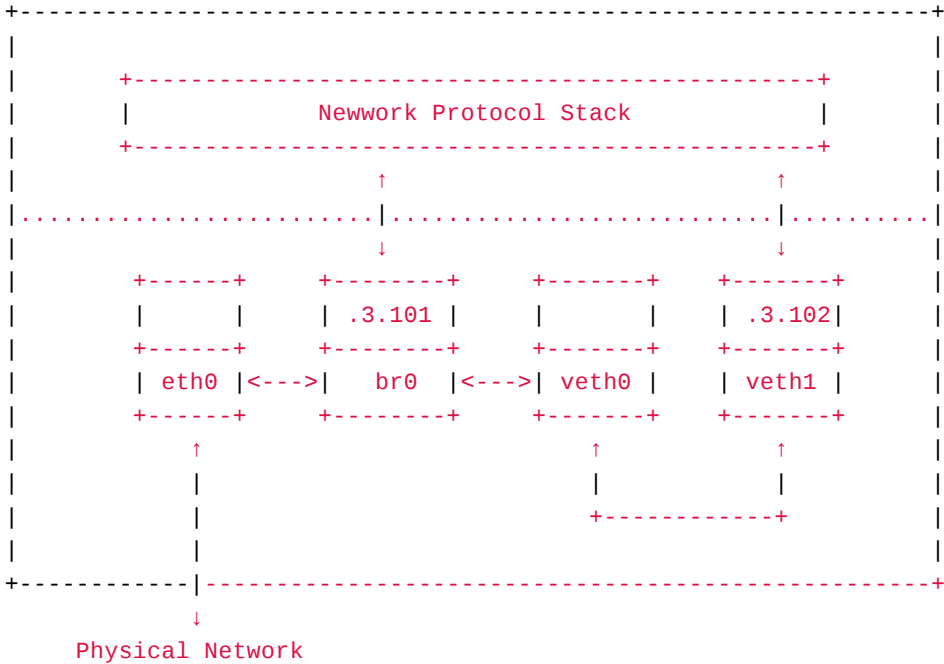
#这是因为eth0没有IP之后, 路由表里面就没有它了, 于是数据包会从veth1出去

```
dev@debian:~$ sudo route -v
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.3.0      *                255.255.255.0    U        0      0      0 veth1
192.168.3.0      *                255.255.255.0    U        0      0      0 br0
```

#从这里也可以看出, 由于原来的默认路由走的是eth0, 所以当eth0的IP被删除之后,
#默认路由不见了, 想要连接192.168.3.0/24以外的网段的话, 需要手动将默认网关加回来

```
#添加默认网关, 然后再ping外网成功
dev@debian:~$ sudo ip route add default via 192.168.3.1
dev@debian:~$ ping -c 1 baidu.com
PING baidu.com (111.12.101.208) 56(84) bytes of data:
```

经过上面一系列的操作后, 网络变成了这个样子:



上面的操作中有一点需要注意:

- 如果是在虚拟机上做上述操作，记得打开网卡的混杂模式（不是在Linux里面，而是在虚拟机的配置上面，如VirtualBox上相应虚拟机的网卡配置项里面），不然veth1的网络会不通，因为eth0不在混杂模式的话，会丢掉目的mac地址是veth1的数据包
- 上面虽然通了，但由于Linux下arp的特性，当协议栈收到外面的arp请求时，不管是问101还是102，都会回复两个arp应答，分别包含br0和veth1的mac地址，也即Linux觉得外面发给101和102的数据包从br0和veth1进协议栈都一样，没有区别。由于回复了两个arp应答，而外面的设备只会用其中的一个，并且具体用哪个会随时间发生变化，于是导致一个问题，就是外面回复给102的数据包可能从101的br0上进来，即通过102 ping外面时，可能在veth1抓不到回复包，而在br0上能抓到回复包。说明数据流在交换机那层没有完全的隔离开，br0和veth1会收到对方的IP应答包。为了解决上述问题，可以配置rp_filter, arp_filter, arp_ignore, arp_announce等参数，但不建议这么做，容易出错，调试比较麻烦。
- 在无线网络环境中，情况会变得比较复杂，因为无线网络需要登录，登陆后无线路由器只认一个mac地址，所有从这台机器出去的mac地址都必须是那一个，于是通过无线网卡上网的机器上的所有虚拟机想要上网的话，都必须依赖虚拟机管理软件（如VirtualBox）将每个虚拟机的网卡mac地址转成出口的mac地址（即无线网卡的mac地址），数据包回来的时候还要转回来，所以如果一个IP有两个ARP应答包的话，有可能导致mac地址的转换有问题，导致网络不通，或者有时通有时不通。解决办法就是将连接进br0的所有设备的mac地址都改成和eth0一样的mac地址，因为eth0的mac地址会被虚拟机正常的做转换。在上面的例子中，执行下面的命令即可：

```
dev@debian:~$ sudo ip link set dev veth1 down
#08:00:27:3b:0d:b9是eth0的mac地址
dev@debian:~$ sudo ip link set dev veth1 address 08:00:27:3b:0d:b9
dev@debian:~$ sudo ip link set dev veth1 up
```

bridge必须要配置IP吗？

在我们常见的物理交换机中，有可以配置IP和不能配置IP两种，不能配置IP的交换机一般通过com口连上去做配置（更简单的交换机连com口的没有，不支持任何配置），而能配置IP的交换机可以在配置好IP之后，通过该IP远程连接上去做配置，从而更方便。

bridge就属于后一种交换机，自带虚拟网卡，可以配置IP，该虚拟网卡一端连在bridge上，另一端跟协议栈相连。和物理交换机一样，bridge的工作不依赖于该虚拟网卡，但bridge工作不代表机器能连上网，要看组网方式。

删除br0上的IP：

```
dev@debian:~$ sudo ip addr del 192.168.3.101/24 dev br0
```

于是网络变成了这样子，相当于br0的一个端口通过eth0连着交换机，另一个端口通过veth0连着veth1：

+-----+
|



首页



问答



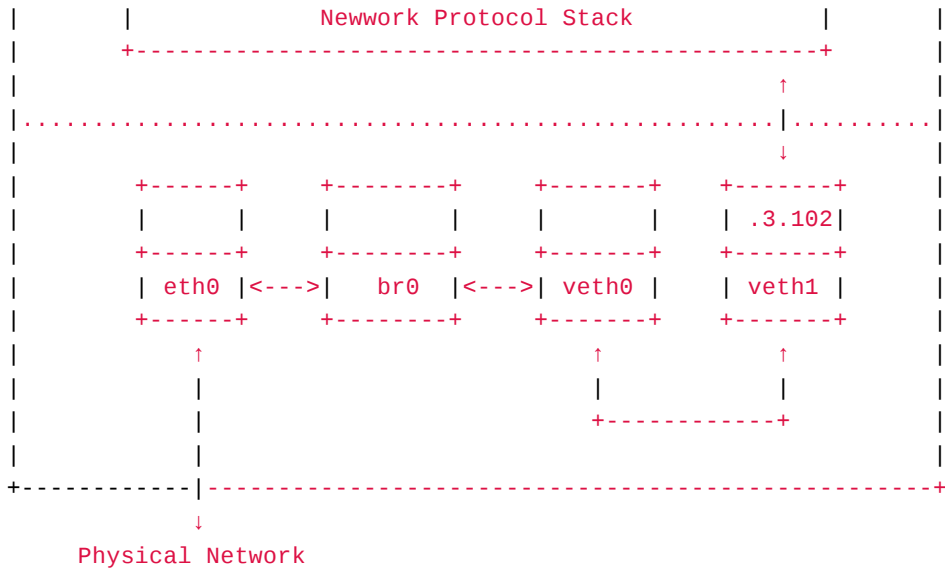
专栏



讲堂



更多



ping网关成功，说明这种情况下br0不配置IP对通信没有影响，数据包还能从veth1出去：

```
dev@debian:~$ ping -c 1 192.168.3.1
PING 192.168.3.1 (192.168.3.1) 56(84) bytes of data.
64 bytes from 192.168.3.1: icmp_seq=1 ttl=64 time=1.24 ms

--- 192.168.3.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.242/1.242/1.242/0.000 ms
```

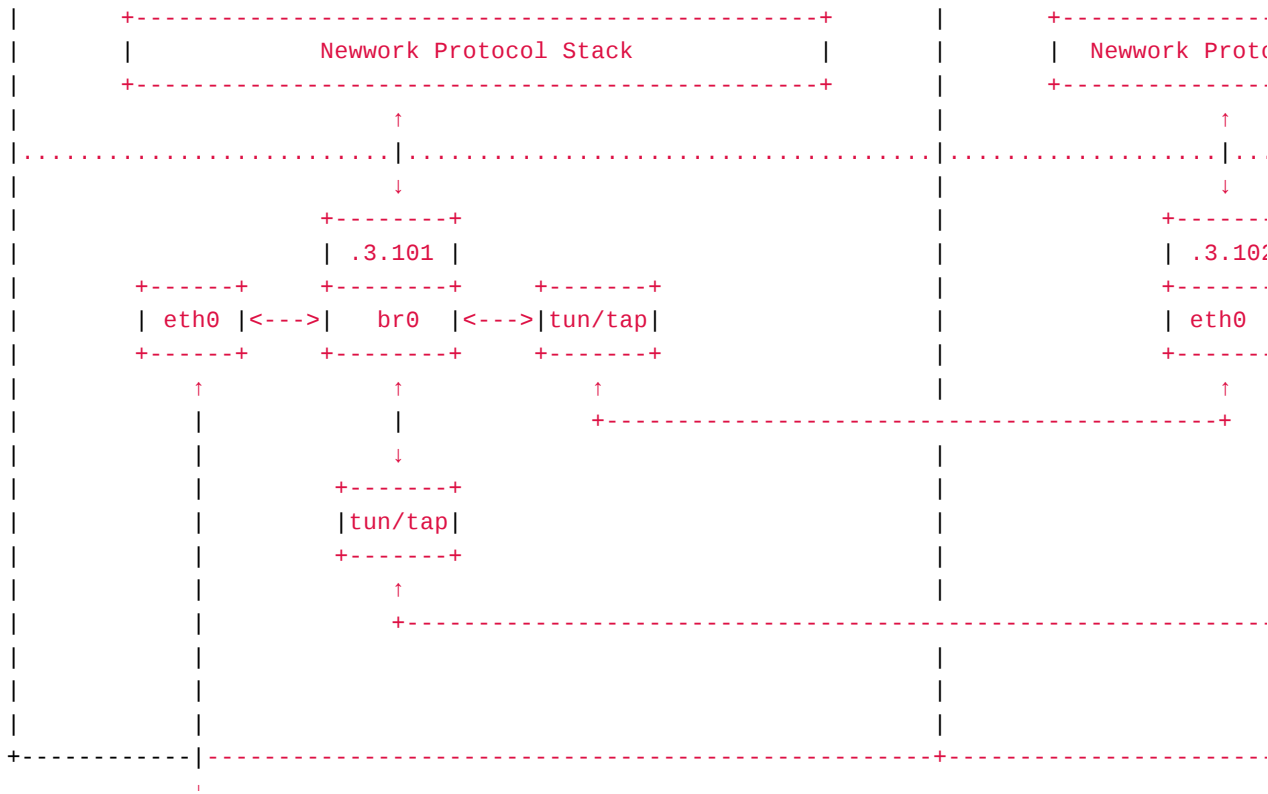
上面如果没有veth0和veth1的话，删除br0上的IP后，网络将会不通，因为没有设备和协议栈完全相连

bridge常用场景

上面通过例子展示了bridge的功能，但例子中的那种部署方式没有什么实际用途，还不如在一个网卡上配置多个IP地址来的直接。这里来介绍两种常见的部署方式。

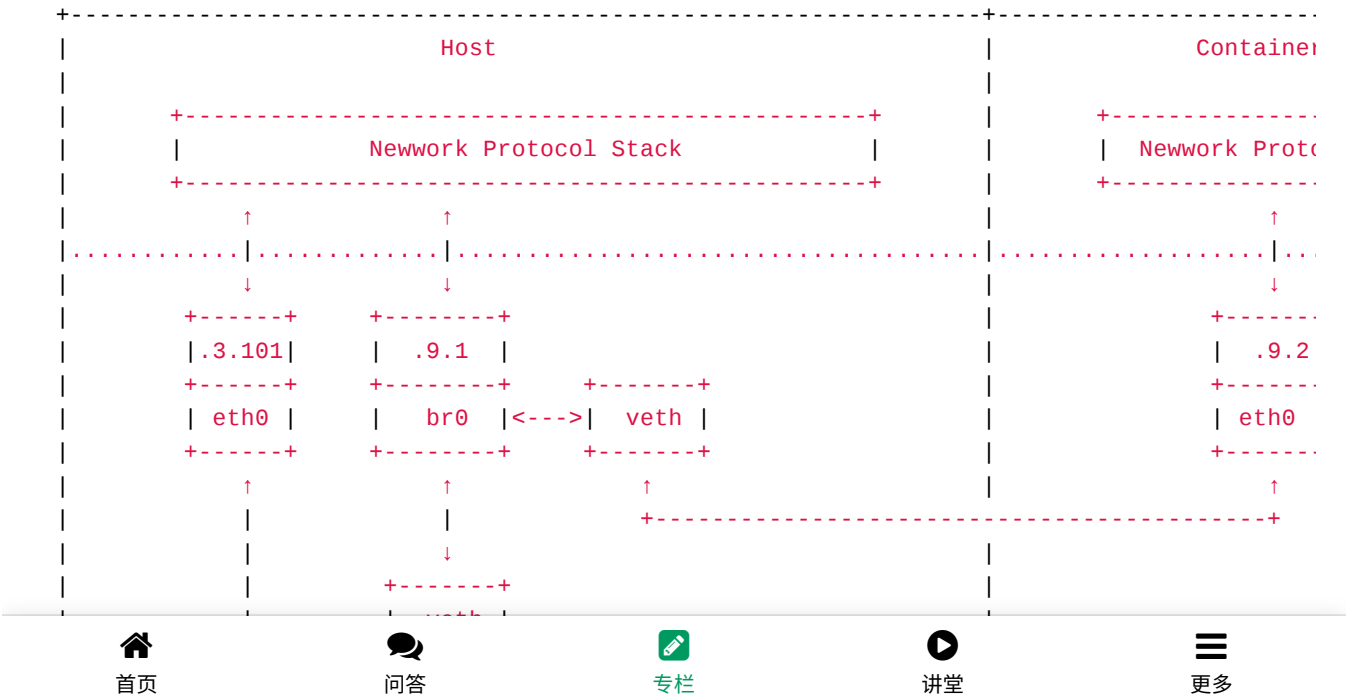
虚拟机

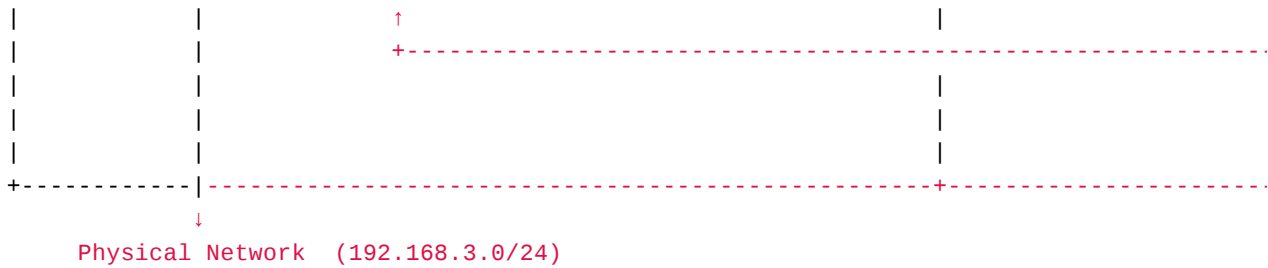
虚拟机通过tun/tap或者其它类似的虚拟网络设备，将虚拟机内的网卡同br0连接起来，这样就达到和真实交换机一样的效果，虚拟机发出去的数据包先到达br0，然后由br0交给eth0发送出去，数据包都不需要经过host机器的协议栈，效率高。



docker

由于容器运行在自己单独的network namespace里面，所以都有自己单独的协议栈，情况和上面的虚拟机差不多，但它采用了另一种方式来和外界通信：





容器中配置网关为.9.1，发出去的数据包先到达br0，然后交给host机器的协议栈，由于目的IP是外网IP，且host机器开启了IP forward功能，于是数据包会通过eth0发送出去，由于.9.1是内网IP，所以一般发出去之前会先做NAT转换（NAT转换和IP forward功能都需要自己配置）。由于要经过host机器的协议栈，并且还要做NAT转换，所以性能没有上面虚拟机那种方案好，优点是容器处于内网中，安全性相对要高点。（由于数据包统一由IP层从eth0转发出去，所以不存在mac地址的问题，在无线网络环境下也工作良好）

上面两种部署方案中，同一网段的每个网卡都有自己单独的协议栈，所以不存在上面说的多个ARP的问题

参考

- [Linux 上的基础网络设备详解](#)
- [Harping on ARP](#)
- [MAC address spoofing](#)
- [It doesn't work with my Wireless card!](#)

2017年05月20日发布 ...

赞 | 8

收藏 | 33

你可能感兴趣的文章

[centos6.6学习笔记：linux网络配置](#) 670 浏览

首页

问答

专栏

讲堂

更多



本作品采用 [署名-非商业性使用-禁止演绎 4.0 国际许可协议](#) 进行许可。

24 条评论

默认排序 时间排序



ruanhao · 2017年05月31日

你好，请问你的ascii图是用什么工具画的啊？

👍 赞 回复

用的vim编辑器，太复杂的图布局起来有点麻烦，画简单的还是挺方便的

— [wuyangchun](#) 作者 · 2017年05月31日

挺好的

— [ruanhao](#) · 2017年06月01日

@[wuyangchun](#) 需要安装什么插件吗？辛苦推荐一下，感谢。

— [knight](#) · 2017年12月11日

[添加回复](#)



ruanhao · 2017年06月13日

你好，在我的环境中，当不设置网桥时，并不能通过veth0 ping 通 veth1 诶，抓包显示 vth1 上能收到 arp request，但是无法回复 arp response，想问下你有做别的什么配置吗？

我的系统：Ubuntu 14.04.1 LTS，参考：<https://serverfault.com/quest...>

👍 赞 回复

补充下，将 veth0 和 veth1 分别放在不同的 namespace 下是可以的，同一个 ns 就不行。。。

— [ruanhao](#) · 2017年06月13日

我一直用debian，所以没遇到你的问题，今天有空在ubuntu上试了一下，果然有你说的的问题，解决办法如下：

```
root@ubuntu:~# echo 1 > /proc/sys/net/ipv4/conf/veth1/accept_local
root@ubuntu:~# echo 1 > /proc/sys/net/ipv4/conf/veth0/accept_local
root@ubuntu:~# echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
root@ubuntu:~# echo 0 > /proc/sys/net/ipv4/conf/veth0/rp_filter
root@ubuntu:~# echo 0 > /proc/sys/net/ipv4/conf/veth1/rp_filter
```

rp_filter和accept_local的含义可以google一下，三言两语解释不清楚。

我也比较了一下debian和ubuntu的这部分配置，debian上面accept_local是0，但没有这个问题，veth1会回复arp应答包，可能有其他的配置项我没找到，或者是debian有自己专门的与arp应答有关的补丁。

看了下思科设备是好像还是在不同的namespace里面比较好，两个设备设备层不同的协议栈就不会触发



首页



问答



专栏



讲堂



更多

— wuyangchun 作者 · 2017年06月14日

@wuyangchun cetnos7.4 也测试不成功，通过veth0 ping veth1, 在veth1抓包只有来自veth1的ICMP echo request, 没有回复

— 折木 · 4月30日

添加回复



ruanhao · 2017年06月13日

还有我想问下这里：“问题就出在veth0收到应答包后没有给协议栈，而是给了br0，于是协议栈得不到veth1的mac地址”，既然br0 收到了 arp reponse，而 br0 又和协议栈相连，为什么说协议栈得不到 veth1 的 mac 呢？

👍 赞 回复

应答包要被协议栈认可，需要经过很多过滤条件，在本文中的情况下：

1. 应答包的mac地址要和br0的mac地址匹配，不然就算开了混杂模式，arp层在收到这样的包的时候会第一时间丢弃掉
2. 就算br0和veth0的mac地址一样，这样的应答包到达arp层后，也会因为br0上面没有IP而被丢弃掉
3. 就算br0配置了IP，由于br0的IP和arp应答包中的IP不一致，也可能被丢弃掉

我对内核中arp相关代码不熟，也就没有仔细的去研究到底会在哪一步被丢弃掉，以及有哪些配置项可以控制这些行为。

如果你有兴趣的话，可以先打开日志功能：root@ubuntu:~# echo 1 > /proc/sys/net/ipv4/conf/veth1/log_martians

然后触发一个arp请求，比如ping一下，再根据日志内容（ubuntu默认在/var/log/syslog）去搜内核的代码，应该就能找到大概的丢包位置

— wuyangchun 作者 · 2017年06月14日

@wuyangchun 谢谢你详尽的解释！你的文章帮助我理解了好些以前没搞明白的细节。对了，我在自己的环境中重复你的实验步骤，发现一个看似和路由表相关的奇怪问题：<https://superuser.com/question/...>，不知你是否遇到过这样的问题呢？

— ruanhao · 2017年06月14日

我查了下rp_filter的原理，我的问题应该就是这个参数搞的鬼。

— ruanhao · 2017年06月15日

添加回复 | 显示更多



skambc · 2017年09月03日

bridge实现了交换功能，没有vlan，vlan有单独的设备实现，能否介绍一下linux里面vlan的实现呢，谢谢！

👍 赞 回复



首页



问答



专栏



讲堂



更多

<https://www.ibm.com/developer...>

你可以看看这篇文章、。

— **daocoder** · 3月8日

[添加回复](#)



ShiyaoMA · 2017年12月28日

你好，看了你的文章，对你的ascii图很感兴趣，请问这些图都是怎么画的？谢谢。

👍 赞 回复

刚看到你前面的回复，“用的vim编辑器，太复杂的图布局起来有点麻烦，画简单的还是挺方便的”，不会手巧的吧？

— **ShiyaoMA** · 2017年12月28日

[添加回复](#)



yiqiaoxihui · 1月26日

您好，在bridge常用场景中的第一个场景：虚拟机中，将主机的ip配给网桥后，请问现在主机网卡没了ip,怎么和局域网通信呢？我试了一下，直接ping 局域网ip，根本ping不通，望解答，谢谢！

👍 赞 回复



yiqiaoxihui · 1月26日

您好，在给bridge配上veth0的IP那块，我发现即使没有给bridge配上veth0的ip之前，bridge就能够ping通veth0和veth1，但是用的源ip竟然是docker0的ip，这是为什么，而且在veth0,veth1，bridge抓包发现都只有request包，没有reply，但是确实能ping通。谢谢！

👍 赞 回复



yiqiaoxihui · 1月30日

您好，“veth0加入了bridge之后，就ping不通veth1了”，我在virtual box 中使用ubuntu14.04测试，发现能够ping通，在lo上抓到reply包，veth1和veth0上也没有reply包（和您“Linux虚拟网络设备之veth”中的情况一致），但是br0上没有抓到任何包？那么为什么能ping通？为什么br0上并不像您所说的抓到包？谢谢！

👍 赞 回复



daocoder · 3月8日

我想修改下你上面docker的图示、不过这个文本编辑器太差了、。



首页



问答



专栏



讲堂



更多

👍 赞 回复



heiline · 3月30日

最后一个实验，我手动添加了默认路由，下一跳指向网关，出接口br0；`ping -c 3 -I br0 114.114.114.114`,能ping通。`ping -c 3 -I veth1 114.114.114.114`,ping不通；分析了下，从veth1发出的数据包通过veth0发送给br0，再由br0发送给eth0。抓包结果在eth0上收到了arp请求包，为什么没有arp应答包呢？ 求解，谢谢！

👍 赞 回复



文明社会，理性评论

发表评论



想在上方展示你的广告？



wuyangchun

关注作者

899 声望

发布于专栏

Linux程序员

专注Linux相关技术

103 人关注

关注专栏



首页



问答



专栏




讲堂



更多

系列文章

- Linux虚拟网络设备之tun/tap 43 收藏， 8k 浏览
- Linux虚拟网络设备之veth 17 收藏， 6.2k 浏览

产品	资源	商务	关于	关注	条款
热门问答	每周精选	人才服务	关于我们	产品技术日志	服务条款
热门专栏	用户排行榜	企业培训	加入我们	社区运营日志	内容许可
热门讲堂	徽章	活动策划	联系我们	市场运营日志	
最新活动	帮助中心	广告投放		团队日志	
圈子	声望与权限	区块链解决方案		社区访谈	
找工作	社区服务中心	合作联系			
移动客户端	开发手册				扫一扫下载 App

Copyright © 2011-2018 SegmentFault. 当前呈现版本 17.06.16
浙ICP备 15005796号-2 浙公网安备 33010602002000号 杭州堆栈科技有限公司版权所有

CDN 存储服务由 又拍云 赞助提供