此文章已于 16:26:38 2017/4/17 重新发布到 MADHEX

linux上通过netns创建vrf模拟MCE设备

转：<http://blog.csdn.net/dog250/article/details/26147357>

上周有厂商到公司测试，拿了一块据说很猛的网络处理加速PCIe板卡，拎在手里沉甸甸的很有分量，最让人意淫的是那4个万兆光口，于是我迫不及待的想要一览光口转发时那种看不见的震撼。

可是，仅凭4个光口怎么测试？起码你要有个“对端”啊！任何人应该都不想扛着三台机器在客户们之间跑来跑去测试其转发性能，当然你也不能指望客户那里就一定有你需要的“对端”设备，比如我们公司就没有这种和万兆光口对接的设备，不过赶巧的是，那天还真有一台设备带有万兆光口，但是只是碰巧了。最佳的测试方式当然是不依赖任何外部设备了，显而易见的方法就是做自环。

RJ45口的双绞线可以做物理层自环，1/3，2/6短接即可，这样一台机器的一块网卡自己就可以既发又收了，但是你能对比头发略粗的光纤做什么呢？真实的做法当然是用软件解决了，在Linux上可以使用netns来解决，即net namespace。

netns是一个很好玩的东西，它可以让你在一台机器上模拟多个网络设备，这样做的意义是非同一般的：

1.使用netns可以充分利用闲置的处理器资源，特别是你的多块网卡性能压不满CPU的时候；

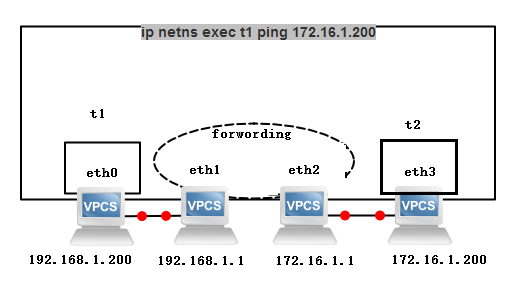
2.使用netns可以将不同类型的网络应用隔离，针对每一类实施不同的策略；

3.使用netns有点玩虚拟化的意思，不过比虚拟机更灵活。

一个net namespace有自己独立的路由表，iptables策略，设备管理机构，和其它的netns完全隔离，比如你将eth0加入了netns1，那么netns2中的应用程序就看不到eth0，网卡设备管理只是netns中的一个元素，还有很多，比如你在netns1中配置的iptables策略对netns2中的数据包没有任何影响。总之，如果你懂Linux内核源码，那么只要附着有net结构体字段的那些结构，比如skb，net\_device，都和netns有关。

那么我应该怎么做自环呢？我的设备有4个网卡，我希望1和4之间通信，通过2和3转发，它的逻辑拓扑如下：

PC1/eth0----PC2/eth1(forward)PC2/eth2----PC3/eth3



很简单，将eth0和eth3设置在两个不同的netns，然后用线缆连接eth0和eth1，同样连接eth2和eth3，最后将eth0和eth1的IP地址设置在一个网段，将eth2和eth3的IP地址设置在另一个不同的网段即可。光说不练假把式，具体应该怎么做呢？同样很简单：

1.添加两个netns

ip netns add t1

ip netns add t2

2.将eth0加入t1，并且设置IP地址

ip link set eth0 netns t1

此时再ifconfig就看不到eth0了，你甚至执行ls /sys/class/net也看不到eth0了，只有执行ip netns exec t1 ls /sys/class/net才能看到。

ip netns exec t1 ifconfig eth0 192.168.1.200/24

3.将eth3加入t2，并且设置IP地址

ip link set eth3 netns t2

此时ifconfig就看不到eth3了，你甚至执行ls /sys/class/net也看不到eth3了，只有执行ip netns exec t2 ls /sys/class/net才能看到。

ip netns exec t1 ifconfig eth3 172.16.1.200/24

4.设置eth1和eth2的地址

ifconfig eth1 192.168.1.1/24

ifconfig eth2 172.16.1.1/24

5.设置两个netns的默认路由

ip netns exec t1 route add default gw 192.168.1.1

ip netns exec t2 route add default gw 172.16.1.1

6.测试

在netns t1中ping netns t2中的eth3地址

ip netns exec t1 ping 172.16.1.200

上述配置之后，从eth0发出的包会通过网线到达eth1(而不是走local路由表的loopback)，然后经过eth1的forward从eth2发出。经由网线到达目的地eth3杯接收。整个过程中就一台机器，展示出的效果好像三台机器的样子。有了这个机制，是不是再也不用为搭建测试环境而发愁了呢？

除了自环测试之外，netns还可以用于设置策略路由，这种策略路由不需要ip rule。试想一种场景，你同时运行了P1和P2两个程序，本机所在的局域网有两个出口到达外网，你希望P1通过gw1和外界通信，P2通过gw2和外界通信，约束条件是你的机器只有一张网卡eth0，怎么办呢？通过iptables为P1和P2的数据包打上不同的mark，然后通过ip rule设置策略路由无疑可以解决，另外直接在P1和P2应用程序中用setsockopt也是可以设置ipmark的，这就不需要iptables了。然而这一切都过时了，2014年我需要一种不同的方式。

我不知道怎么表达我思考的过程，但是给出一个操作序列是简单的事情，因为照着这么做确实可以满足需求，然后看到这篇文章的人照着操作步骤倒推回去，就可以得到一个思考过程。首先你要明白的是Linux内核支持一种虚拟网卡类型，即veth，一般而言veth是成对的，从一个veth发出的数据包可以直接到达它的peer veth，感兴趣的可以看Linux内核的drivers/net/veth.c，和drivers/net/tun.c没什么不同，更简单些罢了。第一步要做的就是建立一对veth：

ip link add veth1 type veth peer name veth2

此时系统中除了eth0之外又多了两块网卡，所有的网卡为lo，eth0，veth1，veth2。中间隐含着一个事实，即veth1和veth2之间有一条虚拟的链路将两块网卡连接起来，就好像一条双绞线连接的两块物理网卡一样。我现在希望P1的数据包通过veth1发出，然后自然而然地就能发到veth2，但是随后怎么通过eth0发到物理线路呢？太简单，太简单，使用bridge吧：

brctl addbr br0

brctl addif br0 eth0 veth2

同时，veth1和br0所在的局域网设置在一个IP网段中，这下子就全通了，该二层网络的逻辑拓扑为：

veth1----veth2(bridge)eth0----gw(1,2)

怎么设置netns我本来不想说了，但是由于小小暂时不跟我玩了，我还是写完吧。首先将veth1设置到netns1(具体怎么创建netns，不再赘述)并设置路由：

ip link set veth1 netns netns1

ip netns exec netns1 route add default gw $gw1

route add default gw $gw2

这就完了？是的，完事了。事实上，保留br0的默认netns即可，没有必要创建netns2了。接下来需要做的就是启动P1和P2了：

ip netns exec netns1 P1

P2

好了，一切结束。

我始终都觉得，在Linux上一般都是不用修改源码就能解决问题，可是我还是喜欢修改代码，原因何在？很简单，源码很容易获得，并且源码很容易修改，我走火入魔般地写了大量的Netfilter扩展以及做了大量的nf\_conntrack修改，甚至还添加了一些该死的socket filter...虽然这些行为都是自娱自乐型的，并没有被应用在工作中，但是这些行为说明我不是网络管理员，而是一名程序员，哈哈，自封的资深软件工程师(我还是觉得这些成果能被应用)。然而，做一名技术精湛的网络管理人员的难度却远远超过做程序员的难度。这不，又一次遇到了OpenVPN的多实例问题，我觉得，单纯的程序员搞不定它，单纯的网管也不行。

TAP模式的多实例已经被我用Linux Bridge完美蹂躏了，但是TUN模式的多实例问题仍然没有完美的方案，虽然修改tun驱动，使用broadcast mode bonding+tun filter可以解决，但是我还是觉得那是一种走火入魔的方式，因此就算在公司我也没能将整个调试测试进行下去，结果落了个不了了之，事实上，是我太不喜欢那种方式。tun的IP filter是我改出来的方案，并非标准的，能不能使用标准的方式进行寻址呢？使用netns，答案就是肯定的。

假设在GW上启动了2个OpenVPN实例ovpn1和ovpn2，虚拟网卡分别为tun1和tun2，在client-connect脚本中得知ovpn2负责N1，ovpn2负责N2。现在问题的关键是，GW后方出发的数据包如何知道是将数据包发送到tun1还是tun2，这个判断能不能自动进行？如果使用netns，那就是可以的，我可以将2个tun分别设置在不同的netns，然后每一个netns对应一个同处一个netns的veth虚拟网卡，这些veth的peer们处在另外一个netns中，这样就可以实现IP层TUN模式虚拟网卡到以太网的TAP模式虚拟网卡的适配。最后将这些peer们Bridge成一个br0，那么TUN模式的OpenVPN就能和TAP模式的OpenVPN采用同一种方式处理了。

不管怎样，当你玩弄netns的时候，你要知道你并不是在玩弄冷酷无情的虚拟化操作系统，也不是真的模拟了两台物理上相互隔离的机器，因为虽然两个程序的网络是隔离的，但是文件系统却是共享的。你要时刻准备着，使用网络隔离和使用内存，文件系统共享相结合。将一台机器既可以作为多台机器使用，又可以作为一台机器共享资源！

不管怎样，当你玩弄netns的时候，你要知道你并不是在玩弄冷酷无情的虚拟化操作系统，也不是真的模拟了两台物理上相互隔离的机器，因为虽然两个程序的网络是隔离的，但是文件系统却是共享的。你要时刻准备着，使用网络隔离和使用内存，文件系统共享相结合。将一台机器既可以作为多台机器使用，又可以作为一台机器共享资源！

理解了上述的例子和最后的总结，那么我来发问，单网卡或者没有网卡怎么玩自环？这个需求可能就是为了测试一下协议栈而已。略去思考的过程，很简单，多加一个层次。比如你有一台机器一块网卡也没有，那么你只需要下面的命令就可以在你的机器上实现IP转发或者bridge转发了：

ip link add v1 type veth peer name vp1

ip link add v2 type veth peer name vp2

brctl addbr br0

brctl addif vp1 vp2

ifconfig vp1 up

ifconfig vp2 up

sysctl -w net.ipv4.ip\_forward=1

ip netns add t1

ip netns add t2

ip link set v1 netns t1

ip link set v2 netns t2

ip netns exec t1 ifconfig v1 1.1.1.1/24

ip netns exec t2 ifconfig v2 1.1.1.2/24

ip netns exec t1 ping 1.1.1.2

...