# 高级网络应用

目录

[高级网络应用 1](#_Toc1955)

[第一章 链路聚合简介 1](#_Toc279)

[1、链路聚合介绍 1](#_Toc17330)

[2、 Linux配置链路聚合的两种方式 1](#_Toc22889)

[第二章 实验 1](#_Toc90)

[1、实验-网卡链路聚合 1](#_Toc4403)

[（1）bonding一共有7种工作模式 1](#_Toc21461)

[（2）新建bond 2](#_Toc23001)

[（3）故障测试 2](#_Toc5568)

[（4）删除bond模式的链路聚合 2](#_Toc4390)

[2、实验-桥接 3](#_Toc1745)

[3、实验-删除虚拟桥接网卡virbr0 3](#_Toc20137)

[4、实验-添加ipv6地址 3](#_Toc6985)

1. **链路聚合简介**

**1、链路聚合介绍**

链路聚合（英语：Link Aggregation）是一个[计算机网络](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C" \t "/root/文档\\x/_blank)术语，指将多个物理端口汇聚在一起，形成一个逻辑端口，以实现出/入流量[吞吐量](https://baike.baidu.com/item/%E5%90%9E%E5%90%90%E9%87%8F" \t "/root/文档\\x/_blank)在各成员端口的负荷分担，交换机根据用户配置的端口负荷分担策略决定[网络封包](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%B0%81%E5%8C%85" \t "/root/文档\\x/_blank)从哪个成员端口发送到对端的交换机。当交换机检测到其中一个成员端口的链路发生故障时，就停止在此端口上发送封包，并根据负荷分担策略在剩下的链路中重新计算报文的发送端口，故障端口恢复后再次担任收发端口。链路聚合在增加链路带宽、实现链路传输弹性和工程冗余等方面是一项很重要的技术。

1. **Linux配置链路聚合的两种方式**

网卡的链路聚合一般常用的有"bond"和"team"两种模式，"bond"模式最多可以添加两块网卡，"team"模式最多可以添加八块网卡。

1. **实验**

**1、实验-网卡链路聚合**

**（1）bonding一共有7种工作模式**

　　0：(balance-rr) Round-robin policy: (平衡轮询策略)：传输数据包顺序是依次传输，直到最后一个传输完毕， 此模式提供负载平衡和容错能力。

　　1：(active-backup) Active-backup policy:(活动备份策略)：只有一个设备处于活动状态。 一个宕掉另一个马上由备份转换为主设备。mac地址是外部可见得。 此模式提供了容错能力。

　　2：(balance-xor) XOR policy:(平衡策略)：传输根据[(源MAC地址xor目标MAC地址)mod 设备数量]的布尔值选择传输设备。 此模式提供负载平衡和容错能力。

　　3：(broadcast) Broadcast policy:(广播策略)：将所有数据包传输给所有设备。 此模式提供了容错能力。

　　4：(802.3ad) IEEE 802.3ad Dynamic link aggregation. IEEE 802.3ad 动态链接聚合：创建共享相同的速度和双工设置的聚合组。此模式提供了容错能力。每个设备需要基于驱动的重新获取速度和全双工支持；如果使用交换机，交换机也需启用 802.3ad 模式。

　　5：(balance-tlb) Adaptive transmit load balancing(适配器传输负载均衡)：通道绑定不需要专用的交换机支持。发出的流量根据当前负载分给每一个设备。由当前设备处理接收，如果接受的设备传不通就用另一个设备接管当前设备正在处理的mac地址。

　　6：(balance-alb) Adaptive load balancing: (适配器负载均衡)：包括mode5，由 ARP 协商完成接收的负载。bonding驱动程序截获 ARP 在本地系统发送出的请求，用其中之一的硬件地址覆盖从属设备的原地址。就像是在服务器上不同的人使用不同的硬件地址一样。

在uos1添加2块新的网卡

nmcli device show | grep DEVICE #发现新增2块网卡ens33和ens37

将原有网卡规则删除掉，以防影响后续实验

nmcli connection delete ens33

**（2）新建bond**

nmcli connection add type bond con-name bond0 mode active-backup ipv4.addresses 192.168.200.201/24" #表示添加一个bond，名称为bond0，工作模式为主备，IP为"192.168.200.201"。

nmcli connection add con-name ens33 ifname ens33 type bond-slave master bond0

#将ens33网卡连接添加到这个bond中。

nmcli connection add con-name ens37 ifname ens37 type bond-slave master bond0

#将ens37网卡连接添加到这个bond中。

nmcli connection up ens33 #启动bond-slave ens33

nmcli connection up ens37 #启动bond-slave ens37

nmcli connection up bond0 #启动bond0

ip a #查看ens33和ens37网卡mac地址相同

cat /proc/net/bonding/nm-bond #查看bond已生效

**（3）故障测试**

nmcli device disconnect ens33 #禁掉当前网卡

cat /proc/net/bonding/nm-bond #查看当前的活动网卡为ens37

nmcli device connect ens33 #重新添加ens33网卡

cat /proc/net/bonding/nm-bond #ens33网卡已成为备用网卡

**（4）删除bond模式的链路聚合**

nmcli connection delete bond0

nmcli connection delete ens33

nmcli connection delete ens37

systemctl restart NetworkManager

**2、实验-桥接**

还原虚拟机到ok状态

nmcli connection delete ens33

nmcli connection add type bridge con-name uosbr ifname uosbr

nmcli connection modify uosbr ipv4.method manual ipv4.addresses 192.168.200.201/24 ipv4.gateway 192.168.200.201 ipv4.dns 192.168.200.201

nmcli connection add type bridge-slave con-name uosbrslave1 ifname ens33 master uosbr

nmcli connection up uosbr

systemctl restart NetworkManager

nmcli connection show #如果出现重名的uosbr配置文件，需要删除错误的uuid，再nmcli connection up uosbr和systemctl restart NetworkManager

ping -I uosbr 192.168.100.202 #-I表示从哪块网卡发出ping包

**3、实验-删除虚拟桥接网卡virbr0**

方法1

apt-get remove -y libvirt-\*

reboot

nmcli connection show #虚拟桥接网卡virbr0已删除

方法2

virsh net-list #查看虚拟网络设备

virsh net-destroy default #删除名为default的设备

virsh net-undefine default #删除配置文件中的相关信息

systemctl restart libvirtd

nmcli connection show #虚拟桥接网卡virbr0已删除

**4、实验-添加ipv6地址**

nmcli connection show #查看配置ipv6的网卡名称，这里为ens33

nmcli connection modify ens33 ipv6.method manual ipv6.addresses 2001::1/64 ipv6.gateway 2001::1 ipv6.dns 2001::1

nmcli connection down ens33 ; nmcli connection up ens33

ping6 2001::1

为加快识别速度，建议在所有节点上填写hosts文件

在uos1（192.168.200.201）上

useradd uos1

echo "uos1:uos1" | chpasswd

systemctl restart sshd

在uos2（192.168.200.202）上

useradd uos2

echo "uos2:uos2" | chpasswd

ssh 192.168.200.201

ssh -X uos1@192.168.200.201

su - uos2

ssh uos1@192.168.200.201

在uos1上

vim /etc/ssh/sshd\_config

PermitRootLogin no #激活并修改，禁止root登录

systemctl restart sshd

在uos2上

ssh 192.168.200.201 #root不能登陆

ssh uos1@192.168.200.201 #其他用户可以登录

su - uos2

ssh-keygen #两台主机免密登录

ssh-copy-id uos1@192.168.200.201

ssh uos1@192.168.200.201 #可以不输入密码直接登录

在uos1上

vim /etc/ssh/sshd\_config

PasswordAuthentication no #不可以通过密码方式登录，只能通过证书登录

systemctl restart sshd

在uos2上

ssh uos1@192.168.200.201 #不能通过密码登录

su - uos2

ssh uos1@192.168.200.201 #只能通过证书登录