Aula 05 - Configuração de rede

Sobre

- Objetivos:
  - Entendimento do funcionamento de bridges e interfaces tap.
  - Conhecimento das opções de dispositivos para frontend.
  - utilização de scripts para configuração de interfaces.

Configuração padrão

- Por padrão o KVM cria uma pilha de rede em espaço de usuário dentro do processo que contém a máquina virtual.
  - Nesse ambiente (dentro do processo) existe uma rede virtual que possui:
    - Servidor DHCP
    - Servidor DNS
    - Gateway
- A interface eth0 dentro da máquina virtual, a grosso modo, está atrás de um NAT.

Checando a configuração padrão

- Dê boot em uma VM, vamos verificar como está a configuração de rede obtida via DHCP:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:3456/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1986 (1.9 KB)  TX bytes:1818 (1.8 KB)
          Interrupt:10 Base address:0xe000

# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0
eth0
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0
eth0
```

```
# dmesg |grep 8139
[    1.097676] 8139cp: 10/100 PCI Ethernet driver v1.3 (Mar 22, 2004)
[    1.100073] 8139cp 0000:00:03.0: PCI INT A -> Link[LNKC] -> GSI 10
(level, high) -> IRQ 10
[    1.105789] eth0: RTL-8139C+ at 0xffffc900004fe000,
52:54:00:12:34:56, IRQ 10
[    1.107021] 8139cp 0000:00:03.0: setting latency timer to 64
[    1.230095] 8139too Fast Ethernet driver 0.9.28
```

Configuração de rede padrão

- Por padrão, o kvm usa os seguintes parâmetros para configurar um dispositivo de rede:

```
# kvm -netdev type=user,id=net0 -device rtl8139,netdev=net0
```

Apenas como referência, essa é a sintaxe antiga para se configurar o mesmo dispositivo de rede:

```
# kvm -net user,vlan=0 -net nic,model=rtl8139,vlan=0
```

- Em ambos os casos o endereço MAC apresentado para o guest é sempre o mesmo.
- O processo do KVM está tunelando tudo todo o tráfego em espaço de usuário.
- Prático para configurações simples.
- A performance é limitada e baixa.
- Difícil e inconveniente configuração para acesso externo.
- Não existe uma "interface de verdade" no host utilizada pelo guest.

Mudando o MAC

```
# kvm -netdev type=user,id=net0 -device
rtl8139,netdev=net0,mac=00:00:00:00:AB:10
```

Opções de frontend e backend

- Principais dispositivos suportados (frontend):
  - rtl8139 (Realtek 8139)
  - e1000 (Intel E1000)
  - virtio-net-pci (Interface paravirtualizada VirtIO)
- Principais backends suportados:
  - tap
  - slirp (Pilha de rede em modo-usuário)

- VDE (Virtual Distributed Ethernet)

Conhecendo interfaces tap

- Vamos criar uma interface do tipo tap:

```
# tunctl -t tap0
Set 'tap0' persistent and owned by uid 0

# ifconfig tap0
tap0      Link encap:Ethernet  HWaddr 9a:a8:03:d2:d9:b8
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Interfaces tap fazem encaminhamento de quadros em nível 2 (Ethernet).
- Residem dentro do kernel.
- Podem ser manipuladas normalmente como se fossem uma interface de rede real.
- O endereço MAC é aleatório.
- Levantando a interface tap0 criada:

```
# ifconfig tap0 up

ou

# ip link set tap0 up
```

Usando a interface tap com o kvm

- Abra um shell como root, e inicie um tcpdump na interface tap0

```
# tcpdump -i tap0
```

- **IMPORTANTE**: Edite o script /etc/kvm/kvm-ifup no host e comente **TODAS** as linhas antes de iniciar a máquina virtual.
- Agora vamos iniciar uma VM qualquer e associar a interface tap0 do host com o dispositivo do guest:

```
# kvm -netdev type=tap,ifname=tap0,id=net0 -device rtl8139,netdev=net0
```

- Configurando o MAC (opcionalmente)

```
# kvm -netdev type=tap,ifname=tap0,id=net0 -device
rtl8139,netdev=net0,mac=00:00:00:00:AB:10
```

- Como a máquina virtual estava configurada para DHCP, você verá no tcpdump os broadcasts (que não estão indo para lugar nenhum).
- Saída parecida com essa:

```
# tcpdump -i tap0
tcpdump: WARNING: tap0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on tap0, link-type EN10MB (Ethernet), capture size 65535
bytes
00:08:40.586856 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:00:00:00:00:10 (oui Ethernet), length 300
00:08:43.952360 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:00:00:00:00:10 (oui Ethernet), length 300
00:08:49.952271 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:00:00:00:00:10 (oui Ethernet), length 300
```

Configurando interfaces tap

- Agora nós temos a interface virtual eth0 do guest "plugada" na interface tap0 do host. Todos os quadros Ethernet são copiados um para o outro. Vamos avançar na configuração.
- Coloque um IP para na interface tap0:

```
# ifconfig tap0 192.168.0.1 netmask 255.255.255.0 up
```

- Coloque um IP para na interface eth0 da máquina virtual:

```
# ifconfig eth0 192.168.0.2 netmask 255.255.255.0 up
```

- Agora vamos pingar! Do host:

```
# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_req=1 ttl=64 time=0.279 ms
64 bytes from 192.168.0.2: icmp_req=2 ttl=64 time=0.313 ms
64 bytes from 192.168.0.2: icmp_req=3 ttl=64 time=0.478 ms
```

- De dentro da máquina virtual:

```
# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
```

```
64 bytes from 192.168.0.2: icmp_req=1 ttl=64 time=0.679 ms
64 bytes from 192.168.0.2: icmp_req=2 ttl=64 time=0.513 ms
64 bytes from 192.168.0.2: icmp_req=3 ttl=64 time=0.488 ms
```

- Podemos ter várias interfaces tap, cada uma plugada em um guest. Não é possível usar o mesmo dispositivo tap para mais de um guest.
- Atividade rápida:
  - Configure a rede em outra VM, da mesma maneira que a anterior porém com IPs 10.0.2.0/24, repetindo o procedimento acima com suas devidas altereações.

Configurando acesso ao mundo externo

- Para dar acesso externo para aos guests, devemos habilitar o encaminhamento de pacotes no host e fazer NAT.

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- Características gerais dessa configuração:
  - Guests não tem acesso ao segmento de rede físico.
  - O acesso ao meio externo precisa ser explicitamente configurado no host.
  - Não é necessário utilizar somente NAT, pode-se utilizar roteamento também.
  - Comunicação entre os guests precisa incluir rotas no host.

XXX Estabelecemos um link direto entre as VMs, fazer diagrama que mostra onde estão as filas de transmissão. Fila do guest, fila do qemu, fila do kernel.

Utilizando bridges

- Como facilitar a comunicação entre os guests? Usando bridges!
  - A bridge é uma maneira de conectar dois segmentos Ethernet, independentemente de protocolo.
  - Os quadros são encaminhados com base no endereço Ethernet, ao invés de endereços IP (como um roteador).
  - Uma vez que o encaminhamento é feito na camada 2, todos os protocolos podem circular de forma transparente através de uma bridge.
  - O código de ponte Linux implementa um subconjunto do padrão ANSI / IEEE 802.1d.

Criando uma bridge

- Para criar uma bridge:

```
# brctl addbr br0
```

- Cria uma brigde lógica chamada br0.
- Sempre é necessário pelo menos uma instância lógica a fazer qualquer ponte em tudo.
- Você pode imaginar a bridge como um agregador de interfaces.
- Cada bridge é representada por uma nova interface de rede.

```
# ifconfig br0
br0       Link encap:Ethernet  HWaddr a2:87:29:42:07:e1
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Para remover uma bridge:

```
# brctl delbr br0
```

> Encare a bridge como um simples switch de camada 2 virtual, que conhece os endereços MAC de cada porta e faz o encaminhamento de quadros porta-a-porta.

Adicionando interfaces a uma bridge

```
# brctl addif br0 tap0
```

- Acrescenta a interface de rede tap0 à brigde br0.
- Todas as interfaces contidas em uma bridge agem como em uma mesmo segmento de rede.
- Não é possível adicionar uma mesma interface em várias bridges.
- Quando uma interface é adicionada a bridge, ela leva um tempo para aprender a endereço Ethernet antes de começar a transmitir.
- Vendo as interfaces em uma bridge:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.baac3fccc978       no              tap0
```

Usando a bridge

- Agora vamos utilizar a bridge. Primeiramente, temos que colocar as interfaces tap dentro da bridge:

```
# brctl addif br0 tap0
# brctl addif br0 tap1
```

- Feito isso, inicie duas máquinas virtuais, cada uma utilizando uma interface tap. Coloque endereços MAC diferentes para as interfaces de cada máquina virtual.
- Configure ambas as máquinas virtuais com IPs pertencentes à mesma rede, ex: 192.168.0.2/24 e 192.168.0.3/24

No HOST devemos iniciar a bridge

```
# ifconfig br0 up
```

- De dentro das máquinas virtuais, tente pingá-las entre si.

Acesso externo com a bridge+NAT

- Nesse momento as interfaces das VMs estão isoladas dentro da bridge, sem nenhuma possibilidade de acesso externo.
- Para que as VMs possam "sair", precisamos colocar um IP na própria bridge.

```
# ifconfig br0 192.168.0.1 up
```

- As bridges tem uma porta especial que pode receber um IP e está atrelada ao host, realmente como uma "saída".
- Tente agora, a partir do host, pingar uma das VMs.
- Se estiver tudo funcionando, nas VMs coloque o IP da 192.168.0.1 como rota padrão:

```
# route add default gw 192.168.0.1
```

- Lembre-se que deixamos o **ip_forward** a regra de **NAT do iptables** configurados, portanto de dentro das VMs já será possível acessar o mundo externo.
- Características dessa configuração:
  - Interfaces de rede das VMs estão no mesmo segmento de camada 2.
  - O tráfego entre as VMs acontece totalmente dentro do kernel do host.
  - Acesso de fora do host para dentro só é possível com regras de iptables.
  - Pode-se utilizar roteamento IP também, ao invés de NAT. (tente fazer como exercício)

Acesso ao segmento de rede físico com

bridges

- Uma outra opção de configuração é colocar as interfaces de rede virtuais em contato praticamente "direto" com o meio físico. Para isso, precisamos colocar a interface eth0 do host dentro da bridge.
- Tire o IP da bridge e tire o IP da interface eth0. **Antes, anote qual é o IP de sua máquina!**

```
# ifconfig br0 0.0.0.0
# ifconfig eth0 0.0.0.0
```

- Agora colocamos a interface eth0 na bridge e nela configuramos também o IP que estava em eth0.

```
# brctl addif br0 eth0
# ifconfig br0 10.1.1.X
```

- Vamos desativar o NAT e o ip_forward.

```
# iptables -t nat -F
# echo 0 > /proc/sys/net/ipv4/ip_forward
```

- Verifique a conectividade do host:

```
# ping 10.1.1.1
```

- Configure suas VMs com IPs da rede 10.1.1.0/24 e de dentro da VM tente pingar 10.1.1.1.

Mais sobre bridges

- Uma vez que a bridge está em funcionamento, o parâmetro showmacs mostra informações sobre os endereços de rede e de tráfego que está sendo encaminhado.

```
# brctl showmacs br0
port no mac addr                is local?       ageing timer
  3     00:16:3e:57:13:a5       no                18.28
  2     00:16:3e:57:20:a5       no                18.28
  1     00:16:3e:57:20:b5       no                59.78
```

- O tempo de envelhecimento (ageing time) é o número de segundos que um endereço MAC será mantido no banco de dados de encaminhamento, depois de ter recebido um pacote a partir deste endereço MAC.

- As entradas no banco de dados de encaminhamento são periodicamente apagadas para garantir que não ficarão para sempre.

Arquivos de configuração

- CentOS/RHEL

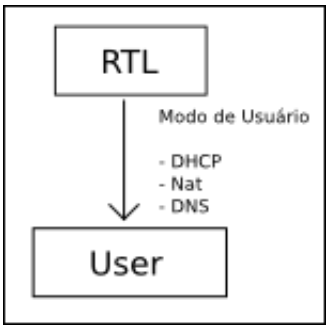
```
# cd /etc/sysconfig/network-scripts/
# cat ifcfg-br0
DEVICE=br0
ONBOOT=yes
BOOTPROTO=static
IPADDR=143.106.7.3
NETMASK=255.255.255.192
NO_ALIASROUTING=yes
GATEWAY=143.106.7.1
TYPE=Bridge

# cat ifcfg-eth0
# Intel Corporation 80003ES2LAN Gigabit Ethernet Controller (Copper)
DEVICE=eth0
ONBOOT=yes
TYPE=Ethernet
HWADDR=00:15:17:4E:CA:DE
BRIDGE=br0
BOOTPROTO=none
```

- Debian / Ubuntu

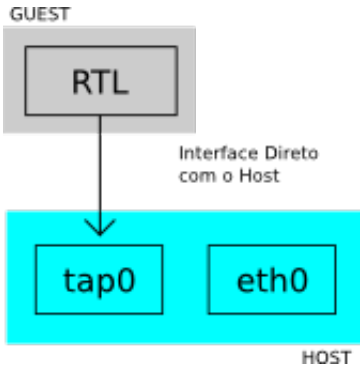
```
# cat /etc/network/interfaces
auto br0
iface br0 inet static
        address 192.168.2.50
        netmask 255.255.255.0
        gateway 192.168.2.1
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
```

Cenários

Modo usuário



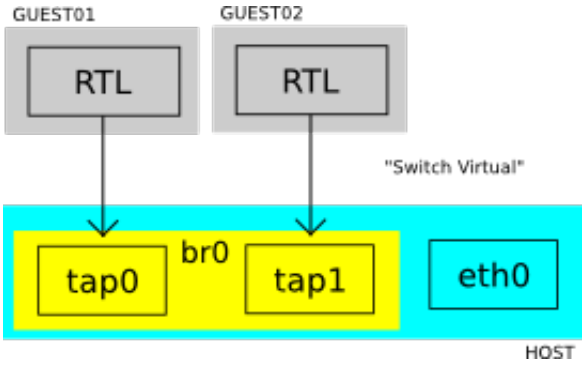
Acesso direto entre Host e Guest



Exemplo:

Host_eth0	143.106.16.1
tap0	192.168.0.1
Guest01	192.168.0.2

Acesso entre 2 Guest e roteamento através do Host

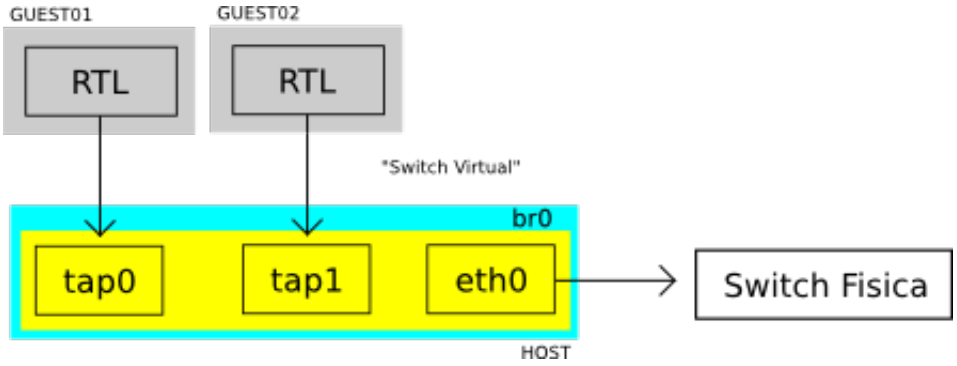


Exemplo:

Host_eth0	143.106.16.1
Bridge	192.168.0.1

Guest01	192.168.0.2
Guest02	192.168.0.3

Acesso dos 2 Guest e Host diretamente na rede física.



Exemplo:

Host_br0	143.106.16.1
Guest01	143.106.16.2
Guest02	143.106.16.3

Referências

- Esquema de rede interno do QEMU: http://people.gnome.org/~markmc/qemu-networking.html
- Gerador de endereços MAC: http://www.easyvmx.com/software/easymac.sh
- Documentação oficial sobre bridges no Linux: http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge
- Análise de performance de bridge do Linux: http://facweb.cti.depaul.edu/jyu/Publications/Yu-Linux-TSM2004.pdf
- Documentação do kernel sobre interfaces tap
- Interface tap centos: http://www.rootninja.com/create-a-network-bridge-for-virtual-machines