Ćwiczenie 2 - interfejsy programowe i sprzętowe

Szymon Chmal
Grupa 1 (wt 9:15)

23/04/2018

Spis treści

1 Cel ćwiczenia

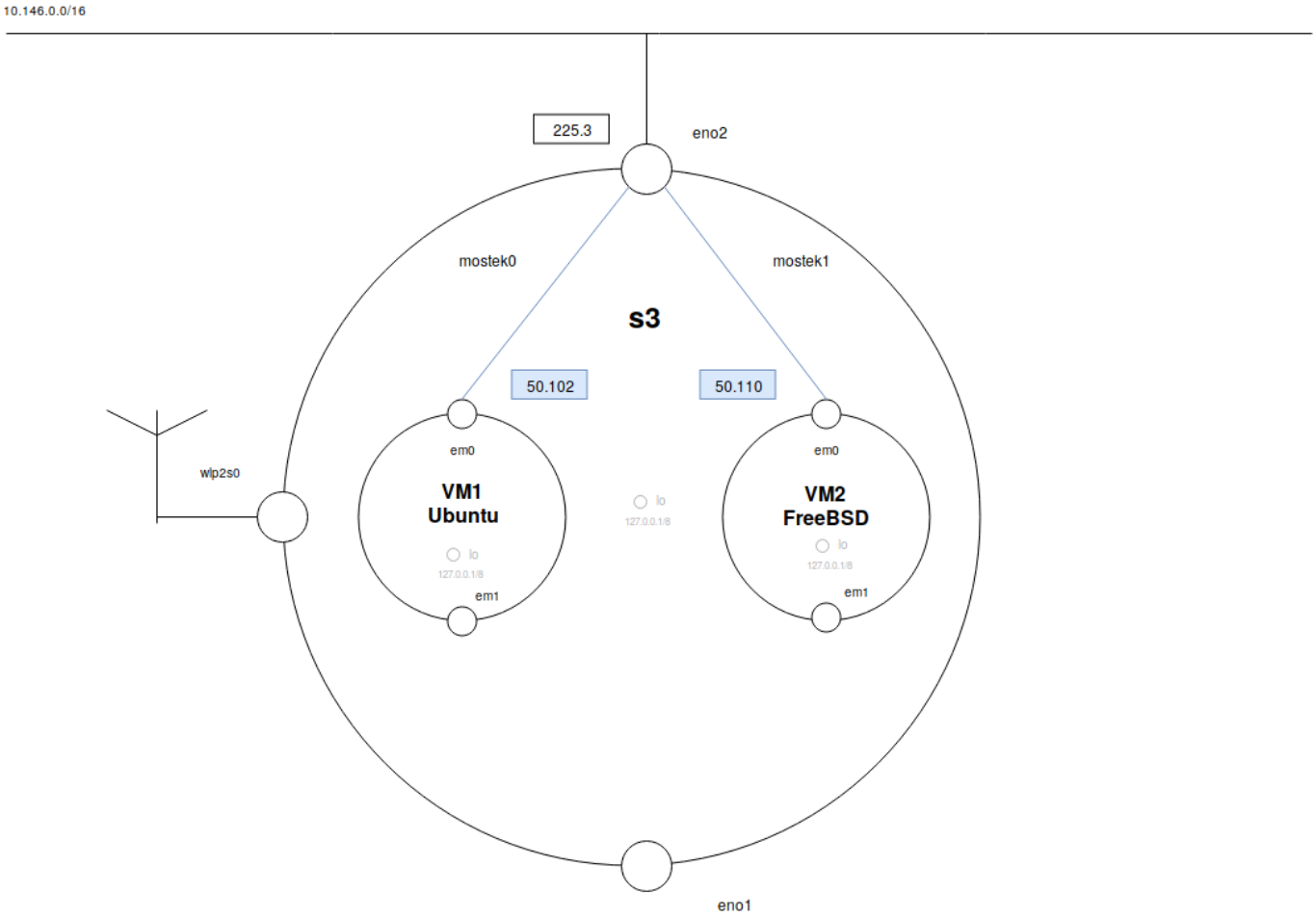
Celem ćwiczenia jest stworzenie maszyn wirtualnych opartych o systemy Ubuntu i FreeBSD, a następnie skonfigurowanie interfejsów sieciowych w zadany sposób, tzn. stworzenie mostka, podłączenie do niego interfejsu fizycznego, wykreowanego interfejsu tap i stworzenie dowolnego innego interfejsu w dowolnym połączeniu. Dodatkowym celem było stworzenie diagramów przedstawiających budowę sieci.

Warunki postawione w zadaniu:

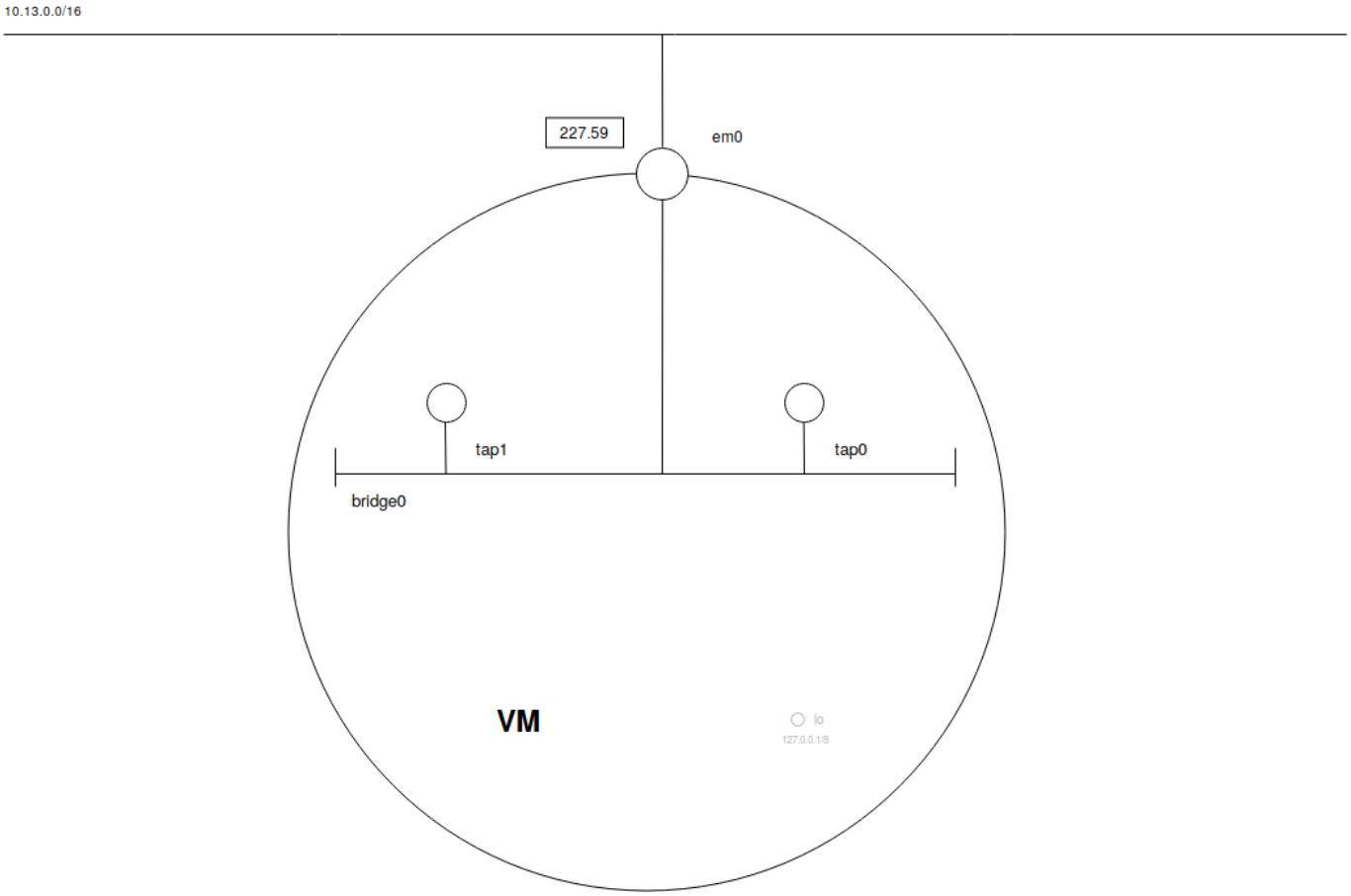
- Zawieranie mostka
- Zawieranie interfejsu tap
- Zawieranie dowolnego interfejsu
- Podłączenia do mostka stworzonych interfejsów i interfejsu fizycznego

2 Schematy sieci

2.1 Sieć stacji roboczej



2.2 Sieć maszyny wirtualnej



3 Podjęte kroki

3.1 Instalacja VirtualBox'a

Do zainstalowania oprogramowania VirtualBox użyłem wcześniej przygotowanego przez siebie skryptu:

```
sudo DEBIAN_FRONTEND=noninteractive apt update
sudo DEBIAN_FRONTEND=noninteractive apt install -y virtualbox-ext-pack
```

Za pomocą polecenia `dpkg -l` sprawdziłem, że VirtualBox został zainstalowany.

```
chmals@s3:~$ dpkg -l | grep virtualbox
ii  virtualbox                          5.1.34-dfsg-0ubuntu1.17.10.2           amd64        x86
     virtualization solution - base binaries
ii  virtualbox-dkms                     5.1.34-dfsg-0ubuntu1.17.10.2           all          x86
     virtualization solution - kernel module sources for dkms
ii  virtualbox-ext-pack                 5.1.34-0ubuntu1.17.10.2                all
    extra capabilities for VirtualBox, downloader.
ii  virtualbox-guest-additions-iso      5.1.34-0ubuntu1.17.10.1                all
    guest additions iso image for VirtualBox
ii  virtualbox-qt                       5.1.34-dfsg-0ubuntu1.17.10.2           amd64        x86
     virtualization solution - Qt based user interface
```

3.2 Stworzenie maszyn wirtualnych

Do stworzenia maszyn wirtualnych użyłem napisany przez siebie skrypt używający VBoxManage.

```
#!/bin/sh

ISO_BSD="/pub/FreeBSD/zetis/freebsd-mfs-12.0-CURRENT-amd64.iso"
```

```
ISO_UBU="/pub/Linux/Ubuntu/ubuntu-16.04-desktop-amd64-zetis.iso"
NAME_UBU="C3-Ubuntu"
NAME_BSD="C3-FreeBSD"

VBoxManage createvm --name $NAME_BSD --ostype "FreeBSD_64" --register
VBoxManage modifyvm $NAME_BSD --pae off
VBoxManage storagectl $NAME_BSD --name "IDE" --add ide
VBoxManage storageattach $NAME_BSD --storagectl "IDE" --port 0 \
--device 0 --type dvddrive --medium $ISO_BSD
VBoxManage modifyvm $NAME_BSD --boot1 dvd --boot2 none --boot3 none --boot4 none
VBoxManage modifyvm $NAME_BSD --memory 2048 --vram 12
VBoxManage modifyvm $NAME_BSD --nic1 bridged --bridgeadapter1 "eno1"
VBoxManage modifyvm $NAME_BSD --usb on --usbehci on

echo "Maszyna FreeBSD zostala utworzona!"

VBoxManage createvm --name $NAME_UBU --ostype $ostype "Ubuntu_64" --register
VBoxManage modifyvm $NAME_UBU --pae off
VBoxManage storagectl $NAME_UBU --name "IDE" --add ide
VBoxManage storageattach $NAME_UBU --storagectl "IDE" --port 0 \
--device 0 --type dvddrive --medium $ISO_UBU
VBoxManage modifyvm $NAME_UBU --boot1 dvd --boot2 none --boot3 none --boot4 none
VBoxManage modifyvm $NAME_UBU --memory 2048 --vram 12
VBoxManage modifyvm $NAME_UBU --nic1 bridged --bridgeadapter1 "eno1"
VBoxManage modifyvm $NAME_UBU --usb on --usbehci on

echo "Maszyna Ubuntu zostala utworzona!"
```

W celu sprawdzenia, czy maszyny zostały utworzone użyłem polecenia `VBoxManage list vms`.

```
"C3-FreeBSD" {ee093da3-29b4-44a7-bdfa-06aa49f4fa72}
"C3-Ubuntu" {6dfad5bb-3f31-4a84-9e01-304f878946c3}
```

Wynik polecenia wskazywał, że skrypt wykonał swoje zadanie.

3.3 Generowanie sieci - FreeBSD

W przypadku maszyny z systemem FreeBSD został użyty następujący skrypt.

```
#!/bin/sh

mostek=$(sudo ifconfig bridge create)
tap1=$(sudo ifconfig tap create)
tap2=$(sudo ifconfig tap create)
echo "Wygenerowano $mostek, $tap1, $tap2"
ifconfig $mostek addm $tap1 addm $tap2 addm em0
echo "Podlaczono interfejsy do $mostek"
```

Wynikiem działania skryptu jest utworzenie mostka i dwóch interfejsów tap. Następnie do mostka dodawane są utworzone interfejsy programowe i interfejs fizyczny em0. W celu sprawdzenia poprawności działania przed wykonaniem skryptu zapisałem wynik polecenia `ipconfig` do pliku `conf-before`, a następnie porównałem nową konfigurację ze starą.

```
root@:~ # ifconfig | diff conf.old -

< em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
<       options=85059b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,TSO4,LRO,VLAN_HWFILTER,
    VLAN_HWTSO>
---
> enp: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
>       options=850099<RXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HWFILTER,VLAN_HWTSO>
21a22,43
> tap0: flags=8903<UP,BROADCAST,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
>       options=80000<LINKSTATE>
```

```
>      ether 00:bd:9a:2f:f7:00
>      nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
>      media: Ethernet autoselect
>      status: no carrier
>      groups: tap
> tap1: flags=8903<UP,BROADCAST,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
>      options=80000<LINKSTATE>
>      ether 00:bd:9a:2f:f7:01
>      nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
>      media: Ethernet autoselect
>      status: no carrier
>      groups: tap
>      groups: tun
> bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
>      ether 02:85:09:b6:b5:00
>      nd6 options=9<PERFORMNUD,IFDISABLED>
>      groups: bridge
>      id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
>      maxage 20 holdcnt 6 proto rstp maxaddr 2000 timeout 1200
>      root id 00:00:00:00:00:00 priority 32768 ifcost 0 port 0
>      member: em0 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
>              ifmaxaddr 0 port 1 priority 128 path cost 20000
>      member: tap0 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
>              ifmaxaddr 0 port 3 priority 128 path cost 2000000
>      member: tap1 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
>              ifmaxaddr 0 port 4 priority 128 path cost 2000000
```

Wynik polecenia wskazuje na to, że faktycznie zostały utworzone zadane interfejsy i podłączone do mostka.

3.4 Generowanie sieci - maszyna Ubuntu

W przypadku maszyny Ubuntu wymagane było napisanie drugiego skryptu w związku z brakiem zainstalowanego pakietu odpowiedzialnego za polecenie ifconfig i z rychłym jego porzuceniem.

```
#!/bin/sh
# Ubuntu

# Tworzenie interfejsow
sudo ip link add name mostek type bridge
sudo ip tuntap add dev tap0 mode tap
sudo ip tuntap add dev tap1 mode tap
echo "Stworzono mostek i 2 interfejsy typu tap"

# Dodawanie interfejsow do mostka
sudo ip link set em0 master mostek
sudo ip link set tap0 master mostek
sudo ip link set tap1 master mostek
echo "Dolaczono do mostka interfejsy typu tap i interfejs fizyczny"
```

Postępując podobnie, tzn. zapisując do pliku wyniki poleceń `ip link show` oraz `bridge link` przed wykonaniem skryptu i porównując z nową sytuacją otrzymałem potwierdzenie poprawnego wykonania zadania

```
root@:~$ ip link show | diff ip_link.old -
3c3
< 2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
    default qlen 1000
---
> 2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master mostek state UP mode
    DEFAULT group default qlen 1000
6a7,12
> 6: mostek: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen
     1000
>     link/ether 56:25:6d:df:ce:43 brd ff:ff:ff:ff:ff:ff
```

```
> 7: tap0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop master mostek state DOWN mode DEFAULT group
    default qlen 1000
>     link/ether 5e:7b:c6:95:7f:4b brd ff:ff:ff:ff:ff:ff
> 8: tap1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop master mostek state DOWN mode DEFAULT group
    default qlen 1000
>     link/ether 56:25:6d:df:ce:43 brd ff:ff:ff:ff:ff:ff

aitwar@ubuntu-ge72:~/c2$ bridge link | diff bridge_link.old -
0a1,3
> 2: enp3s0 state UP : <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master mostek state disabled
    priority 32 cost 19
> 7: tap0 state DOWN : <BROADCAST,MULTICAST> mtu 1500 master mostek state disabled priority 32
    cost 100
> 8: tap1 state DOWN : <BROADCAST,MULTICAST> mtu 1500 master mostek state disabled priority 32
    cost 100
```

3.5 Czyszczenie

Aby usunąć stworzone interfejsy należy użyć następujących skryptów:

```
#!/bin/sh
# FreeBSD

# Usuwanie interfejsow z mostka
ifconfig bridge0 deletem tap0
ifconfig bridge0 deletem tap1
ifconfig bridge0 deletem em0
echo "Interfejsy zostaly wypiete z mostka!"

# Usuwanie interfejsow
ifconfig bridge0 destroy
ifconfig tap0 destory
ifconfig tap1 destroy
echo "Interfejsy zostaly usuniete!"
```

```
#!/bin/sh
# Ubuntu

# Usuwanie interfejsow z mostka
sudo ip link set dev tap0 nomaster
sudo ip link set dev tap1 nomaster
sudo ip link set dev enp3s0 nomaster
echo "Interfejsy zostaly wypiete z mostka!"

# Usuwanie interfejsow
sudo ip tuntap del dev tap0 mode tap
sudo ip tuntap del dev tap1 mode tap
sudo ip link delete br0 type bridge
echo "Interfejsy zostaly usuniete!"
```

4 Podsumowanie

Maszyny wirtualne są użytecznym środowiskiem m. in. dydaktycznym. Dzięki ich zastosowaniu osoba ćwicząca konfiguracje sieci nie musi obawiać się uszkodzenia istniejącej już konfiguracji. Nieocenione okazały się skrypty instalujące VirtualBox'a i tworzące maszyny wirtualne według zadanych wymagań. Dzięki nim praca stała się szybsza i bardziej ergonomiczna. Przydatne okazało się stworzenie schematu połączeń sieciowych, aby uniknąć błędów w konfiguracji.