Redes Definidas por Software

Relatório Lab 1

Namespaces

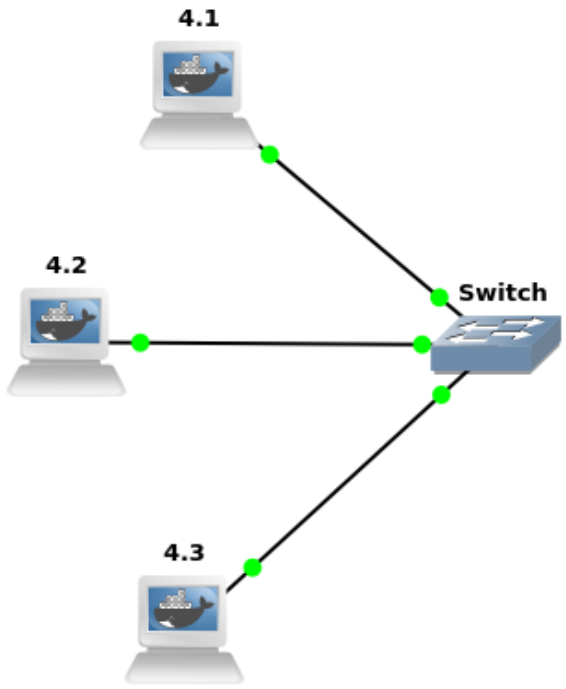
Virtual networking using Linux namespaces

Grupo 6
Miguel Carreiro, nº 82012

Ambiente de teste

Para executar os testes ao bom funcionamento dos comandos, foi usado o programa GNS3, em que foi emulado uma instância de uma distribuição Linux (Ubuntu), usando para tal um Docker container por alínea (o Dockerfile usado para a criação da imagem usada encontra-se na secção dos anexos).

O diagrama da rede é o seguinte:



Os containers foram ligados a um switch, no entanto, apenas serviu para permitir o arranque dos mesmos, não tendo qualquer influência nos resultados obtidos.

Para a análise da conectividade, foram usados os comandos `ping` (para efetuar o ping entre as várias interfaces virtuais), e o `ip route` (para analisar as rotas de cada um dos namespaces). Por vezes, foi usado o comando `netstat -r`, no entanto, este comando apresenta praticamente a mesma informação do ip route, apenas alterando o esquema de apresentação no terminal.

Ao efetuar o ping entre duas interfaces, é enviado um pacote ICMP echo para o endereço IP de destino, e, caso o recetor o receba corretamente, envia um pacote ICMP echo reply para o endereço IP que enviou o pacote original (neste caso, o

ICMP echo). Para que o ping funcione sem problemas, é necessário que a rota entre a source (que realiza o ping) e o target (host cuja conectividade está a ser testada através do ping), bem como a rota inversa, entre o target e a source, não tenham qualquer problema, isto é, entre outros fatores, que as suas tabelas de encaminhamento estejam bem configuradas (nas duas direções). Isto significa que, ao efetuar o ping, estamos a testar a configuração das tabelas de encaminhamento em ambas as direções. Uma falha na configuração das tabelas de encaminhamento numa das direções poderá inviabilizar a execução do ping.

Exercício 4.1

Configuração da rede

O ambiente de teste foi configurado com os parâmetros descritos no enunciado (endereços IP, namespaces, e o nome das interfaces). Os comandos usados encontram-se nos anexos.

A configuração é iniciada pela criação do namespace 'examplens', seguida da criação das interfaces virtuais, e respetiva associação aos namespaces correspondentes. Por último, os endereços IP são configurados nas interfaces virtuais, e as mesmas são iniciadas através do comando `ip link set dev <nome_interface> up`.

Após as configurações dos namespaces e das interfaces virtuais, procedeu-se aos testes de funcionamento e conectividade entre as interfaces dos diferentes namespaces.

Testes de funcionamento

Ao listar as rotas de encaminhamento, usando o comando ip route, temos os seguintes resultados:

```
root@4:~# ip route
10.0.0.0/24 dev external proto kernel scope link src 10.0.0.1
```

A tabela de encaminhamento do namespace padrão possui a rota para a gama de endereços 10.0.0.0/24, criada automaticamente aquando a atribuição do endereço da interface external.

Em relação à tabela de encaminhamento do namespace 'examplens', o resultado é o seguinte:

```
root@4:~# ip netns exec examplens ip route
10.0.0.0/24 dev internal proto kernel scope link src 10.0.0.2
```

Podemos constatar que a gama de endereços 10.0.0.0/24 também se encontra presente na tabela, com o devido encaminhamento para a interface 'internal'.

Para verificar que os dois namespaces se encontram comunicáveis, foi efetuado um ping desde o namespace padrão para a interface 'internal' (endereço 10.0.0.2), pertencente ao namespace 'examplens'. O resultado obtido foi o seguinte:

```
root@4:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.036 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.069 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3070ms
rtt min/avg/max/mdev = 0.035/0.049/0.069/0.014 ms
```

O ping foi executado com sucesso, confirmando assim que as duas interfaces virtuais se encontram comunicáveis. Ainda assim, para assegurar a veracidade desta afirmação, foi efetuado outro ping, no entanto, desta vez foi originado a partir do namespace 'examplens', e teve como destino a interface 'external' (endereço 10.0.0.1). A captura do resultado é a seguinte:

```
root@4:~# ip netns exec examplens ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.054 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
rtt min/avg/max/mdev = 0.038/0.048/0.054/0.006 ms
```

Como podemos observar, o ping também foi bem sucedido, confirmando assim que a interface 'external', pertencente ao namespace padrão, também se encontra acessível a partir do namespace 'examplens'.

Exercício 4.2

Configuração da rede

À semelhança do exercício anterior, os endereços IP das interfaces e o seu nome, bem como os namespaces, são definidos no enunciado, tendo configurado assim o ambiente de teste com os dados fornecidos. Os comandos usados encontram-se nos anexos.

O script de configuração encontra-se dividido em 4 secções. As interfaces 'one' e 'three' são criadas na secção correspondente ao namespace da interface que se encontra ligada ('near' e 'far' respetivamente):

- Near namespace - Nesta secção encontram-se os comandos para a configuração do namespace 'near', bem como o par de interfaces 'one' (localizado no namespace padrão), e 'two', localizada no namespace 'near', incluindo também as correspondentes atribuições dos endereços IP, bem como a ativação das interfaces;
- Far namespace - Nesta secção encontram-se os comandos para a configuração do namespace 'far', bem como a configuração do par de interfaces 'three' e four', bem como a interface 'lo' (loopback);
- Routing rules - Nesta secção são inseridas as regras de encaminhamento que permitem o correto encaminhamento dos pacotes entre os namespaces;
- Activate IP forwarding for namespaces - Por último, nesta secção encontram-se os comandos para ativar a capacidade do sistema encaminhar pacotes para uma rede diferente (neste caso, para um namespace diferente). Esta ativação tem de ser efetuada em cada um dos namespaces existentes.

Testes de funcionamento

A tabela de encaminhamento do namespace padrão resultante é a seguinte:

```
root@4:~# ip route
10.0.11.0/24 dev one proto kernel scope link src 10.0.11.1
10.0.12.0/24 via 10.0.11.2 dev one
10.0.13.0/24 via 10.0.11.2 dev one
```

Podemos observar que os pacotes nas gamas de endereços 10.0.12.0/24 e 10.0.13.0/24 são redirecionados para a interface 'one', tendo como next-hop o endereço da interface 'two', já no namespace 'near'. É de notar que os namespaces não necessitam de conhecer a rota completa para o endereço de destino, mas

apenas o endereço IP e a interface do next-hop, uma vez que o namespace 'near' possui a informação suficiente (será falado mais à frente) para encaminhar os pacotes para o namespace 'far'.

Acerca do namespace 'near', a tabela de encaminhamento é a seguinte:

```
root@4:~# ip netns exec near ip route
10.0.11.0/24 dev two proto kernel scope link src 10.0.11.2
10.0.11.2 via 10.0.11.2 dev two
10.0.12.0/24 dev three proto kernel scope link src 10.0.12.1
10.0.12.1 via 10.0.12.1 dev three
10.0.13.0/24 via 10.0.12.2 dev three
```

Podemos observar que o namespace possui as regras de encaminhamento para o endereços do namespace padrão (10.0.11.1), bem como para os endereços do namespace 'far' (10.0.12.2 e 10.0.13.1). Também é possível observar que os endereços 10.0.11.2 (interface 'two') e 10.0.12.1 (interface 'three') possuem regras específicas de encaminhamento, uma vez que o next-hop das regras da gama de endereços a qual pertencem têm como next-hop o endereço da interface oposta, o que significa que, caso estas regras não fossem adicionadas, não seria possível comunicar com as interfaces 'two' e 'three' a partir do namespace 'near' (ainda assim, as restantes interfaces permaneceriam comunicáveis, independentemente da adição destas regras específicas).

Por último, a tabela de encaminhamento do namespace 'far' é a seguinte:

```
root@4:~# ip netns exec far ip route
10.0.11.0/24 via 10.0.12.1 dev four
10.0.12.0/24 dev four proto kernel scope link src 10.0.12.2
```

Podemos observar que, a partir deste namespace, também é possível chegar aos restantes namespaces.

Segue-se os resultados obtidos após a realização dos pings entre as interfaces, começando pelo ping entre o namespace padrão e a interface loopback do namespace 'far':

```
root@4:~# ping 10.0.13.1
PING 10.0.13.1 (10.0.13.1) 56(84) bytes of data.
64 bytes from 10.0.13.1: icmp_seq=1 ttl=63 time=0.167 ms
64 bytes from 10.0.13.1: icmp_seq=2 ttl=63 time=0.054 ms
64 bytes from 10.0.13.1: icmp_seq=3 ttl=63 time=0.102 ms
64 bytes from 10.0.13.1: icmp_seq=4 ttl=63 time=0.071 ms
^C
--- 10.0.13.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.054/0.098/0.167/0.043 ms
```

O ping foi realizado com sucesso, o que comprova que existe conectividade entre o namespace padrão e o namespace 'far'.

Também foram realizados pings entre os namespaces 'far' e 'near' (interface 'two'), e entre o namespace padrão e o namespace 'near' (interface 'three'). Os resultados foram os que se seguem:

Ping 'far' - 'two':

```
root@4:~# ip netns exec far ping 10.0.11.2
PING 10.0.11.2 (10.0.11.2) 56(84) bytes of data.
64 bytes from 10.0.11.2: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 10.0.11.2: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 10.0.11.2: icmp_seq=3 ttl=64 time=0.057 ms
64 bytes from 10.0.11.2: icmp_seq=4 ttl=64 time=0.064 ms
^C
--- 10.0.11.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.044/0.058/0.070/0.009 ms
```

Ping default - 'three':

```
root@4:~# ping 10.0.12.1
PING 10.0.12.1 (10.0.12.1) 56(84) bytes of data.
64 bytes from 10.0.12.1: icmp_seq=1 ttl=64 time=0.058 ms
64 bytes from 10.0.12.1: icmp_seq=2 ttl=64 time=0.112 ms
64 bytes from 10.0.12.1: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 10.0.12.1: icmp_seq=4 ttl=64 time=0.087 ms
^C
--- 10.0.12.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3071ms
rtt min/avg/max/mdev = 0.058/0.081/0.112/0.020 ms
```

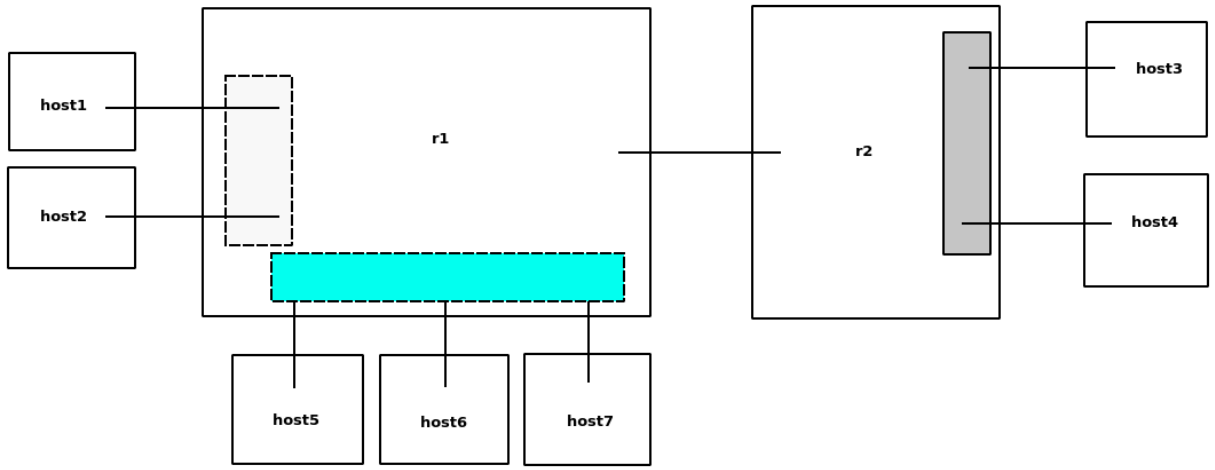
Os resultados demonstram a total conectividade entre os namespaces.

Exercício 4.3

Configuração da rede

De forma a simplificar a topologia da rede do exercício 4.3, em vez de ser criado um namespace por equipamento, foram criados apenas os namespaces correspondentes aos hosts, e os namespaces correspondentes aos routers, eliminando assim os namespaces correspondentes aos switches das redes.

Ainda assim, os endereços atribuídos às redes que se interligam com o router R1 mantêm-se separados entre si (isto é, existem duas gamas de endereços /24, correspondentes às duas redes existentes). A topologia da rede criada é a seguinte:



Para facilitar a identificação das redes, foi atribuído um número a cada uma das redes:

- Cinzento claro - Rede 1
- Cinzento escuro - Rede 2
- Azul claro - Rede 3

Cada quadrado corresponde a um namespace, e o texto inserido dentro de cada um deles corresponde ao nome atribuído ao mesmo.

A atribuição dos nomes das interfaces e dos endereços IP foi feita com base no número do host e no número da rede. As interfaces dos hosts têm o nome vhostX.Y (X - número da rede, Y - número do host*10), as interfaces dos namespaces dos routers e que ligam aos hosts têm o nome vethX.Y, e as interfaces que interligam os namespaces r1 e r2 têm o nome W.Z (W - namespace onde se localiza a interface, Z - namespace que interliga).

Os endereços das interfaces inseridas nos namespaces dos hosts têm o formato 10.0.X.Y (X - número da rede, Y - número do host*10+1), os endereços das

interfaces que estão inseridas no namespace dos routers e que interligam a um host têm o formato 10.0.X.Y' (X - número da rede, Y' - número do host*10), e, por último, as interfaces r1.r2 e r2.r1 têm, respetivamente, os endereços 10.0.12.1/24 e 10.0.12.2/24. Os restantes endereços e namespaces encontram-se detalhados nos anexos.

Acerca das rotas de encaminhamento, foi necessário proceder à eliminação das rotas criadas por defeito, pois o endereço de destino predefinido corresponde à gama de endereços do host, e não apenas ao endereço específico do mesmo, o que poderia levar a problemas no encaminhamento dos pacotes que circulam dentro da mesma rede. Assim, as rotas adicionadas no namespace dos routers correspondem aos endereços específicos de cada host, excepto nos casos em que é necessário encaminhar o tráfego para outro router, onde é adicionada uma regra para a gama de endereços completa, sendo o next-hop o router vizinho. Nos hosts, apenas foi configurada uma rota que permite o encaminhamento por defeito de todos os pacotes para o namespace do router a que se encontra conectado.

Testes de funcionamento

Para verificar o correto funcionamento da rede, foram efetuados pings entre cada uma das redes, bem como entre dois hosts de uma mesma rede. O resultado dos pings foi o seguinte:

Ping Net1-Net2 | Host1-Host3:

```
root@4:~# ip netns exec host1 ping 10.0.2.31
PING 10.0.2.31 (10.0.2.31) 56(84) bytes of data.
64 bytes from 10.0.2.31: icmp_seq=1 ttl=62 time=0.046 ms
64 bytes from 10.0.2.31: icmp_seq=2 ttl=62 time=0.114 ms
64 bytes from 10.0.2.31: icmp_seq=3 ttl=62 time=0.104 ms
64 bytes from 10.0.2.31: icmp_seq=4 ttl=62 time=0.055 ms
^C
--- 10.0.2.31 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.046/0.079/0.114/0.029 ms
```

Ping intra-net1 | Host1-Host:

```
root@4:~# ip netns exec host1 ping 10.0.1.21
PING 10.0.1.21 (10.0.1.21) 56(84) bytes of data.
64 bytes from 10.0.1.21: icmp_seq=1 ttl=63 time=0.079 ms
64 bytes from 10.0.1.21: icmp_seq=2 ttl=63 time=0.055 ms
64 bytes from 10.0.1.21: icmp_seq=3 ttl=63 time=0.135 ms
64 bytes from 10.0.1.21: icmp_seq=4 ttl=63 time=0.081 ms
^C
--- 10.0.1.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.055/0.087/0.135/0.029 ms
```

Ping net2-net1 | Host3-Host2:

```
root@4:~# ip netns exec host3 ping 10.0.1.21
PING 10.0.1.21 (10.0.1.21) 56(84) bytes of data.
64 bytes from 10.0.1.21: icmp_seq=1 ttl=62 time=0.057 ms
64 bytes from 10.0.1.21: icmp_seq=2 ttl=62 time=0.086 ms
64 bytes from 10.0.1.21: icmp_seq=3 ttl=62 time=0.102 ms
64 bytes from 10.0.1.21: icmp_seq=4 ttl=62 time=0.109 ms
^C
--- 10.0.1.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3055ms
rtt min/avg/max/mdev = 0.057/0.088/0.109/0.020 ms
```

Ping intra-net2 | Host3-Host4:

```
root@4:~# ip netns exec host3 ping 10.0.2.41
PING 10.0.2.41 (10.0.2.41) 56(84) bytes of data.
64 bytes from 10.0.2.41: icmp_seq=1 ttl=63 time=0.051 ms
64 bytes from 10.0.2.41: icmp_seq=2 ttl=63 time=0.071 ms
64 bytes from 10.0.2.41: icmp_seq=3 ttl=63 time=0.062 ms
64 bytes from 10.0.2.41: icmp_seq=4 ttl=63 time=0.105 ms
^C
--- 10.0.2.41 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3077ms
rtt min/avg/max/mdev = 0.051/0.072/0.105/0.020 ms
```

Ping net3-net1 | Host5-Host1:

```
root@4:~# ip netns exec host5 ping 10.0.1.11
PING 10.0.1.11 (10.0.1.11) 56(84) bytes of data.
64 bytes from 10.0.1.11: icmp_seq=1 ttl=63 time=0.045 ms
64 bytes from 10.0.1.11: icmp_seq=2 ttl=63 time=0.093 ms
64 bytes from 10.0.1.11: icmp_seq=3 ttl=63 time=0.079 ms
64 bytes from 10.0.1.11: icmp_seq=4 ttl=63 time=0.113 ms
^C
--- 10.0.1.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3071ms
rtt min/avg/max/mdev = 0.045/0.082/0.113/0.024 ms
```

Ping intra-net3 | Host6-Host7:

```
root@4:~# ip netns exec host6 ping 10.0.3.71
PING 10.0.3.71 (10.0.3.71) 56(84) bytes of data.
64 bytes from 10.0.3.71: icmp_seq=1 ttl=63 time=0.050 ms
64 bytes from 10.0.3.71: icmp_seq=2 ttl=63 time=0.084 ms
64 bytes from 10.0.3.71: icmp_seq=3 ttl=63 time=0.054 ms
64 bytes from 10.0.3.71: icmp_seq=4 ttl=63 time=0.077 ms
^C
--- 10.0.3.71 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3070ms
rtt min/avg/max/mdev = 0.050/0.066/0.084/0.014 ms
```

Ping net3-net2 | Host6-Host4:

```
root@4:~# ip netns exec host6 ping 10.0.2.41
PING 10.0.2.41 (10.0.2.41) 56(84) bytes of data.
64 bytes from 10.0.2.41: icmp_seq=1 ttl=62 time=0.106 ms
64 bytes from 10.0.2.41: icmp_seq=2 ttl=62 time=0.091 ms
64 bytes from 10.0.2.41: icmp_seq=3 ttl=62 time=0.080 ms
64 bytes from 10.0.2.41: icmp_seq=4 ttl=62 time=0.114 ms
^C
--- 10.0.2.41 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3068ms
rtt min/avg/max/mdev = 0.080/0.097/0.114/0.013 ms
```

Através dos resultados obtidos, é possível constatar que existe conectividade entre todas as redes, bem como intra-redes, cumprindo assim o objetivo proposto no enunciado.

Observando as rotas de encaminhamento do namespace r1, podemos observar que todos os endereços dos hosts podem ser alcançados com as regras existentes:

```
root@4:~# ip netns exec r1 ip route
10.0.1.11 dev veth1.10 scope link
10.0.1.21 dev veth1.20 scope link
10.0.2.0/24 via 10.0.12.2 dev r1.r2
10.0.3.51 dev veth3.50 scope link
10.0.3.61 dev veth3.60 scope link
10.0.3.71 dev veth3.70 scope link
10.0.12.0/24 dev r1.r2 proto kernel scope link src 10.0.12.1
```

Apesar da simplificação dos namespaces trazer benefícios na configuração da rede, esta abordagem de atribuição de rotas tem como desvantagem o facto de, cada vez que um host é adicionado à rede, torna-se necessário adicionar e/ou proceder à alteração das rotas criadas por defeito.

Anexos

Dockerfile - Gera a imagem dos containers usados neste laboratório

```
FROM ubuntu:latest

RUN DEBIAN_FRONTEND=noninteractive apt-get update -y && apt-get install -y
--no-install-recommends apt-utils && \
   apt-get install -y --no-install-recommends python3-pip python3-dev
build-essential net-tools iproute2 tcpdump \
   telnet traceroute curl iperf3 knot-host openssh-client mtr-tiny socat nano
vim-tiny nmap iputils-ping && \
   rm -rf /var/lib/apt/lists/*

VOLUME [ "/root" ]

WORKDIR /root
```

Exercício 4.1 - Comandos

```
ip netns add examplens
ip link add external type veth peer name internal
ip link set internal netns examplens

ip netns exec examplens ip addr add 10.0.0.2/24 dev internal
ip netns exec examplens ip link set dev internal up

ip addr add 10.0.0.1/24 dev external
ip link set dev external up
```

Exercício 4.2 - Comandos

```
# Near namespace

ip netns add near
ip link add one type veth peer name two
ip link set two netns near

ip netns exec near ip addr add 10.0.11.2/24 dev two
ip netns exec near ip link set dev two up

ip addr add 10.0.11.1/24 dev one
ip link set dev one up

# Far namespace

ip netns add far
ip link add three type veth peer name four
ip link set three netns near
ip link set four netns far

ip netns exec near ip addr add 10.0.12.1/24 dev three
ip netns exec near ip link set dev three up

ip netns exec far ip addr add 10.0.12.2/24 dev four
ip netns exec far ip link set dev four up

ip netns exec far ip addr add 10.0.13.1/24 dev lo
ip netns exec far ip link set dev lo up

# Routing rules

ip netns exec near ip route add 10.0.12.1 dev three via 10.0.12.1
ip netns exec near ip route add 10.0.11.2 dev two via 10.0.11.2
ip netns exec near ip route add 10.0.13.0/24 dev three via 10.0.12.2
ip netns exec far ip route add 10.0.11.0/24 dev four via 10.0.12.1

ip route add 10.0.12.0/24 dev one via 10.0.11.2
ip route add 10.0.13.0/24 dev one via 10.0.11.2

# Activate IP forwarding for namespaces

ip netns exec near sysctl -w net.ipv4.ip_forward=1
ip netns exec far sysctl -w net.ipv4.ip_forward=1
```

Exercício 4.3 - Plano de endereçamento

```
# Network 1

# Host 1

Namespace: host1

veth interface name | IP address
Net1 interface: vhost1.10 | 10.0.1.11/24

# Host 2

Namespace: host2

veth interface name | IP address
Net1 interface: vhost1.20 | 10.0.1.21/24

# Network 2

# Host 3

Namespace: host3

veth interface name | IP address
Net1 interface: vhost2.30 | 10.0.2.31/24

# Host 4

Namespace: host4

veth interface name | IP address
Net1 interface: vhost2.40 | 10.0.2.41/24

# Network 3

# Host 5

Namespace: host5
```

```
veth interface name | IP address
Net1 interface: vhost3.50 | 10.0.3.51/24

# Host 6

Namespace: host6

veth interface name | IP address
Net1 interface: vhost3.60 | 10.0.3.61/24

# Host 7

Namespace: host7

veth interface name | IP address
Net1 interface: vhost3.70 | 10.0.3.71/24

# Router 1

Namespace: r1

veth interface name | IP address
R2 interface: r1.r2 | 10.0.12.1/24
H1: veth1.10 | 10.0.1.10/24
H2: veth1.20 | 10.0.1.20/24
H5: veth3.50 | 10.0.3.50/24
H6: veth3.60 | 10.0.3.60/24
H7: veth3.70 | 10.0.3.70/24

# Router 2

Namespace: r2

veth interface name | IP address
R1 interface: r2.r1 | 10.0.12.2/24
H3: veth2.30 | 10.0.2.30/24
H4: veth2.40 | 10.0.2.40/24
```

Exercício 4.3 - Comandos

```
# Add R1 and R2 namespaces

ip netns add r1
ip netns exec r1 ip link set dev lo up

ip netns add r2
ip netns exec r2 ip link set dev lo up

# R1-R2 interface

ip link add r1.r2 type veth peer name r2.r1

ip link set r1.r2 netns r1
ip link set r2.r1 netns r2

ip netns exec r1 ip addr add 10.0.12.1/24 dev r1.r2
ip netns exec r1 ip link set dev r1.r2 up

ip netns exec r2 ip addr add 10.0.12.2/24 dev r2.r1
ip netns exec r2 ip link set dev r2.r1 up

# R1 namespace

# R1 - Net1 hosts configuration

ip netns add host1
ip link add veth1.10 type veth peer name vhost1.10
ip link set veth1.10 netns r1
ip link set vhost1.10 netns host1

ip netns exec host1 ip link set dev lo up

ip netns exec r1 ip addr add 10.0.1.10/24 dev veth1.10
ip netns exec r1 ip link set dev veth1.10 up

ip netns exec host1 ip addr add 10.0.1.11/24 dev vhost1.10
ip netns exec host1 ip link set dev vhost1.10 up
```

```
ip netns add host2
ip link add veth1.20 type veth peer name vhost1.20
ip link set veth1.20 netns r1
ip link set vhost1.20 netns host2

ip netns exec host2 ip link set dev lo up

ip netns exec r1 ip addr add 10.0.1.20/24 dev veth1.20
ip netns exec r1 ip link set dev veth1.20 up

ip netns exec host2 ip addr add 10.0.1.21/24 dev vhost1.20
ip netns exec host2 ip link set dev vhost1.20 up

# R1 - Net3 hosts configuration

ip netns add host5
ip link add veth3.50 type veth peer name vhost3.50
ip link set veth3.50 netns r1
ip link set vhost3.50 netns host5

ip netns exec host5 ip link set dev lo up

ip netns exec r1 ip addr add 10.0.3.50/24 dev veth3.50
ip netns exec r1 ip link set dev veth3.50 up

ip netns exec host5 ip addr add 10.0.3.51/24 dev vhost3.50
ip netns exec host5 ip link set dev vhost3.50 up

ip netns add host6
ip link add veth3.60 type veth peer name vhost3.60
ip link set veth3.60 netns r1
ip link set vhost3.60 netns host6

ip netns exec host6 ip link set dev lo up

ip netns exec r1 ip addr add 10.0.3.60/24 dev veth3.60
ip netns exec r1 ip link set dev veth3.60 up
```

```
ip netns exec host6 ip addr add 10.0.3.61/24 dev vhost3.60
ip netns exec host6 ip link set dev vhost3.60 up

ip netns add host7
ip link add veth3.70 type veth peer name vhost3.70
ip link set veth3.70 netns r1
ip link set vhost3.70 netns host7

ip netns exec host7 ip link set dev lo up

ip netns exec r1 ip addr add 10.0.3.70/24 dev veth3.70
ip netns exec r1 ip link set dev veth3.70 up

ip netns exec host7 ip addr add 10.0.3.71/24 dev vhost3.70
ip netns exec host7 ip link set dev vhost3.70 up

# R2 namespace

# R2 - Net2 hosts configuration

ip netns add host3
ip link add veth2.30 type veth peer name vhost2.30
ip link set veth2.30 netns r2
ip link set vhost2.30 netns host3

ip netns exec host3 ip link set dev lo up

ip netns exec r2 ip addr add 10.0.2.30/24 dev veth2.30
ip netns exec r2 ip link set dev veth2.30 up

ip netns exec host3 ip addr add 10.0.2.31/24 dev vhost2.30
ip netns exec host3 ip link set dev vhost2.30 up

ip netns add host4
ip link add veth2.40 type veth peer name vhost2.40
ip link set veth2.40 netns r2
ip link set vhost2.40 netns host4

ip netns exec host4 ip link set dev lo up
```

```
ip netns exec r2 ip addr add 10.0.2.40/24 dev veth2.40
ip netns exec r2 ip link set dev veth2.40 up

ip netns exec host4 ip addr add 10.0.2.41/24 dev vhost2.40
ip netns exec host4 ip link set dev vhost2.40 up

# Routing rules

ip netns exec r1 ip route add 10.0.2.0/24 dev r1.r2 via 10.0.12.2
ip netns exec r2 ip route add 10.0.1.0/24 dev r2.r1 via 10.0.12.1
ip netns exec r2 ip route add 10.0.3.0/24 dev r2.r1 via 10.0.12.1

ip netns exec r1 ip route add 10.0.1.11 dev veth1.10
ip netns exec host1 ip route add default dev vhost1.10 via 10.0.1.10

ip netns exec r1 ip route add 10.0.1.21 dev veth1.20
ip netns exec host2 ip route add default dev vhost1.20 via 10.0.1.20

ip netns exec r1 ip route add 10.0.3.51 dev veth3.50
ip netns exec host5 ip route add default dev vhost3.50 via 10.0.3.50

ip netns exec r1 ip route add 10.0.3.61 dev veth3.60
ip netns exec host6 ip route add default dev vhost3.60 via 10.0.3.60

ip netns exec r1 ip route add 10.0.3.71 dev veth3.70
ip netns exec host7 ip route add default dev vhost3.70 via 10.0.3.70

ip netns exec r2 ip route add 10.0.2.31 dev veth2.30
ip netns exec host3 ip route add default dev vhost2.30 via 10.0.2.30

ip netns exec r2 ip route add 10.0.2.41 dev veth2.40
ip netns exec host4 ip route add default dev vhost2.40 via 10.0.2.40

ip netns exec r1 ip route del 10.0.1.0/24 dev veth1.10 proto kernel
scope link src 10.0.1.10
ip netns exec r1 ip route del 10.0.1.0/24 dev veth1.20 proto kernel
scope link src 10.0.1.20
```

```
ip netns exec r1 ip route del 10.0.3.0/24 dev veth3.50 proto kernel
scope link src 10.0.3.50
ip netns exec r1 ip route del 10.0.3.0/24 dev veth3.60 proto kernel
scope link src 10.0.3.60
ip netns exec r1 ip route del 10.0.3.0/24 dev veth3.70 proto kernel
scope link src 10.0.3.70

ip netns exec host1 ip route del 10.0.1.0/24
ip netns exec host2 ip route del 10.0.1.0/24
ip netns exec host3 ip route del 10.0.2.0/24
ip netns exec host4 ip route del 10.0.2.0/24
ip netns exec host5 ip route del 10.0.3.0/24
ip netns exec host6 ip route del 10.0.3.0/24
ip netns exec host7 ip route del 10.0.3.0/24

# Activate IP forwarding for namespaces

ip netns exec r1 sysctl -w net.ipv4.ip_forward=1
ip netns exec r2 sysctl -w net.ipv4.ip_forward=1
ip netns exec host1 sysctl -w net.ipv4.ip_forward=1
ip netns exec host2 sysctl -w net.ipv4.ip_forward=1
ip netns exec host3 sysctl -w net.ipv4.ip_forward=1
ip netns exec host4 sysctl -w net.ipv4.ip_forward=1
ip netns exec host5 sysctl -w net.ipv4.ip_forward=1
ip netns exec host6 sysctl -w net.ipv4.ip_forward=1
ip netns exec host7 sysctl -w net.ipv4.ip_forward=1
```