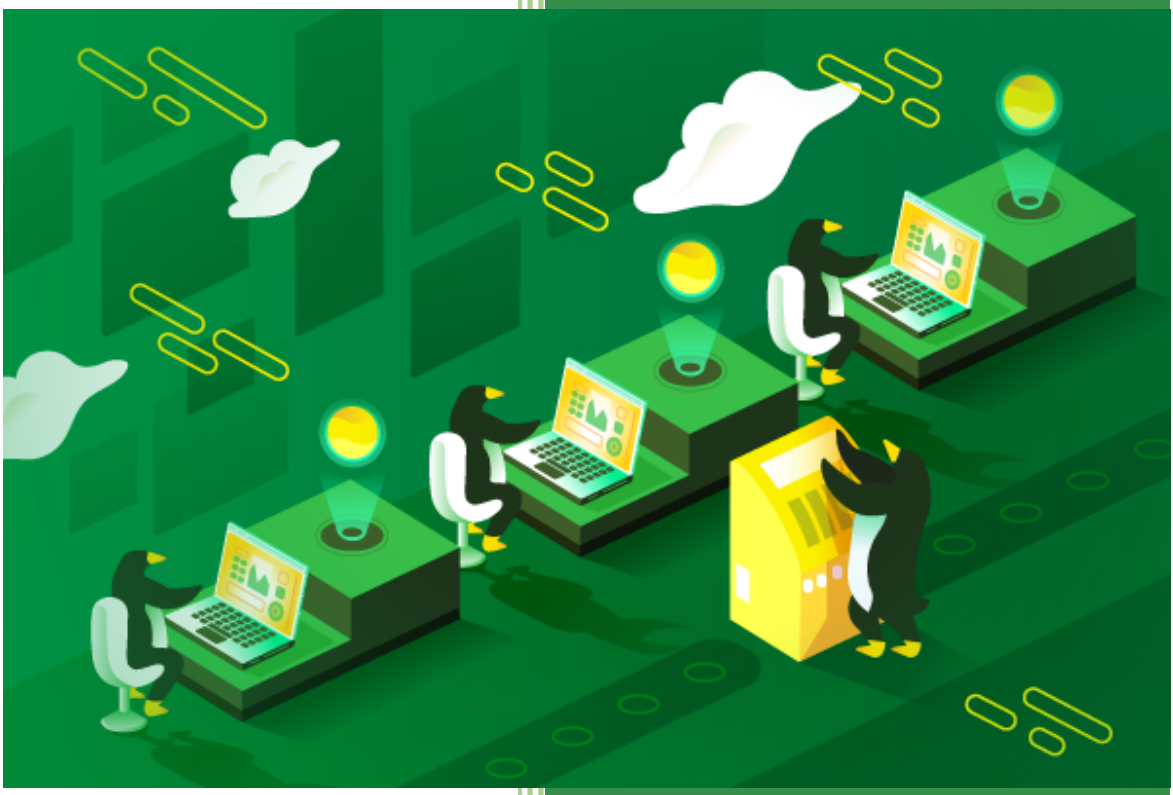
Linux Admin Pro

Configuração de Rede - Módulo 14



Material de Apoio

Professor: Vagner Fonseca

Índice

Configuração de Redes no Linux

As interfaces de rede no Linux podem ser configuradas via comandos ou em arquivos, via comandos as configurações geralmente são temporárias e não persistem na reinicialização, em arquivos as configurações são permanentes.

Comandos de Configuração de Rede

IFCONFIG

Comando ifconfig é utilizado para configurar as interfaces de rede, desde configurar ip, máscara, broadcast até Mac Address. É um comando obsoleto e muitas distros nem colocam mais em sua instalação principal. Apresentado para conhecimento caso o encontre em alguma distro mais antiga. Considerado obsoleto e substituído pelo comando IP.

A sintaxe do comando é a seguinte:
ifconfig <interface> <opções>

Pode ser utilizado da seguinte forma para listar as interface ativas:
debian:~# ifconfig

Para listar todas as interfaces:
server:~# ifconfig -a

Para configurar uma interface:
server:~# ifconfig eth0 10.20.1.100 netmask 255.255.255.0
Ou:
server:~# ifconfig eth0 10.20.1.100/24

Para desativar uma interface:
server:~# ifconfig eth0 down

Para mascarar o endereço Mac (a interface deve estar desativada):
server:~# ifconfig eth0 hw ether 00:1A:2B:F8:32:AF

Para ativar uma interface:
server:~# ifconfig eth0 up

Para criar uma interface virtual (o nome da interface : e um nome ou número):
server:~# ifconfig eth0:0 10.25.1.100 netmask 255.255.255.0

Ou:
server:~# ifconfig eth0:0 10.25.1.100/24

ROUTE

O comando route é utilizado para listar e criar rotas, ou seja, ele manipula a tabela de roteamento. Lembrando que é uma tabela dinâmica, portanto, existe apenas em memória, para manter em funcionamento é preciso gravar os comandos em algum arquivo de configuração. Considerado obsoleto e substituído pelo comando IP.

Para listar a tabela de roteamento:
server:~# route

Para listar a tabela de roteamento sem resolver nomes:
server:~# route -n

Adiciona uma rota para o default gateway:
server:~# route add default gw 10.20.1.254

Apaga rota para o default gateway:
server:~# route del default gw 10.20.1.254

Adiciona uma rota para uma rede diferente:

server:~# route add -net 10.20.2.0 netmask 255.255.255.0 gw 10.20.1.100
Ou:
server:~# route add -net 10.20.2.0/24 gw 10.20.1.100

Adiciona uma rota para um host:
server:~# route add -host 10.20.3.100 netmask 255.255.255.255 gw 10.20.1.100

IPROUTE2

O pacote iproute2 provê funcionalidades de configuração de rede e roteamento em um só comando: o comando ip

O comando ip tem a seguinte sintaxe:
ip <opções> <objeto> <comando>

Verificando as configurações das interfaces de rede:
server:~# ip addr show

Desativando uma interface:
server:~# ip link set eth0 down

Ativando uma interface:
server:~# ip link set eth0 up

Limpando a configuração de uma interface:
server:~# ip addr flush eth0

Atribuindo ip a uma interface:
server:~# ip addr add 10.13.1.1/24 dev eth0

Removendo um ip de uma interface:
server:~# ip addr del 10.13.1.1/24 dev eth0
Mostrar a tabela de roteamento:

server:~# ip route show

Mostrar a tabela de roteamento sem resolver nomes:
server:~# ip route -r show

Adicionar uma rota default:
server:~# ip route add default via 10.13.1.254 dev eth0

Adicionar rota para uma rede:
server:~# ip route add 172.31.1.0/24 via 10.13.1.200 dev eth0

Arquivos de Configuração de Rede

Debian e derivados

Em sistemas Debian e derivados todas as interfaces de rede e seus parâmetros são configurados no arquivo **/etc/network/interfaces**.

```
# Exemplo de Arquivo interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 10.20.1.100
        netmask 255.255.255.0
        network 10.20.1.0
        broadcast 10.20.1.255
        gateway 10.20.1.254
```

As principais variáveis utilizadas no arquivo são:

auto	- ativa uma interface ao iniciar o sistema
allow-hotplug	- ativa uma interface quando detectada conexão
iface	- nome da interface

inet - tipo de configuração de ip, pode ser:
dhcp - ip dinâmico
static- ip fixo
inet6 - tipo de configuração de ip, pode ser:
dhcp - ip dinâmico
static- ip fixo
auto - autoconfiguração IPv6
address - endereço ip
netmask - máscara de rede
network - rede a que pertence
broadcast - broadcast da rede
gateway - endereço do default gateway da rede
hwaddress ether - endereço do Mac address
dns-nameservers- endereço dos servidores DNS (caso use o pacote resolvconf e queira forçar um DNS em específico)
pre-up - executa comandos ANTES de ativar a interface
up - executa comandos APÓS ativar a interface
post-down - executa comandos APÓS desativar a interface
bridge_ports - define as interfaces usadas na bridge, geralmente vem acompanhado de um comando *up* e um comando *brctl* ...
bridge_stp on - permite que várias bridges se comuniquem entre si para descoberta de rede e prevenção de loop
bridge_maxwait - tempo em segundos que o sistema aguarda a inicialização da bridge
vlan-raw-device - qual interface física é usada pra essa vlan em configuração
wpa-ssid - identifica qual a ESSID da rede a se conectar
wpa-psk - especifica o PASSWORD ou KEY da rede

Red Hat e derivados

Em sistemas Red Hat e derivados a configurações das interfaces de rede ficam em /etc/sysconfig/network-scripts/ifcfg-**INTERFACE** onde cada arquivo configura uma interface específica de rede.

 # Exemplo de Arquivo de cada interface

ONBOOT=yes
USERCTL=no
IPV6INIT=no
PEERDNS=yes
TYPE=Ethernet
DEVICE=eth0
HWADDR=00:0f:ea:db:92:15
BOOTPROTO=static
NETMASK=255.255.255.0
IPADDR=192.168.1.200
GATEWAY=192.168.1.254

As principais variáveis utilizadas no arquivo são:

ONBOOT	- ativar a interface a iniciar o sistema
USERCTL	- permite ou não o controle por usuários
IPV6INIT	- carrega o suporte a IPv6 para esta interface
IPV6ADDR	- endereço IPv6 e máscara
PEERDNS	- usa o dns fornecido por terceiros
TYPE	- tipo de encapsulamento de link
DEVICE	- interface
HWADDR	- endereço Mac
BOOTPROTO	- tipo de configuração de IP
NETMASK	- mascara de rede
IPADDR	- endereço de IPv4
GATEWAY	- endereço do default gateway
HOSTNAME	- nome da máquina
VLAN	- habilita o uso de vlan naquela placa
ESSID	- identifica qual a ESSID da rede a se conectar
KEY	- especifica a KEY hexadecimal da rede
WPA	- habilita uso de WPA nas redes

WPA SUPPLICANT

Em sistemas baseados em Red Hat o pacote ***wpa_supplicant*** permite utilizar o padrão WPA e WPA2 para conexão com a rede. Para funcionar nesses sistemas devemos seguir os seguintes passos.

Primeiro criamos o script para inicialização do WPA:
Edite o /etc/sysconfig/network-scripts/ifup-wireless e coloque no final do arquivo as seguinte linhas:

```
if [ "$WPA" = "yes" -a -x /etc/init.d/wpa_supplicant ]; then
   /sbin/service wpa_supplicant start
fi
```

No arquivo /etc/wpa_supplicant/wpa_supplicant.conf configure sua rede e chave pre compatilhada:

```
network={
        ssid="MEUSSIDDEREDE"
        scan_ssid=1
        key_mgmt=WPA-PSK
        psk="MINHASENHAPSK"
}
```

Lembrando em em no arquivo ifcfg-INTERFACE deve estar:
TYPE=Wireless
WPA=yes

Em sistemas baseados em Debian para usar o **wpasupplicant** adicionamos apenas as seguintes linhas ao arquivo interfaces na seção da interface wireless:

```
auto wlan0
iface wlan0 inet dhcp
        wpa-driver wext
        wpa-conf /etc/wpa_supplicant.conf
```

Sem necessidade de criar um script para inicializar o ***wpasupplicant***.

Netplan

Netplan é um utilitário usado para configurar a rede em sistemas Linux. Ele utiliza arquivos **YAML** para configurar as redes. A grande vantagem do Netplan é que ele permite configurar todo tipo de interface no mesmo arquivo, apontar os servidores de DNS, criar rotas para as redes, criar interfaces bond, criar vlan, etc.

Ele é agnóstico em relação à distribuição e simples de gerenciar tão logo se acostume com seus arquivos e variáveis de configuração.

Ele lê arquivos .yaml em 3 locais, onde encontrar ele utiliza par configurar a rede:

/run/netplan/*.yaml
/etc/netplan/*.yaml
/lib/netplan/*.yaml

Ele pode usar alguns backends para configuração de redes: *systemd-networkd* ou *NetworkManager*. Geralmente optamos pelo *systemd-networkd* pois já está integrado a maioria dos sistemas Linux atuais.

Ao usar o *systemd* quando é carregado, geralmente na inicialização, o netplan lê os diretórios mencionados à procura de arquivos .yaml e cria arquivos .network para cada interface de rede em **/run/systemd/network/** e entrega ao daemon de rede do *systemd* o controle do dispositivo.

Exemplo de arquivo de configuração usado pelo netplan, como o arquivo é YAML temos que respeitar a indentação para não ter erros de leitura:

```
network:
  version: 2
  renderer: networkd
  ethernets:
```

```
    enp3s0:
      addresses:
        - 10.10.10.2/24
      gateway4: 10.10.10.1
      nameservers:
        search: [dominio01, dominio02.local]
        addresses: [10.10.10.1, 1.1.1.1]
```

Geralmente colocamos os arquivos no /etc/netplan com nomes como: 10-wired.yaml, 10-netcfg.yaml, 20-wireless.yaml, etc. Assim podemos até ter mais de um arquivo de configuração que serão lidos na ordem numérica que apresentarem, mas normalmente usamos apenas um arquivo para as configurações.

Outro exemplo de configuração para redes cabeadas e sem fio:

```
network:
    version: 2
    renderer: network-manager
    ethernets:
        enp4s0f2:
            optional: yes
            addresses: []
            dhcp4: no
            dhcp6: no
    wifis:
        wlp3s0:
            optional: yes
            dhcp4: no
            dhcp6: no
            addresses: [192.168.1.10/24, "2001:cafe:face::1/64"]
            gateway4: 192.168.1.254
            nameservers:
                addresses: [192.168.1.254, 9.9.9.9, 8.8.8.8]
            access-points:
                "linuxrules":
                    password: "linustorvalds"
```

Os arquivos são estruturados da seguinte forma:

network: - Onde abre as configurações das redes, todos os itens pertencentes a essa opção devem estar indentados, geralmente usamos indentação com 4 caracteres de espaço
version: 2 - Onde descreve a versão do netplan usada
renderer: networkd - O daemon de rede utilizado para gerir as interfaces (networkd (systemd) ou NetworkManager). Pode ser aplicada globalmente ou para um dispositivo específico
ethernets: - Seção onde configuramos a interfaces ethernets, outras opções de tipos de interfaces:
ethernets: para redes cabeadas (incluindo adaptadores USB-REDE)
modems: para dispositivos de modems
wifis: para redes sem fio
bridges: para dispositivos de bridge
vlan: para dispositivos de vlan
match: - Define um critério para casar com uma ou mais interfaces, usado para opções como set-name (mudar nome da placa). Os critérios usados pelo match podem ser:
name: - nome da interface como enp3* (qualquer no 3 barramento PCI)
macaddress: - pelo enderço MAC
driver: - pelo driver de rede
optional: yes - Se a interface é obrigatória pára continuar o boot da máquina, se o valor for yes a interface não precisa ser configurada nem mesmo existir para o boot continuar
dhcp4: yes - Ativa ou não o dhcp cliente em IPv4 para a interface
dhcp6: no - Ativa ou não o dhcp cliente em IPv6 para a interface
dhcp-identifier: mac - Identificador enviado pelo DHCP (usado para integração com DHCP do Windows)
mtu: 1500 - Especifica o MTU daquela interfaces
eth0:/enp0s3:/wlp3s0: - Nome da interface a ser cconfigurada
addresses: [x.x.x.x/y] - Endereço de IP da interface, na seção
gateway4: 192.168.0.1 - Gateway IPv4
gateway6: "2001:dead:beef::1" - Gateway IPv6
macaddress: 00:50:56:3f:56:d5 - Endereço MAC da interface
nameservers: - Seção que configura os servidores de DNS

consultados
search: [dom1.local, dom2.lan] - Lista de domínios de pesquisa que podem ser usados como sufixo na hora de localizar um host
routes: - Configuração de rotas para a interfce, as opções são:
- to: 192.168.3.0/24 - Rede para onde a rota se aplica
via: 192.168.3.1 - Gateway usado para chegar na rede
table: 101 - Tabela de roteamento usada por essa rota
metric: 100 - Metrica da rede

Exemplo de múltiplos gateway em uma interface com vários IPs:

```
ethernets:
    enp3s0:
     addresses:
        - 192.168.1.100/24
        - 172.16.25.100/24
        - 10.10.1.200/24

     routes:
        - to: 0.0.0.0/0
          via: 192.168.1.1
          metric: 100
        - to: 0.0.0.0/0
          via: 172.16.25.1
          metric: 100
        - to: 0.0.0.0/0
          via: 10.10.1.1
          metric: 100
```

Opções para redes wifi:
access-points: - Seção onde são configurados os access points
Para redes WPA basta colocar o nome da rede (ssid:) e indentado abaixo a senha (password: senhadaredewireless) como no exemplo abaixo:

```
        access-points:
          "linuxrules":
              password: "linustorvalds"
```

Para redes WAP Enterprise que precisa de parametros adicionais como tipo de chave, método, identidade, etc. Veja o exemplo abaixo uma rede com WPA-EAP e TTLS:

```
    wifis:
      wlan0:
        access-points:
          workplace:
            auth:
              key-management: eap
              method: ttls
              anonymous-identity: "@empresa.local"
              identity: "nome@empresa.local"
              password: "s3nh@d1f1c1l01"
        dhcp4: yes
```

Para redes WAP Enterprise que precisa de parametros adicionais como tipo de chave, método, identidade, etc. Veja o exemplo abaixo uma rede com WPA-EAP e TLS:

```
    wifis:
      wl0:
        access-points:
          university:
            auth:
              key-management: eap
              method: tls
              anonymous-identity: "@empresa.local"
              identity: "nome-user@empresa.local"
              ca-certificate: /etc/ssl/CERT-DA-CA-DO-
SERVER.pem
              client-certificate: /etc/ssl/CERT-DO-
SERVER.pem
              client-key: /etc/ssl/CHAVE-DO-SERVER-
KEY.pem
              client-key-password: "s3nh@d1f1c1l02"
        dhcp4: yes
```

Outras configurações de rede:

Para criar bridges basta declarar a interface e depois configurar a bridge.

briges: - Seção que configura as bridges
br0: - Define o nome da bridge que será configurada
interfaces: [enp3s0, enp4s0] - Define as interfaces que farão parte dessa bridge, podemos usar interfaces físicas ou vlans

Exemplo de configuração de interfaces bridges:

```
network:
  version: 2
  renderer: networkd
  ethernets:u
    enp3s0:
      dhcp4: no
  bridges:
    br0:
      dhcp4: yes
      interfaces:
        - enp3s0
```

Para fazer vínculo entre placas (Bonding) primeiro devemos declarar as interfaces a serem utilizadas e depois criar os vínculos (bonds).

bonds: - Seção que configura as bonds
bond01-lan: - Define o nome da bond que será configurada
interfaces: [enp1s0, enp4s0] - Define as interfaces que farão parte dessa bond
parameter: - Seção que define os paramêtros dessa bond
mode: balance-rr - Define o modo de funcionamento da bond
gratuitous-arp: 5 - Define quantos pacotes ARP serão enviados após o failover
mii-monitor-interval: 1 - Intervalo em segundos que o **MII** irá monitorar as interfaces

Modos de funcionamento de uma bond:

balance-rr - Define uma política round-robin para tolerância a falhas e balanceamento de carga. As transmissões são recebidas e enviadas sequencialmente em cada interface escrava vinculada, começando com a primeira disponível

active-backup - Define uma política de backup ativo para tolerância a falhas. As transmissões são recebidas e enviadas por meio da primeira interface escrava vinculada disponível. Outra interface escrava vinculada é usada apenas se a interface escrava vinculada ativa falhar.

802.3ad - Ou LACP (Link Aggregation Control Protocol), define uma política de agregação de link dinâmico IEEE 802.3ad. Cria grupos de agregação que compartilham a mesma velocidade e configurações duplex. Transmite e recebe em todos os escravos no agregador ativo. Requer um switch compatível com 802.3ad

balance-tlb - Define uma política de balanceamento de carga de transmissão (TLB) para tolerância a falhas e balanceamento de carga. O tráfego de saída é distribuído de acordo com a carga atual em cada interface escrava. O tráfego de entrada é recebido pelo escravo atual. Se o escravo receptor falhar, outro escravo assume o controle do endereço MAC do escravo com falha. Este modo é adequado apenas para endereços locais conhecidos pelo módulo de bonding do kernel e, portanto, não pode ser usado por trás de uma bridge com máquinas virtuais.

Exemplo de configuração de interfaces bonding:

```
network:
   version: 2
   renderer: networkd
   ethernets:
      enp1s0:
         dhcp4: no
      enp2s0:
```

```
      dhcp4: no
    enp3s0:
      dhcp4: no
      optional: true
    enp4s0:
      dhcp4: no
      optional: true
    enp5s0:
      dhcp4: no
      optional: true
    enp6s0:
      dhcp4: no
      optional: true
  bonds:
    bond-lan:
      interfaces: [enp2s0, enp3s0]
      addresses: [192.168.93.2/24]
      parameters:
        mode: 802.3ad
        mii-monitor-interval: 1
    bond-wan:
      interfaces: [enp1s0, enp4s0]
      addresses: [192.168.1.252/24]
      gateway4: 192.168.1.1
      nameservers:
        search: [local]
        addresses: [8.8.8.8, 8.8.4.4]
      parameters:
        mode: active-backup
        mii-monitor-interval: 1
        gratuitious-arp: 5
    bond-conntrack:
      interfaces: [enp5s0, enp6s0]
      addresses: [192.168.254.2/24]
      parameters:
        mode: balance-rr
        mii-monitor-interval: 1
```

Para criar vlans basta declarar a interface e depois configurar a vlan.

vlans: - Seção que configura as vlans
vlan10: - Define o nome da vlan configurada
id: 10 - ID da vlan
link: enp3s0 - Define a interface que fará parte dessa vlan, a mesma interface pode fazer parte de várias vlans
accept-ra: no - Faz com que o kernel configure o IPv6 sozinho. Quando habilitado, aceita anúncios de roteador. Quando desabilitado, não responde a anúncios do roteador. Se não estiver definido, usa a configuração padrão do kernel do host.

Exemplo de configuração de vlan:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth1:
      match:
        macaddress: "bb:dd:aa:ef:ca:fe"
      set-name: eth1
      addresses: [ "192.168.3.5/23" ]
      gateway4: 192.168.3.1
      nameservers:
        addresses: [ "8.8.8.8", "8.8.4.4" ]
        search: [ empresa.com.br ]
  vlans:
    vlan15:
      id: 15
      link: eth1
      addresses: [ "172.16.25.5/24" ]
    vlan10:
      id: 10
      link: eth1
      addresses: [ "192.168.5.10/24" ]
      nameservers:
        addresses: [ "127.0.0.1" ]
        search: [ empresa.local, lab.com ]
```

Netplan em sistemas RPM

Instale o netplan via snap:

server:~# dnf install epel-release
server:~# dnf install snapd
server:~# systemctl enable --now snapd.socket
server:~# ln -s /var/lib/snapd/snap /snap
server:~# snap refresh
server:~# snap find netplan
server:~# snap install netplan --edge --classic

Instale o systemd-networkd
server:~# dnf install systemd-networkd

Instale o python3
server:~# dnf install python3

Desabilite ou remova o Network-Manager
server:~# systemctl disable NetworkManager.service

Habilite o systemd-networkd
server:~# systemctl start systemd-networkd.service
server:~# systemctl enable systemd-networkd

Crie o diretório do netplan
server:~# mkdir /etc/netplan

Copie um dos modelos de configuração de acordo com seu tipo de placa, para configuração de placa ethernet com ip estático use static.yaml
server:~# cp /var/lib/snapd/snap/netplan/224/usr/share/doc/netplan/examples/static.yaml /etc/netplan/

Edite de acordo com suas configurações de rede

Crie um link do generate do netplan

server:~# mkdir /lib/netplan

server:~# ln -s
/var/lib/snapd/snap/netplan/224/lib/netplan/generate
/lib/netplan/

Teste para ver se sua configuração do netplan está ok:
server:~# netplan try

Crie uma configuração de bypass do selinux para o netplan
server:~# ausearch -c '(netplan)' --raw | audit2allow -M my-netplan
server:~# semodule -X 300 -i my-netplan.pp

Crie um arquivo de inicialização do netplan para o systemd
server:~# vi /etc/systemd/system/netplan.service

[Unit]
Description=netplan watcher

[Service]
Type=oneshot
ExecStart=/var/lib/snapd/snap/bin/netplan apply

[Install]
WantedBy=multi-user.target

Habilite o netplan no boot
systemctl enable netplan.service

Pronto. Ao inicializar o sistema o netplan estará em
funcionamento.

Roteamento

Dadas 4 redes distintas e isoladas uma da outra, onde todas as redes se comunicam através de seus roteadores e cada roteador uma máquina rodando Linux. De acordo com o desenho abaixo onde cada roteador é uma máquina Linux com duas placas de rede e que as redes estão ligadas em série e não diretamente no roteador de internet teremos o seguinte:

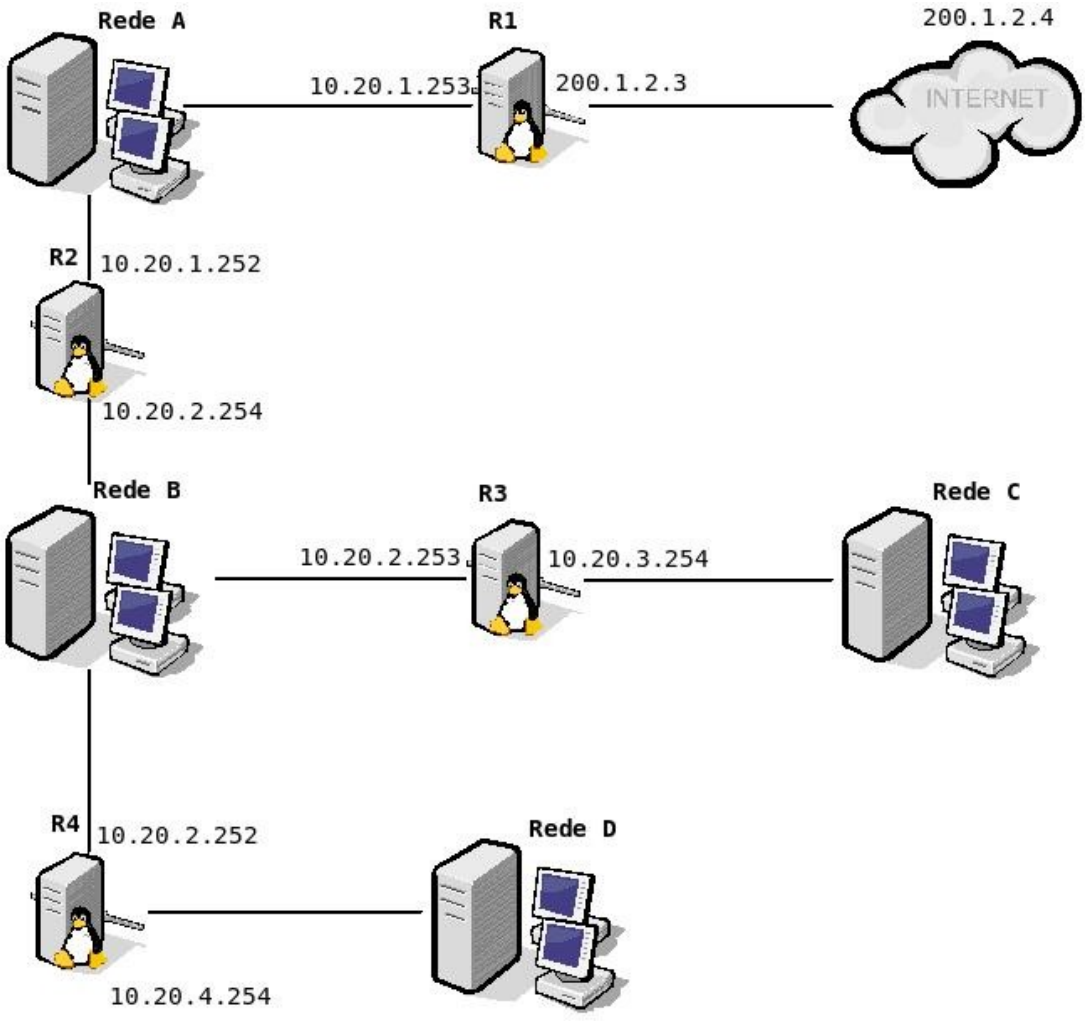
Rede A – 10.20.1.0/24

Rede B – 10.20.2.0/24

Rede C – 10.20.3.0/24

Rede D – 10.20.4.0/24

Veja como ficaria o roteamento entre as redes.



Comandos de rede colocados em cada Roteador:

R1
ip route add default via 200.1.2.4 dev eth0
ip route add 10.20.2.0/24 via 10.20.1.252 dev eth1
ip route add 10.20.3.0/24 via 10.20.1.252 dev eth1
ip route add 10.20.4.0/24 via 10.20.1.252 dev eth1

R2
ip route add default via 10.20.1.253 dev eth0
ip route add 10.20.3.0/24 via 10.20.2.253 dev eth1
route add -net 10.20.4.0 netmask 255.255.255.0 gw 10.20.2.252

R3
ip route add default via 10.20.2.254 dev eth0
ip route add 10.20.4.0/24 via 10.20.2.252 dev eth1

R4
ip route add default via 10.20.2.254 dev eth0
ip route add 10.20.3.0/24 via 10.20.2.253 dev eth1

Roteamento das máquinas clientes de cada rede:
Rede A:
ip route add default via 10.20.1.253
Rede B:
ip route add default via 10.20.2.254
Rede C:
ip route add default via 10.20.3.254
Rede D:
ip route add default via 10.20.4.254

No roteadores a rota default é sempre para a rede desconhecida (geralmente a internet ou a rede que liga com a internet) e as rotas adicionais apontam para o caminho que leva as outras redes.

Um máquina só pode falar com máquinas de rede que ela mesma faça parte, por isso o gateway para uma rede para uma rede é sempre uma máquina que ela conheça e que esteja interligando ela com a rede destino.