

Dltec do Brasil®

www.dltec.com.br

info@dltec.com.br | 413045.7810



DLTEC DO
BRASIL

CURSO CONFIGURAÇÃO DE REDE (TÓPICO 205 DA LPI 201)

Curso Configuração de Rede

Dltec do Brasil®

Todos os direitos reservados©

Copyright © 2019.

É expressamente proibida cópia, reprodução parcial, reprografia, fotocópia ou qualquer forma de extração de informações deste sem prévia autorização da DLteC do Brasil conforme legislação vigente.

O conteúdo dessa apostila é uma adaptação da matéria online do Curso Configuração de Rede.

Aviso Importante!

Esse material é de propriedade da DLteC do Brasil e é protegido pela lei de direitos autorais 9610/98.

Seu uso pessoal e intransferível é somente para os alunos devidamente matriculados no curso. A cópia e distribuição é expressamente proibida e seu descumprimento implica em processo cível de danos morais e material previstos na legislação contra quem copia e para quem distribui.

Para mais informação visite www.dltec.com.br

Introdução

Olá!

Como parte integrante do curso Linux LPI-201, da Dltec do Brasil, esta apostila representa uma adaptação textual do material disponibilizado online pelo curso. Por isso, recomendamos que você utilize-a como um importante recurso offline. Combinando-a com o conteúdo online, você estará muito melhor preparado(a) para realizar o exame 201.

É de suma importância que você, além de participar dos fóruns, realize o máximo possível de exercícios e simulados. Para iniciar, assista aos vídeos introdutórios do capítulo 01 (disponibilizados na Área do Aluno). Lá você obterá mais detalhes sobre o funcionamento geral do curso.

Esperamos que você aproveite ao máximo este material, que foi idealizado com o intuito verdadeiro de fazê-lo(a) obter êxito no exame. Estamos torcendo pelo seu sucesso!

Bons estudos!

Chegamos ao Curso 06!

As configurações que podem ser aplicadas sobre o subsistema de redes do Linux representam tópicos que devem ser estudados com bastante atenção e acuidade.

É essencial que você domine cada aspecto aqui apresentado – já que muito do que será estudado poderá ser aplicado no dia a dia – são assuntos práticos e imprescindíveis para o seu progresso na administração do Linux.

Devido a esta importância, o LPI irá cobrar no exame 11 questões referentes aos tópicos que estudaremos neste capítulo. Sendo assim, continue no mesmo empenho: faça exercícios o máximo que puder e, sobretudo, abra a sua máquina virtual e teste os comandos – leia os manuais, teste as possibilidades, veja os cenários apresentados etc. Esteja sempre a frente nos estudos – vá sempre além!

A Dltec está contigo nesta jornada!

Bons estudos!

Curso 06 –Configuração de Rede

Peso: 11

Objetivos

Ao final desse curso você deverá ter conhecimentos sobre:

- Aplicar configurações básicas e avançadas de rede.
- Identificar e solucionar problemas relacionados a rede.
- Identificar o impacto exercido pelo NetworkManager nas configurações de rede do sistema.

Sumário

1	Introdução	6
1.1	Introdução ao Curso	6
1.2	Sobre a LPI e a LPIC-2	7
1.3	Plano de Estudos para a LPIC-2 e LPI-201	8
1.4	Como Estudar	10
2	Configuração Básica de Rede	11
2.1	Introdução	11
2.2	Utilitários para Manipulação de Interfaces de Rede	11
2.2.1	Comandos ifconfig e ip	12
2.2.2	Manipulação de Interfaces Wireless	17
2.2.3	Manipulação da Tabela de Roteamento IP	22
2.2.4	Manipulação da Tabela ARP	25
3	Configuração Avançada de Rede e Soluções de Problemas	27
3.1	Introdução	27
3.2	Análises e Soluções de Problemas Relacionados a Conectividade e Rotas	28
3.3	Testes Avançados de Conectividade	34

3.4	Escaneamento de Portas e Análise do Tráfego de Pacotes	38
4	<i>Troubleshooting para Problemas de Redes</i>	42
4.1	Introdução	42
4.2	Análise de Estatísticas e Depuração de Rotas	43
4.3	Análise de Estado das Conexões de Rede	48
4.4	Aplicação de Configurações Permanentes nas Interfaces	50
4.5	Determinação das Configurações de DNS	55
4.6	NetworkManager	63
4.7	Investigações em Arquivos de Log e Uso do TCP Wrapper	70
5	<i>Conclusão e Certificado</i>	76
5.1	Conclusão e Certificado	76

1 Introdução

1.1 Introdução ao Curso

Bem-vindo ao curso sobre a **Configuração de Rede**, o qual também faz parte do conteúdo preparatório para a prova de certificação **LPIC-2**, mais especificamente do exame LPI-201, cujo código é **205-450**.

Ao final desse curso você deverá ter conhecimentos sobre:

- Aplicar configurações básicas e avançadas de rede.
- Identificar e solucionar problemas relacionados a rede.
- Identificar o impacto exercido pelo NetworkManager nas configurações de rede do sistema.

Mesmo que não esteja trilhando os estudos para a certificação, você poderá fazer este curso para aumentar os seus conhecimentos no mundo Linux.

Mas se você está na trilha da certificação, saiba que esse curso aborda o **Tópico 205** da prova e tem **peso 11** nessa certificação.

A seguir, vamos falar mais sobre as provas da LPI, da certificação **LPIC-2** e como tudo isso está planejado aqui em nosso Portal.

Não esqueça de que, ao final do curso, você poderá emitir o seu certificado!

1.2 Sobre a LPI e a LPIC-2

"A maior e mais reconhecida certificação Linux do mundo".

A **Linux Professional Institute (LPI)** é a organização mais respeitada no credenciamento de profissionais Linux do mundo, sendo adotada por várias empresas como o padrão global de certificação e organização de suporte de carreira para **profissionais do mundo Linux**.

Atualmente eles possuem **mais de 175,000 profissionais certificados** em **mais de 180 países**.

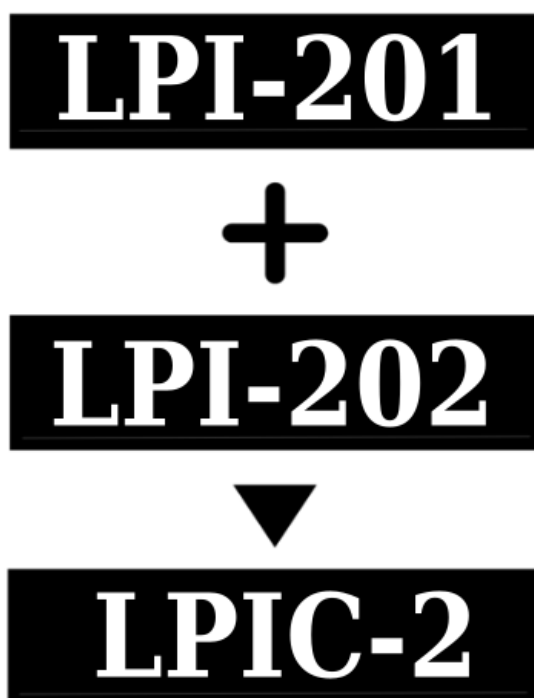
Suas certificações são denominadas de "LPIC" ou "Linux Professional Institute Certification" e, mais especificamente, a certificação **LPIC-2** é a segunda a ser realizada no programa de certificação profissional Linux multi-nível da **LPI** ou **Linux Professional Institute**.

Para seu conhecimento, a **LPI** prevê atualmente **TRÊS** níveis de certificação, ou seja, **LPIC-1**, **LPIC-2** e **LPIC-3**.

Após passar por um rigoroso processo de desenvolvimento, a certificação **LPIC-2** traz em seus objetivos diversas competências a serem testadas nos candidatos que desejam trabalhar com o Linux em um nível muito mais estratégico – mais avançado do que a anterior.

Por esta razão, a **LPIC-2** validará a sua capacidade como candidato em administrar redes de comunicação de pequeno a médio porte, incluindo cenários onde o Linux seja utilizado em conjunto com outros sistemas – compondo, assim, um ambiente misto.

A versão atual da certificação **LPIC-2** é a **4.5**, composta por duas provas com códigos de exame **"201-450"** (ou **"LPI-201"**) e **"202-450"** (ou **"LPI-202"**).



O único requisito para obter a certificação **LPIC-2** é que você possua a certificação **LPIC-1** ainda ativa. Porém, os exames necessários para obtê-las poderão ser realizados em qualquer ordem. Em outras palavras: mesmo que você ainda não seja certificado **LPIC-1**, você poderá fazer ambos os exames da **LPIC-2**. No entanto, você somente receberá esta certificação quando for aprovado(a) nos exames requeridos pela **LPIC-1**.

A certificação **LPIC-2** tem **validade de 5 anos** e um dos seus diferenciais é o fato dela poder **ser realizada em Português!**

Atualmente os exames da **LPIC-2** podem ser realizada nos idiomas Inglês, Alemão, Japonês e **Português** (Brasileiro).

Os exames da LPI podem ser realizados em centros autorizados da **Pearson Vue**, sendo necessário cadastro e matrícula para a realização do exame diretamente pelo site da **Pearson Vue**. O processo de cadastramento e aquisição dos exames serão demonstrados dentro do preparatório durante o curso.

Para tornar-se certificado **LPIC-2** você deverá ser capaz de:

- Realizar tarefas administrativas avançadas, incluindo aquelas relacionadas ao kernel Linux, inicialização e manutenção geral do sistema.
- Efetuar o gerenciamento avançado de dispositivos de bloco e de sistemas de arquivos, além de solucionar problemas de rede envolvendo a autenticação de usuários e questões sobre segurança (incluindo firewall e VPN).
- Instalar e configurar serviços de rede fundamentais, como DHCP, DNS, SSH, servidores Web, servidores de arquivos usando FTP, NFS e Samba – além de serviços de email.
- Supervisionar assistentes e aconselhar os níveis gerenciais sobre medidas a serem tomadas relativas a automação e compras.

A seguir vamos falar sobre como a preparação para a **LPIC-2** está dividida no **Portal da DlteC** e como você deverá utilizar nossa material para conquistar sua certificação.

1.3 Plano de Estudos para a LPIC-2 e LPI-201

Como você já sabe, a certificação **LPIC-2** é composta por DOIS exames: **LPI-201 (prova 201-450)** e **LPI-202 (prova 202-450)**.

Este curso em que você está matriculado faz parte da trilha de preparação da **LPI-201**.

Existe um curso no Portal chamado "**LPIC-2: LPI 201-450**", o qual **agrega todos os cursos da trilha da LPI-201** e que contém os simulados e fóruns de tira dúvidas com o professor responsável pelo curso.

O **plano de estudos** para você ter sucesso na **LPI-201** é o seguinte:

1. Estudar o conteúdo de cada Curso Express (sequência de cursos a seguir).
2. Repetir os comandos e demonstrações práticas realizadas pelo Prof. Leandro durante as vídeo aulas como laboratório.
3. Fazer os simulados que estão dentro do curso "**LPIC-2: LPI 201-450**".
4. A qualquer momento tirar as dúvidas do conteúdo utilizando os fóruns correspondentes.

A **LPI-201** é composta dos cursos conforme sequência da figura abaixo:



Prova LPI 201-450 LPIC-2 Versão 4.5

O exame **LPI-201** é composto por 60 questões, a serem resolvidas em 90 minutos.

Cada um dos tópicos que compõem este exame possui os seus **Objetivos**; cada objetivo possui um **Peso**.

Quanto maior o peso, maior será a quantidade de questões no exame.

Para ser aprovado(a), você deverá conseguir obter (no mínimo) 500 pontos na **LPI-201**.

Se você fez a matrícula nesse curso com o objetivo de tirar a certificação então a partir de agora, foco total no objetivo: **OBTER A CERTIFICAÇÃO**.

Você será aprovado(a) – já coloque isso “na cabeça”.

Para isso, pratique os comandos, leia os tópicos com cautela e marque logo o dia do seu exame (para você já ter uma data limite).

Faça o seu cronograma, estipule as horas de estudo e, sinceramente, não tem erro.

Repita essa frase todos os dias: **Eu serei aprovado(a).**

Se você assumir esse compromisso com sinceridade e vontade de vencer, **tudo dará certo.**

Estamos ao seu lado! Bons estudos!

(*) Os fóruns do curso são exclusivos para TIRAR AS DÚVIDAS DO CURSO, caso você tenha dúvidas do dia a dia ou que não tenham correlação com o curso utilize os grupos do Facebook ou Telegram para troca de ideias.

1.4 Como Estudar

Nesse curso você terá **vídeo-aulas** e **material de leitura** para o aprendizado do conteúdo.

Posso somente ler ou assistir aos vídeos? NÃO RECOMENDAMOS!

O ideal é você assistir aos vídeos e na sequência ler os conteúdos ou...

... se preferir leia os conteúdos e depois assistia aos vídeos, tanto faz.

POR QUE LER E ASSISTIR?

Simples, porque **um conteúdo complementa o outro**. Principalmente se você está se preparando para a prova de certificação é crucial que você tanto veja os vídeos como a matéria de leitura!

2 Configuração Básica de Rede

2.1 Introdução

Subtópico: 205.1 Peso: 3

Descrição do objetivo: Você deverá conseguir configurar um dispositivo de rede para que este consiga se conectar em uma rede local (cabeadas ou sem fio) e em uma WAN (wide-area network). Este objetivo inclui a comunicação entre várias subredes contidas em uma rede, incluindo redes IPv4 e IPv6.

Áreas-chave:

- Utilitários para configurar e manipular interfaces de rede ethernet.
- Configuração básica do acesso a redes sem fio.

Principais termos, arquivos e utilidades:

- ip
- ifconfig
- route
- arp
- iw
- iwconfig
- iwlist

2.2 Utilitários para Manipulação de Interfaces de Rede

Apesar de ser possível realizar a manipulação de interfaces de rede a partir de ferramentas gráficas, em muitos ambientes de servidor estas não encontram-se disponíveis. Sendo assim, são disponibilizados diferentes comandos que poderão ser utilizados nesses cenários.

Nesta seção do capítulo, veremos alguns que poderão ser cobrados no dia do seu exame. Inclusive, alguns deles você já estudou durante os preparativos para a certificação **LPIC-1**. Dessa forma, aproveite o momento para revisá-los.

2.2.1 Comandos ifconfig e ip



O comando **ifconfig**, parte integrante do pacote "net-tools", é responsável por aplicar configurações nas interfaces de rede durante o momento do boot.

Com o sistema já iniciado, normalmente o comando somente é usado em cenários de depurações/testes.

Atualmente considerado obsoleto e substituído pelo comando **ip**, muitas vezes o **ifconfig** já não costuma vir instalado por padrão nas distribuições mais modernas.

Quando utilizado sem qualquer opção, o comando **ifconfig** exibe o estado de todas as interfaces de rede atualmente ativas no sistema:

```
[root@curso6:~]# ifconfig

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.90 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe27:569e prefixlen 64 scopeid 0x20<link>
ether 08:00:27:27:56:9e txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 46 bytes 2936 (2.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet172.16.30.52 netmask 255.255.255.0 broadcast 172.16.30.255
inet6 fe80::a00:27ff:fe27:569e prefixlen 64 scopeid 0x20<link>
ether 08:00:27:cd:b6:7e txqueuelen 1000 (Ethernet)
RX packets 38 bytes 2280 (2.2 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 8 bytes 656 (656.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>mtu 65536
inet127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Loopback Local)
RX packets 24 bytes 2520 (2.4 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
```

```
TX packets 24  bytes 2520 (2.4 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note que além da interface de loopback ("lo" – tipo especial de interface), outras duas foram identificadas como ativas ("UP") e em execução ("RUNNING"): "enp0s3" e "enp0s8".

Veja também que o comando exibe outras informações importantes, como o endereço **MAC** (Media Access Control) associado a cada uma das interfaces (campo "ether"), o **MTU** (Maximum Transmission Unit) de cada uma, estatísticas relacionadas à transmissão (**TX**) e recebimento (**RX**) de pacotes, os endereços **IP** configurados etc.

Quando o **ifconfig** é utilizado junto à opção **-a**, o comando também exibe aquelas interfaces que não encontram-se ativas. Para compor o exemplo, vamos "derrubar" a interface "enp0s8", passando a instrução "down" junto ao comando:

```
[root@curso6:~]#ifconfig enp0s8 down
```

Ao rodarmos novamente o comando sem qualquer opção, as informações sobre a interface em questão não são exibidas:

```
[root@curso6:~]# ifconfig
```

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.90  netmask 255.255.255.0  broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe27:569e  prefixlen 64  scopeid 0x20<link>
ether 08:00:27:27:56:9e  txqueuelen 1000  (Ethernet)
RX packets 52193  bytes 76614278 (73.0 MiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 14029  bytes 989485 (966.2 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>mtu 65536
inet127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop txqueuelen 1000  (Loopback Local)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Para "subir" novamente a interface "enp0s8", basta utilizar a instrução "up" junto ao comando:

```
[root@curso6:~]#ifconfig enp0s8 up
```

O comando **ifconfig** poderá ser utilizado em diferentes cenários de testes. Por exemplo, vemos que a interface "enp0s8" possui o endereço **IPv4** configurado como 172.16.30.52/24. Para alterar de forma temporária o seu endereço para 192.168.0.78/24, por exemplo, basta especificá-lo após a string identificadora da interface:

```
[root@curso6:~]# ifconfig enp0s8 192.168.0.78 netmask 255.255.255.0
```

```
[root@curso6:~]# ifconfig enp0s8
```

```
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.78  netmask 255.255.255.0  broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe27:569e  prefixlen 64  scopeid 0x20<link>
ether 08:00:27:cd:b6:7e  txqueuelen 1000  (Ethernet)
```

```
RX packets 159 bytes 9540 (9.3 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 11 bytes 836 (836.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

A instrução "netmask" é opcional. Caso não seja informada, a máscara de rede padrão da classe do endereço **IPv4** informado será configurada.

É também importante ressaltar que uma interface poderá possuir mais de um endereço **IP**. Para acrescentar mais um à interface "enp0s8", utilizamos o comando **ifconfig** junto com a instrução "add":

```
[root@curso6:~]# ifconfig enp0s8 add 192.168.0.27

[root@curso6:~]# ifconfig

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.90 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe27:569e prefixlen 64 scopeid 0x20<link>
ether 08:00:27:27:56:9e txqueuelen 1000 (Ethernet)
RX packets 12 bytes 720 (720.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 200 bytes 12176 (11.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.78 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe27:569e prefixlen 64 scopeid 0x20<link>
ether 08:00:27:27:56:9e txqueuelen 1000 (Ethernet)
RX packets 192 bytes 11520 (11.2 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 20 bytes 1376 (1.3 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8:0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.27 netmask 255.255.255.0 broadcast 192.168.0.255
ether 08:00:27:27:56:9e txqueuelen 1000 (Ethernet)

lo: flags=73<UP,LOOPBACK,RUNNING>mtu 65536
inet127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Loopback Local)
RX packets 142 bytes 15008 (14.6 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 142 bytes 15008 (14.6 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

O comando **ifconfig** também permite alterar (de forma temporária) o endereço **MAC** de uma determinada interface. Acompanhe:

```
[root@curso6:~]# ifconfig enp0s8 | grep ether

ether 08:00:27:27:56:9e txqueuelen 1000 (Ethernet)

[root@curso6:~]# ifconfig enp0s8 hw ether 02:02:02:02:02:02

[root@curso6:~]# ifconfig enp0s8 | grep ether

ether 02:02:02:02:02:02 txqueuelen 1000 (Ethernet)
```

O versátil comando **ip** não somente substitui o comando anterior, mas também o comando **route** (que estudaremos mais a frente).

Para visualizar dados gerais do funcionamento das interfaces ativas ou inativas (incluindo os seus endereços **MAC**), utilizamos o comando junto ao objeto "link" (podendo ser abreviado para "l") – a ação "show" (ou "sh") é opcional:

```
[root@curso6:~]# ip link show
```

```
1: lo: <LOOPBACK,UP,LOWER_UP>mtu 65536 qdiscnoqueue state UNKNOWN mode DEFAULT
group default qlen 1000
link/loopback 00:00:00:00:00:00brd 00:00:00:00:00:00
```

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
mode DEFAULT group default qlen 1000
link/ether 08:00:27:27:56:9e brdff:ff:ff:ff:ff:ff
```

```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
mode DEFAULT group default qlen 1000
link/ether 08:00:27:cd:b6:7e brdff:ff:ff:ff:ff:ff
```

Se a intenção for visualizar as configurações de endereçamento **IP** aplicadas às interfaces, o objeto "address" (podendo ser abreviado para "a" ou "addr") deverá ser utilizado – neste caso, o objeto "show" (ou "list") é opcional:

```
[root@curso6:~]# ip address show
```

```
# Ou
```

```
[root@curso6:~]# ip address
```

```
1: lo: <LOOPBACK,UP,LOWER_UP>mtu 65536 qdiscnoqueue state UNKNOWN group default
qlen 1000
link/loopback 00:00:00:00:00:00brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
```

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:27:56:9e brdff:ff:ff:ff:ff:ff
inet 192.168.0.90/24 brd 192.168.0.255 scope global noprefixroute enp0s3
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe27:569e/64 scope link
valid_lft forever preferred_lft forever
```

```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 02:02:02:02:02:02brdff:ff:ff:ff:ff:ff
inet 192.168.0.78/24 brd 192.168.0.255 scope global noprefixroute enp0s8
valid_lft forever preferred_lft forever
inet 192.168.0.27/24 brd 192.168.0.255 scope global secondary enp0s8:0
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fedc:b67e/64 scope link
valid_lft forever preferred_lft forever
```

Informação adicional:

Caso a opção **-4** seja utilizada antes dos objetos "address" e "show", somente são exibidos endereços **IPv4** atribuídos às interfaces (ao contrário da opção **-6**, que exibe apenas os endereços **IPv6**).

O comando **ip** também poderá ser usado para atribuir um endereço **IP** temporário à interface. Por exemplo, se for necessário adicionar um endereço **IPv4** secundário à interface "enp0s3", podemos utilizar o objeto "address" junto com a ação "add":

```
[root@curso6:~]# ip address add 192.168.0.129/24 dev enp0s3
```

```
[root@curso6:~]# ip address show enp0s3
```

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:27:56:9e brdff:ff:ff:ff:ff:ff
inet 192.168.0.90/24 brd 192.168.0.255 scope global noprefixroute enp0s3
valid_lft forever preferred_lft forever
inet 192.168.0.129/24 scope global secondary enp0s3
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe27:569e/64 scope link
valid_lft forever preferred_lft forever
```

Note que utilizamos a notação **CIDR** (ClasslessInter-DomainRouting) para especificar a máscara de rede 255.255.255.0. Esta também poderá ser declarada de forma explícita no comando:

```
[root@curso6:~]# ip address add 192.168.0.129/255.255.255.0 dev enp0s3
```

Vamos agora utilizar o comando **ip** para remover este último endereço adicionado à interface "enp0s3". Para isto, basta utilizar o objeto "address" junto a ação "del":

```
[root@curso6:~]# ip address del 192.168.0.129/24 dev enp0s3
```

```
[root@curso6:~]# ip address show enp0s3
```

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:27:56:9e brdff:ff:ff:ff:ff:ff
inet 192.168.0.90/24 brd 192.168.0.255 scope global noprefixroute enp0s3
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe27:569e/64 scope link
valid_lft forever preferred_lft forever
```

Para remover quaisquer configurações de endereçamento **IP** atribuídos a uma interface, o comando **ip** oferece a ação "flush":

```
[root@curso6:~]# ip address flush dev enp0s3
```

```
[root@curso6:~]# ip address show enp0s3
```

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:27:56:9e brdff:ff:ff:ff:ff:ff
```


O estado das interfaces também poderá ser manipulado a partir do comando **ip**. Para "derrubar" e, em seguida, "subir" a interface "enps03", por exemplo, basta executar:

```
# Derrubando a interface enp0s3

[root@curso6:~]# ip link set dev enp0s3 down

# Subindo novamente a interface enp0s3

[root@curso6:~]# ip link set dev enp0s3 up
```

2.2.2 Manipulação de Interfaces Wireless



Até agora manipulamos interfaces de rede "wired", ou seja, que utilizam fios. Geralmente essas interfaces são identificadas pelas strings "enp*" ou "eth*". As interfaces wireless (sem fio), por outro lado, são identificadas como "wl*".

Utilizaremos a seguir um host físico onde uma interface wireless encontra-se disponível. Ao executar o comando **ifconfig** sobre ela, notamos que a interface (identificada como "wlp2s0") já possui suas configurações de endereçamento IP e já se encontra em atividade:

```
[root@curso6:~]# ifconfig wlp2s0
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet 192.168.0.106 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::5008:925a:95a5:b2b7 prefixlen 64 scopeid 0x20<link>
ether a4:17:31:fc:df:0d txqueuelen 1000 (Ethernet)
RX packets 295 bytes 100879 (100.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 157 bytes 19959 (19.9 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Nosso objetivo será conectá-la (via shell) a um ponto de acesso sem fio que utilize o modo de autenticação **WEP** (WiredEquivalentPrivacy).

Para conseguirmos isso, utilizaremos diferentes comandos específicos para manipular esses tipos de interfaces.

O primeiro que veremos é o **iwconfig**, responsável por exibir e aplicar configurações em diferentes parâmetros da interface. Sua lógica de funcionamento é similar ao **ifconfig**, mas específico para ser utilizado em interfaces sem fio:

```
[root@curso6:~]# iwconfig wlp2s0
wlp2s0 IEEE 802.11 ESSID:"Tux"
Mode: Managed Frequency:2.457 GHz Access Point: 70:4F:57:D2:0E:B8
Bit Rate=1 Mb/s Tx-Power=15 dBm
Retry short limit:7 RTSThr:off Fragment thr:off
Encryption key:off
Power Management:off
Link Quality=70/70 Signal level=-21 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:48 Missed beacon:0
```

Note que a interface já encontra-se associada a uma rede identificada pelo **ESSID** "Tux". Além disso, note que o comando também exibe informações importantes, como o modo de operação da interface ("Mode"), sua frequência ("Frequency"), o endereço **MAC** do ponto de acesso em que está conectada ("Access Point"), a qualidade do link ("Link Quality"), o nível do sinal ("Signallevel") dentre outras.

Outro comando que poderá entrar em cena neste momento é o **iw**. Além de também conseguir exibir informações diversas sobre as interfaces sem fio, o comando também pode ser utilizado para aplicar alterações determinadas pelo administrador.

Por exemplo, para exibir todas as interfaces wireless disponíveis no sistema, basta utilizar o comando **iw** junto ao objeto "dev":

```
[root@curso6:~]# iw dev
phy#0
interface wlp2s0
ifindex 3
wdev 0x1
addr a4:17:31:fc:df:0d
ssid Tux
type managed
channel 10 (2457 MHz), width: 40 MHz, center1: 2447 MHz
txpower 15.00 dBm
```

Veja que o comando também exibe a identificação da rede em que a única interface disponível no sistema encontra-se conectada.

Além disso, o comando também exibe o seu modo de operação ("type"), o canal utilizado ("channel"), o endereço **MAC** da interface ("addr") dentre outras informações úteis. Caso seja necessário obter informações sobre o link, utilizamos o objeto "link":

```
[root@curso6:~]## iw dev wlp2s0 link
Connected to 70:4f:57:d2:0e:b8 (on wlp2s0)
SSID: Tux
freq: 2457
RX: 2731676 bytes (20639 packets)
TX: 23398 bytes (260 packets)
signal: -23 dBm
tx bitrate: 54.0 Mbit/s MCS 3 40MHz

bss flags: short-slot-time
dtim period: 1
beacon int: 100
```

Veja que o comando agora exibe, além da rede que está sendo utilizada, o sinal que está sendo utilizado, quantidade de pacotes recebidos ("RX"), transmitidos ("TX"), endereço **MAC** do ponto de acesso dentre outras questões.

Prosseguindo, vamos agora derrubar a interface sem fio. Para isto, utilizamos o comando **ifconfig**:

```
[root@curso6:~]# ifconfig wlp2s0 down
```

Em seguida, após pararmos o serviço **NetworkManager**, vamos lançar novamente os comandos **iwconfig** e **iw** sobre a interface. Acompanhe:

```
[root@curso6:~]# systemctl stop NetworkManager
```

```
[root@curso6:~]# iwconfig wlp2s0
wlp2s0 IEEE 802.11 ESSID:off/any
Mode: Managed Access Point: Not-Associated Tx-Power=15 dBm
Retry short limit:7 RTSthr:off Fragment thr:off
Encryption key:off
Power Management:off
```

```
[root@curso6:~]# iw dev wlp2s0 link
Notconnected
```

Repare que, agora, a interface não encontra-se associada a nenhum ponto de acesso. Para finalizar esta primeira etapa, vamos remover quaisquer configurações de endereçamento **IP** aplicadas sobre a interface:

```
[root@curso6:~]# ip address flush wlp2s0
```

Perceba na saída do comando **iwconfig** anterior que o modo de operação da interface "wlp2s0" está configurado como "Managed" (um dos modos possíveis de serem utilizados pelas interfaces).

Apenas como um exemplo, podemos utilizar tanto o comando **iwconfig** quanto **iw** para realizar a alteração deste parâmetro para o modo "monitor". Acompanhe:

```
[root@curso6:~]# iwconfig wlp2s0 mode monitor
```

```
[root@curso6:~]# iwconfig wlp2s0
wlp2s0 IEEE 802.11 Mode:Monitor Tx-Power=15 dBm
Retry short limit:7 RTSthr:off Fragment thr:off
Power Management:off
```

Para retornarmos ao modo "Managed", podemos lançar o comando **iwconfig** substituindo "monitor" por "managed" ou utilizar o comando **iw**:

```
[root@curso6:~]# iw dev wlp2s0 set type managed
```

```
[root@curso6:~]# iw dev
phy#0
Interface wlp2s0
ifindex 3
wdev 0x1
addr a4:17:31:fc:df:0d
typemanaged
txpower 15.00 dBm
```

Esta é uma das possibilidades de alteração de parâmetros relacionados às interfaces de rede sem fio.

Agora entra em cena um comando muito importante, que é o **iwlist**. Este comando consegue exibir ainda mais informações a respeito dessas interfaces – indo mais além do que aquelas exibidas pelo **iwconfig**.

Por exemplo, se for necessário exibir todos os canais/frequências que poderão ser aplicadas a interface "wlp2s0", lançamos o **iwlist** da seguinte forma:

```
[root@curso6:~]# iwlist wlp2s0 frequency

# Ou

[root@curso6:~]# iwlist wlp2s0 channel
wlp2s0      14 channels in total; available frequencies:
Channel 01: 2.412 GHz
Channel 02: 2.417 GHz
Channel 03: 2.422 GHz
Channel 04: 2.427 GHz
Channel 05: 2.432 GHz
Channel 06: 2.437 GHz
Channel 07: 2.442 GHz
Channel 08: 2.447 GHz
Channel 09: 2.452 GHz
Channel 10: 2.457 GHz
Channel 11: 2.462 GHz
Channel 12: 2.467 GHz
Channel 13: 2.472 GHz
Channel 14: 2.484 GHz
```

O comando **iw** também poderá ser utilizado para visualizar essas frequências. Ao lançarmos este comando junto à opção "list", essas informações serão listadas (além de muitas outras úteis).

Seguindo os exemplos, agora vamos utilizar o comando **iwlist** para exibir todos os pontos de acesso disponíveis nas redondezas – ou seja, todas as redes que a interface "wlp2s0" consegue identificar. Para isto, após levantar a interface, utilizamos a opção "scanning" (ou "scan"):

```
[root@curso6:~]# ifconfig wlp2s0 up

[root@curso6:~]# iwlist wlp2s0 scan
```

Ao executar o comando **iwlist** para pesquisar as redes disponíveis, várias células serão exibidas junto com os seus respectivos **ESSIDs**, qualidade e nível de sinal, frequências, taxas de bits dentre outras informações.

O comando **iw** também poderá ser utilizado para o mesmo propósito:

```
[root@curso6:~]# iw dev wlp2s0 scan
```

Para os nossos exemplos, vamos trabalhar com a rede de **ESSID** igual a "Teste_Dltec". Vamos agora derrubar novamente a interface e, em seguida, utilizar o comando **iwconfig** para associar este **ESSID** à interface "wlp2s0" – ou seja, vamos alterar este parâmetro para que ela esteja associada à rede em questão:

```
[root@curso6:~]# ifconfig wlp2s0 down

[root@curso6:~]# iwconfig wlp2s0 essidTeste_Dltec

root@curso6:~# iwconfig wlp2s0
wlp2s0 IEEE 802.11 ESSID:"Teste_Dltec"
Mode:Managed Access Point: Not-Associated Tx-Power=15 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off
```

Repare que a rede "Teste_Dltec" foi associada com sucesso à interface, porém esta não encontra-se ainda associada a um ponto de acesso. Para isto, lançaremos novamente o comando **iwconfig** junto à senha que está sendo utilizada para proteger esta rede. Por padrão, a opção "key" do comando aceita esta senha utilizando valores em hexadecimal. Para que esta seja interpretada como **ASCII**, devemos precedê-la pelos caracteres "s:". Acompanhe:

```
[root@curso6:~]# iwconfig wlp2s0 key s:12345

root@curso6:~# iwconfig wlp2s0
wlp2s0 IEEE 802.11 ESSID:"Teste_Dltec"
Mode:Managed Access Point: Not-Associated Tx-Power=15 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:3132-3334-35
Power Management:off
```

Note que o campo "Encryptionkey" foi agora devidamente configurado, indicando que o procedimento anterior foi realizado com sucesso. Agora, para fecharmos esta sequência, precisamos configurar um endereço **IP** na interface "wlp2s0". Para que este seja obtido via **DHCP**, vamos executar o comando dhclient:

```
[root@curso6:~]# dhclient wlp2s0
```

Rodando o comando **iwconfig** sobre a interface, vemos que agora o ponto de acesso referente à rede "Teste_Dltec" foi devidamente associado à interface:

```
[root@curso6:~]# iwconfig wlp2s0
wlp2s0 IEEE 802.11 ESSID:"Teste_Dltec"
Mode:Managed Frequency:2.412 Access Point: 8A:38:EB:68:5A:53
Bit Rate=54 Mb/s Tx-Power=15dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:3132-3334-35
Power Management:off
Link Quality=66/70 Signal level=-44 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:2 Missed beacon:0
```

Informação adicional:

Os comandos **iwconfig** e **iw** não suportam os esquemas de autenticação **WPA/WPA2**. Dessa forma, eles somente conseguirão conectar-se a redes que utilizam **WEP** (ou naquelas abertas). Para realizar a conexão em redes que utilizam esses outros métodos de autenticação, o pacote "wpa_supplicant" costuma ser utilizado.

2.2.3 Manipulação da Tabela de Roteamento IP

O comando **route**, parte integrante do pacote "net-tools", é utilizado para exibir e manipular a tabela de roteamento **IP** do kernel. Para simplificar os exemplos iniciais, vamos "derrubar" a interface "enp0s8" e, em seguida, utilizar o comando **route** junto à opção **-n**, de forma que não haja o processo de resolução de nomes:

```
[root@curso6:~]# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3

Quando pacotes precisam ser enviados aos hosts da rede **IP** 192.168.0.0/24, esses são diretamente enviados a partir da interface "enp0s3". O roteador ("Gateway") não precisará se envolver neste cenário. Por isso, seu campo encontra-se ilustrado como "0.0.0.0".

Vamos agora limpar as configurações de endereçamento **IP** da interface "enp0s8":

```
[root@curso6:~]# ip a flush dev enp0s8
```

Em seguida, vamos atribuir o endereço **IP** 10.0.0.56/8 à interface "enp0s8" e subir a interface através da instrução "up" do **ifconfig**:

```
[root@curso6:~]# ifconfig enp0s8 10.0.0.56 up
```

Vamos executar novamente o comando **route** para que seja exibida a interface de roteamento **IP** do kernel:

```
[root@curso6:~]# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	enp0s8
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3

Veja que a rota para pacotes destinados à rede 10.0.0.0/24 foi automaticamente configurada para serem enviados à interface "enp0s8".

Também podemos criar uma rota para uma determinada rede. Por exemplo, vamos especificar a seguir que pacotes destinados à rede 192.168.50.0/24 sejam enviados para o host 192.168.0.1 (atuando, nesse caso, como um gateway para a rede de destino):

```
[root@curso6:~]# route add -net 192.168.50.0/24 gw 192.168.0.1
```

```
[root@curso6:~]# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	enp0s8
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3
192.168.50.0	192.168.0.1	255.255.255.0	UG	0	0	0	enp0s3

Como "enp0s3" encontra-se configurada com um endereço **IPv4** pertencente ao mesmo range que o host atuante como gateway, a saída dos pacotes destinados a rede 192.168.50.0/24 se dará a partir desta interface rumo ao host em questão.

Agora precisamos definir a rota que os pacotes irão tomar caso não sejam destinados a nenhuma das redes explicitamente declaradas.

Por exemplo, se neste momento rodarmos o comando **ping**, de forma a tentarmos enviar 4 pacotes **ICMP** (Internet Control Message Protocol) ao host remoto "www.google.com" usando o protocolo **IP**, a seguinte mensagem será exibida:

```
[root@curso6:~]# ping -c 4 www.google.com
connect: Network is unreachable
```

A rede em questão encontra-se inalcançável porque o sistema não sabe ainda o que fazer com pacotes que não sejam destinados a nenhuma das redes conhecidas. Por isso, precisamos agora definir a rota padrão:

```
[root@curso6:~]# route add default gw 192.168.0.1
```

```
[root@curso6:~]# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	192.168.0.1	0.0.0.0	UG	0	0	0	enp0s3
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	enp0s8
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3
192.168.50.0	192.168.0.1	255.255.255.0	UG	0	0	0	enp0s3

Sendo assim, caso um pacote precise ser enviado a uma rede que não seja 10.0.0.0/8, 192.168.0.0/24 nem 192.168.50.0/24, este será então enviado ao "default gateway" – neste cenário será o host com o endereço **IPv4** 192.168.0.1. E, obviamente, a saída desse pacote se dará a partir da interface "enp0s3".

```
# Testando novamente a conectividade com o host remoto
```

```
[root@curso6:~]# ping -c 4 www.google.com
PING www.google.com (172.217.162.164) 56(84) bytes of data.
64 bytes from rio01s25-in-f4.1e100.net (172.217.162.164): icmp_seq=1 ttl=52
time=14.1 ms
64 bytes from rio01s25-in-f4.1e100.net (172.217.162.164): icmp_seq=2 ttl=52
time=17.1 ms
64 bytes from rio01s25-in-f4.1e100.net (172.217.162.164): icmp_seq=3 ttl=52
time=19.2 ms
64 bytes from rio01s25-in-f4.1e100.net (172.217.162.164): icmp_seq=4 ttl=52
time=20.6 ms
```

```
--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 14.139/17.803/20.649/2.453 ms
```

Já aprendemos a adicionar uma rota para uma determinada rede – no nosso exemplo, para a rede 192.168.50.0/24. Se for necessário remover tal rota, basta informar as opções "del" e "-net":

```
[root@curso6:~]# route del -net 192.168.50.0/24
```

Também é possível criar uma configuração para rejeitar o envio de pacotes destinados a alguma em especial. Por exemplo, se for necessário rejeitar pacotes destinados a rede 192.168.10.0/24, o seguinte comando poderá ser executado:

```
[root@curso6:~]# route add -net 192.168.10.0 netmask 255.255.255.0 reject
```

```
[root@curso6:~]# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
0.0.0.0          192.168.0.1     0.0.0.0          UG    0      0      0 enp0s3
10.0.0.0         0.0.0.0         255.0.0.0        U      0      0      0 enp0s8
192.168.0.0      0.0.0.0         255.255.255.0    U      102    0      0 enp0s3
192.168.10.0     -               255.255.255.0    !      0      -      0 -
```

Para o exame, não esqueça que quando o campo "Flags" apresentar o caractere "!", significa que a rota em questão encontra-se configurada para ser rejeitada.

O comando **ip** também poderá ser utilizado para exibir e manipular a tabela de roteamento **IP** do kernel. Para exibi-la, utilizamos o objeto "route". A ação "show" (ou "list") é opcional:

```
[root@curso6:~]# ip route show
default via 192.168.0.1 dev enp0s3
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.125
192.168.50.0/24 via 192.168.0.1 dev enp0s3
```


Para adicionar uma nova rota à rede 172.16.0.0/16 e configurar para que os pacotes a ela destinados sejam encaminhados ao host 192.168.0.1 através da interface "enp0s3", utilizamos o comando da seguinte forma:

```
[root@curso6:~]#ip r add 172.16.0.0/16 via 192.168.0.1 dev enp0s3

[root@curso6:~]#ip r s
default via 192.168.0.1 dev enp0s3
172.16.0.0/16 via 192.168.0.1 dev enp0s3
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.125
192.168.50.0/24 via 192.168.0.1 dev enp0s3
```

Para que a rota padrão seja removida, basta utilizar o objeto "route" junto à ação "del" – especificando em seguida a string "default":

```
[root@curso6:~]#ip r del default

[root@curso6:~]#iproute 1
172.16.0.0/16 via 192.168.0.1 dev enp0s3
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.125
192.168.50.0/24 via 192.168.0.1 dev enp0s3
```

Para adicioná-la novamente, utilizamos a ação "add" e especificamos o endereço **IP** do gateway:

```
[root@curso6:~]#ip r add default via 192.168.0.1

[root@curso6:~]#ip r s
default via 192.168.0.1 dev enp0s3
172.16.0.0/16 via 192.168.0.1 dev enp0s3
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.125
192.168.50.0/24 via 192.168.0.1 dev enp0s3
```

2.2.4 Manipulação da Tabela ARP

Um sistema em rede torna-se identificável a partir de dois endereços únicos: **IP** (seja versão 4 ou 6) e o endereço de hardware, conhecido como **MAC** (composto por seis grupos de 2 dígitos em hexadecimal – geralmente separados entre si pelos dois pontos). Este último é atribuído pelo próprio fabricante da interface.

Quando este sistema começa a se comunicar com os demais dispositivos integrantes da rede, é criado um cache contendo o mapeamento entre os endereços **MAC** com os endereços **IP** de cada um dos demais dispositivos.

Isto é feito através do protocolo **ARP** (AddressResolutionProtocol) e, devido a isto, denominamos esta tabela como "Tabela ARP". Para visualizar o conteúdo desta tabela, utilizamos o comando **arp**. Para visualizarmos endereços **IP** (ao contrário de nomes), utilizamos a opção **-n**:

```
[root@curso6:~]# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.1	ether	70:4f:57:d2:0e:b8	C		enp0s3

Neste momento, somente existe um único mapeamento na "Tabela ARP". Ou seja, o endereço **IP** 192.168.0.1 encontra-se mapeado ao endereço de hardware (**MAC**) 70:4f:57:d2:0e:b8. Aliás, é importante destacar que o comando **ip** também poderá ser utilizado para o mesmo propósito:

```
[root@curso6:~]# ipneigh show
```

Vamos realizar uma conexão via **SSH** com o host 192.168.0.105 e, em seguida, rodarmos novamente o comando **arp**. Perceberemos que a associação "**IPxMAC**" já devidamente posicionada na tabela:

```
[root@curso6:~]# ssh dltec@192.168.0.105
```

```
[root@curso6:~]# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.1	ether	70:4f:57:d2:0e:b8	C		enp0s3
192.168.0.105	ether	08:00:27:7c:33:c4	C		enp0s3

Também é possível editar manualmente o conteúdo da "Tabela ARP". Por exemplo, para adicionar um mapeamento "**IPxMAC**", basta utilizar a opção **-s**:

```
[root@curso6:~]# arp -s 192.168.0.109 02:02:02:02:02:02
```

```
[root@curso6:~]# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.109	ether	02:02:02:02:02:02	CM		enp0s3
192.168.0.1	ether	70:4f:57:d2:0e:b8	C		enp0s3
192.168.0.105	ether	08:00:27:7c:33:c4	C		enp0s3

Por outro lado, se for necessário remover (por exemplo) a associação criada anteriormente, basta utilizar a opção **-d**:

```
[root@curso6:~]# arp -d 192.168.0.109
```

```
[root@curso6:~]# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.1	ether	70:4f:57:d2:0e:b8	C		enp0s3
192.168.0.105	ether	08:00:27:7c:33:c4	C		enp0s3

3 Configuração Avançada de Rede e Soluções de Problemas

3.1 Introdução

Subtópico: 205.2 Peso: 4

Descrição do objetivo: Você deverá conseguir configurar um dispositivo de rede para implementar vários esquemas de autenticação de rede. Este objetivo inclui a configuração de um dispositivo de rede "multi-homed" e resolução de problemas de comunicação.

Áreas-chave:

- Utilitários para manipular tabelas de roteamento.
- Utilitários para configurar e manipular interfaces de rede ethernet.
- Utilitários para analisar o estado dos dispositivos de rede.
- Utilitários para monitorar e analisar o tráfego TCP/IP.

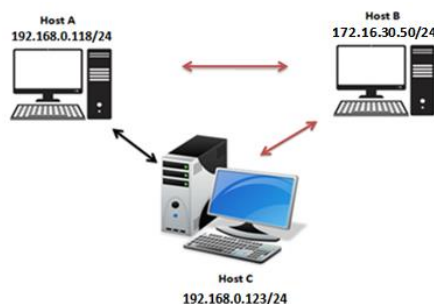
Principais termos, arquivos e utilidades:

- ip
- ifconfig
- route
- arp
- ss
- netstat
- lsof
- ping, ping6
- nc
- tcpdump
- nmap

3.2 Análises e Soluções de Problemas Relacionados a Conectividade e Rotas



É importante que você saiba que um sistema Linux pode atuar como um gateway entre duas redes distintas. Para isto, ele poderá utilizar duas placas – cada uma pertencendo a uma rede. Observe o cenário inicial que iremos trabalhar:



Como podemos perceber, o host C somente possui um endereço **IP** configurado, encontrando-se na mesma rede que o host A (192.168.0.0/24).

Dessa forma, já existe a comunicação entre ambos. Apenas como um teste, vamos executar o comando **ping** no host C para que sejam enviados quatro pacotes **ICMP** ao host A, de forma a testarmos a disponibilidade da conexão:

```
[root@curso6:~]# ping -c 4 192.168.0.118
```

```
PING 192.168.0.118 (192.168.0.118) 56(84) bytes of data.  
64 bytes from 192.168.0.118: icmp_seq=1 ttl=64 time=0.811 ms  
64 bytes from 192.168.0.118: icmp_seq=2 ttl=64 time=0.789 ms  
64 bytes from 192.168.0.118: icmp_seq=3 ttl=64 time=0.814 ms  
64 bytes from 192.168.0.118: icmp_seq=4 ttl=64 time=0.807 ms  
  
--- 192.168.0.118 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3004ms  
rtt min/avg/max/mdev = 0.789/0.805/0.814/0.022 ms
```

Perceba que mensagens "ECHO_REPLY" são obtidas a partir do host destinatário (como resposta às mensagens "ECHO_REQUEST" que lhe foram enviadas), indicando que ambos conseguem se comunicar com sucesso na rede.

Neste momento, inclusive, podemos rodar o comando **arp** em ambos os hosts envolvidos na comunicação e visualizar o mapeamento "IPxMAC" contido em suas "Tabelas ARP":

Host C

```
[root@curso6:~]# arp -n
Address                  HWtypeHWaddress        Flags Mask              Iface
192.168.0.118            ether    08:00:27:1a:41:18      C                       enp0s3
```

Host A

```
[root@curso6:~]# arp -n
Address                  HWtypeHWaddress        Flags Mask              Iface
192.168.0.123            ether    08:00:27:79:82:cc      C                       enp0s3
```

É importante destacarmos que o host de destino precisa suportar pacotes **ICMP** para que consiga enviar as mensagens "ECHO_REPLY" ao remetente. Muitos administradores têm desabilitado este recurso para que sejam evitados ataques de DOS (denial-of-service) em seus sistemas.

Outro comando que também poderá ser usado para realizar o teste de conexão é o **ping6**, responsável por enviar pacotes "ICMP6_ECHO_REQUEST" a um dado host, objetivando obter respostas "ICMP6_ECHO_REPLY". Devemos também saber que o comando **ping6** não é nada mais do que um simples link simbólico para o comando **ping**:

```
[root@curso6:~]# ls -l /bin/ping6
lrwxrwxrwx 1 root root 4 nov10 2016 /bin/ping6 -> ping
```

Sendo assim, podemos utilizar tanto o comando **ping6** como também o comando **ping** junto à opção **-6** para obter o mesmo resultado.

Como pudemos visualizar na imagem anterior, por enquanto o host C ainda não consegue comunicar-se com o host B, já que encontram-se em redes diferentes – da mesma forma que A ainda não consegue "conversar" diretamente com B.

Observe a saída da execução do comando **ping** no host C, indicando que, dos 4 pacotes **ICMP** transmitidos ao host B, nenhum foi recebido:

```
[root@curso6:~]# ping 172.16.30.50
connect: Network is unreachable
```

Nosso objetivo inicial será disponibilizar mais uma interface no host C para que esta faça parte da mesma rede que B. Mais a frente também configuraremos o roteamento necessário para que A e B comuniquem-se de forma direta – o host C atuará como um "gateway" entre ambas as redes.

Como já sabemos, os comandos **ip** ou **ifconfig** poderão ser utilizados para exibir as atuais configurações de endereço **IP** de um dado host. Observe a execução do comando **ifconfig** sobre o host C:

```
[root@curso6:~]# ifconfig -a

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.123 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe79:82cc prefixlen 64 scopeid 0x20<link>
ether 08:00:27:79:82:cc txqueuelen 1000 (Ethernet)
RX packets 81 bytes 9672 (9.4 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 63 bytes 6904 (6.7 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4098<BROADCAST,MULTICAST>mtu 1500
ether 08:00:27:53:46:33txqueuelen 1000 (Ethernet)
RX packets 19 bytes 2320 (2.2 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 35 bytes 4896 (4.7 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>mtu 65536
inet127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1 (Loopback Local)
RX packets 4 bytes 156 (156.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 4 bytes 156 (156.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Note que a interface "enp0s8" também encontra-se disponível, mas ainda não possui um endereço **IP** configurado. Para atribuir-lhe um endereço, vamos utilizar o comando **ifconfig** (o **ip** também poderia ser utilizado, como já estudamos):

```
[root@curso6:~]# ifconfig enp0s8 172.16.30.59/24

[root@curso6:~]# ifconfig enp0s8

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet172.16.30.59 netmask 255.255.255.0 broadcast 172.16.30.255
inet6 fe80::a00:27ff:fe53:4633 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:53:46:33txqueuelen 1000 (Ethernet)
RX packets 45 bytes 4950 (4.8 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 48 bytes 5646 (5.5 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

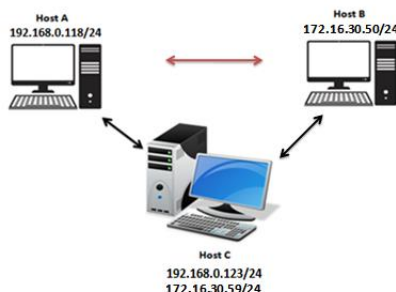
Agora, a partir do host C, vamos testar a comunicação com B:

```
[root@curso6:~]# ping -c 4 172.16.30.50

PING 172.16.30.50 (172.16.30.50) 56(84) bytes of data.
64 bytes from 172.16.30.50: icmp_seq=1 ttl=64 time=0.573 ms
64 bytes from 172.16.30.50: icmp_seq=2 ttl=64 time=0.367 ms
64 bytes from 172.16.30.50: icmp_seq=3 ttl=64 time=0.789 ms
64 bytes from 172.16.30.50: icmp_seq=4 ttl=64 time=0.998 ms

--- 172.16.30.50 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.367/0.681/0.998/0.237 ms
```

Veja que mensagens foram retornadas pelo host de destino, confirmando o sucesso da comunicação entre ambos. Dessa forma, nosso cenário atual é este:



Observe o conteúdo da "Tabela ARP" do host C:

```
[root@curso6:~]# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
172.16.30.50	ether	08:00:27:54:c3:31	C		enp0s8
192.168.0.118	ether	08:00:27:1a:41:18	C		enp0s3

Se neste momento tentarmos enviar pacotes **ICMP** a partir do host A em direção ao host B, veremos que a comunicação ainda não poderá ser estabelecida:

```
[root@curso6:~]# ping -c 4 172.16.30.50
connect: Network is unreachable
```

Este erro ocorre porque o host A ainda não conhece a rede 172.16.30.0/24 – não possuindo uma rota configurada para ela. Aliás, para visualizarmos a tabela de rotas do kernel referente ao host A, podemos lançar os comandos **route** ou **ip**. Veja a execução de ambos:

```
[root@curso6:~]# route -n
```

Kernel IP routingtable							
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3

```
[root@curso6:~]# ip route show

default via 192.168.0.1 dev enp0s3 onlink
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.118
```

Dessa forma, criaremos uma rota no host A de forma que todos os pacotes destinados a rede 172.16.30.0/24 sejam enviados à interface "enp0s3" do host C:

```
# Criação da rota no host A
```

```
[root@curso6:~]# route add -net 172.16.30.0/24 gw 192.168.0.123
```

```
# Visualizando a atual tabela de rotas do kernel
```

```
[root@curso6:~]# route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.30.0	192.168.0.123	255.255.255.0	UG	0	0	0	enp0s3
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3

Neste momento, a partir do host A, já podemos utilizar o comando **ping** para verificar que a interface "enp0s8" do host C já consegue receber as mensagens **ICMP** "ECHO_REQUEST":

```
[root@curso6:~]# ping -c 4 172.16.30.59
```

```
PING 172.16.30.59 (172.16.30.59) 56(84) bytes of data.  
64 bytes from 172.16.30.59: icmp_seq=1 ttl=64 time=0.771 ms  
64 bytes from 172.16.30.59: icmp_seq=2 ttl=64 time=0.823 ms  
64 bytes from 172.16.30.59: icmp_seq=3 ttl=64 time=0.405 ms  
64 bytes from 172.16.30.59: icmp_seq=4 ttl=64 time=0.822 ms
```

```
--- 172.16.30.59 ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3045ms  
rtt min/avg/max/mdev = 0.405/0.705/0.823/0.175 ms
```

Vamos agora até o host B e criar uma rota especificando que todos os pacotes destinados a rede 192.168.0.0/24 sejam enviados à interface "enp0s8" do host C:

```
[root@curso6:~]# ip route add 192.168.0.0/24 via 172.16.30.59
```

```
[root@curso6:~]# route -n
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.30.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3
192.168.0.0	172.16.30.59	255.255.255.0	UG	0	0	0	enp0s3

Apesar do host B já conseguir se comunicar com a interface "enp0s3" de C, a comunicação direta com A ainda não poderá ser estabelecida:

```
# Execução do comando ping no host B em direção à interface "enp0s3" do host C
```

```
[root@curso6:~]# ping -c 4 192.168.0.123
```

```
PING 192.168.0.123 (192.168.0.123) 56(84) bytes of data.  
64 bytes from 192.168.0.123: icmp_seq=1 ttl=64 time=1.46 ms  
64 bytes from 192.168.0.123: icmp_seq=2 ttl=64 time=0.811 ms  
64 bytes from 192.168.0.123: icmp_seq=3 ttl=64 time=0.809 ms  
64 bytes from 192.168.0.123: icmp_seq=4 ttl=64 time=0.401 ms
```

```
--- 192.168.0.123 ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3007ms  
rtt min/avg/max/mdev = 0.401/0.870/1.460/0.379 ms
```



```
# Tentativa de comunicação direta com o host A
```

```
[root@curso6:~]# ping -c 4 192.168.0.118
```

```
PING 192.168.0.118 (192.168.0.118) 56(84) bytes of data.
```

```
--- 192.168.0.118 ping statistics ---
```

```
4 packets transmitted, 0 received, 100% packet loss, time 3080ms
```

Para permitir que tanto o host A quanto B comuniquem-se diretamente, precisaremos ir ao host C e habilitar o kernel para que este consiga realizar o encaminhamento de pacotes de uma rede para outra, recurso conhecido como "IP Forward".

A habilitação deste recurso é determinada pelo conteúdo do arquivo **/proc/sys/net/ipv4/ip_forward**, cujo valor padrão é 0. Para fazer com que o host C aja como um gateway entre as duas redes, vamos utilizar o comando **sysctl** para configurar o seu valor como 1:

```
[root@curso6:~]# sysctl -w net.ipv4.ip_forward=1  
net.ipv4.ip_forward = 1
```

Agora podemos realizar o teste de comunicação direta entre os hosts A e B:

```
# Enviando pacotes ICMP ao host B
```

```
[root@curso6:~]# ping -c 4 172.16.30.50
```

```
PING 172.16.30.50 (172.16.30.50) 56(84) bytes of data.
```

```
64 bytes from 172.16.30.50: icmp_seq=1 ttl=63 time=0.877 ms
```

```
64 bytes from 172.16.30.50: icmp_seq=2 ttl=63 time=1.44 ms
```

```
64 bytes from 172.16.30.50: icmp_seq=3 ttl=63 time=0.756 ms
```

```
64 bytes from 172.16.30.50: icmp_seq=4 ttl=63 time=1.50 ms
```

```
--- 172.16.30.50 ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3023ms
```

```
rtt min/avg/max/mdev = 0.756/1.146/1.503/0.332 ms
```

```
# Enviando pacotes ICMP ao host A
```

```
root@curso6:~# ping -c 4 192.168.0.118
```

```
PING 192.168.0.118 (192.168.0.118) 56(84) bytes of data.
```

```
64 bytes from 192.168.0.118: icmp_seq=1 ttl=63 time=0.946 ms
```

```
64 bytes from 192.168.0.118: icmp_seq=2 ttl=63 time=0.704 ms
```

```
64 bytes from 192.168.0.118: icmp_seq=3 ttl=63 time=1.61 ms
```

```
64 bytes from 192.168.0.118: icmp_seq=4 ttl=63 time=1.53 ms
```

```
--- 192.168.0.118 ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3023ms
```

```
rtt min/avg/max/mdev = 0.704/1.200/1.617/0.388 ms
```

3.3 Testes Avançados de Conectividade



Em alguns cenários, o uso do comando **ping** para analisar a conexão entre dois hosts não é o bastante. Um administrador poderá desejar simular o tráfego de dados reais através de uma rede.

Uma das importantes ferramentas que poderá ser utilizada é o comando **nc** (também conhecido como **netcat**). Funcionando de forma semelhante ao comando **telnet** (mas minimizando erros de interpretação que eram comuns neste), o **nc** pode ser utilizado para realizar conexões em portas abertas em hosts remotos (ou até mesmo na máquina local).

Por exemplo, o comando pode ser usado para criar uma conexão em uma dada porta em um host local. A seguir, criaremos uma conexão na porta 80 (serviço **HTTP**) do host B (cujo **IP** é 192.168.0.32):

```
[root@curso6:~]#nc -l 80
```

Detalhes sobre esta conexão poderão ser vistos a partir do comando **lsof**, utilizado para exibir arquivos que encontram-se abertos no sistema, como sockets rede, arquivos comuns, diretórios, bibliotecas etc:

```
[root@curso6:~]# lsof -i :80
```

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
nc	3362	root	3u	IPv4	40963	0t0	TCP	*:http (LISTEN)

Agora, a partir do host A, podemos também usar o comando **nc** para testar a conectividade com a porta 80 oferecida por B:

```
[root@curso6:~]#nc 192.168.0.32 80
```

Como a conexão foi realizada com sucesso, ela representa um arquivo aberto no host B. Dessa maneira, seus detalhes poderão ser visualizados através do uso do comando **lsof**:

```
[root@curso6:~]# lsof -i :80

COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nc        2990 root    3u    IPv4  38944      0t0  TCP *:http (LISTEN)
nc        2990 root    4u    IPv4  38945      0t0  TCP 192.168.0.32:http-
>192.168.0.90:40396 (ESTABLISHED)
```

Qualquer coisa digitada no host A irá ser exibida no shell dedicado à criação da conexão no host B.

Outro exemplo de uso do comando **nc** é a verificação da disponibilidade de conexão em uma faixa de portas. No exemplo a seguir, utilizando o host A, vamos fechar a conexão (Ctrl + c) e em seguida, rodar o comando **nc** para identificar portas que encontram-se disponíveis no host B no range entre 80 a 90:

```
[root@curso6:~]# nc -vz 192.168.32 80-90

Connection to 192.168.0.90 80 port [tcp/http] succeeded!
nc: connect to 192.168.0.90 port 81 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 82 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 83 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 84 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 85 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 86 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 87 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 88 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 89 (tcp) failed: Connection refused
nc: connect to 192.168.0.90 port 90 (tcp) failed: Connection refused
```

Outro comando muito útil que surge neste sentido é o **netstat**, responsável por (dentre outras coisas) exibir as conexões de rede que foram estabelecidas. Para visualizarmos as conexões **TCP** que foram estabelecidas, utilizamos a opção **-t**:

```
[root@curso6:~]# netstat -t

Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.0.32:www-http   192.168.0.90:40402      ESTABLISHED
```

Para exibir todas (**-a**) as conexões **TCP** (**-t**) em modo "LISTEN" ou aquelas "ESTABLISHED", não havendo o processo de resolução de nomes (**-n**) lançamos o comando da seguinte forma:

```
[root@curso6:~]# netstat -ta

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25             0.0.0.0:*               LISTEN
tcp        0      0 192.168.0.32:80          192.168.0.90:40402      ESTABLISHED
tcp6       0      0 :::22                   :::*                     LISTEN
tcp6       0      0 :::1:631                 :::*                     LISTEN
tcp6       0      0 :::1:25                   :::*                     LISTEN
```

O comando **netstat** consegue ainda exibir outras diversas informações importantes para o diagnóstico e aplicação de soluções em problemas relacionados a rede. Apesar de já ter sido praticamente substituído por outros comandos, como o **ss** por exemplo, ele ainda costuma ser bastante utilizado.

Ao ser executado sem quaisquer opções, o comando **netstat** exibe uma lista completa de todos os sockets/conexões atualmente abertos/ativos. Observe um exemplo parcial e adaptado:

```
[root@curso6:~]# netstat | less
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.0.32:www-http   192.168.0.90:40402     ESTABLISHED

Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags               Type                   State                  I-Node   Path
unix    2      [ ]                 DGRAM                  27674                 /run/user/1000/
unix    2      [ ]                 DGRAM                  19235                 /var/run/chrony/
unix    2      [ ]                 DGRAM                  25281                 /run/user/463/systemd/
unix    2      [ ]                 DGRAM                  13513                 /run/systemd/journal/
unix    3      [ ]                 DGRAM                  10468                 /run/systemd/notify
```

Se a intenção também for visualizar aqueles sockets que estejam em modo "LISTEN", basta utilizar a opção **-a** (ou **--all**). Se o administrador desejar visualizar os **PIDs** e nomes dos programas relacionados a cada um desses sockets, a opção **-p** (ou **--program**) também poderá ser usada.

No dia a dia, o processo de pesquisa por sockets se dá de maneira mais refinada. No exemplo a seguir, vamos exibir os sockets **TCP** (**-t**) e **UDP** (**-u**) que estejam em modo "LISTEN" (**-l**), não realizando o processo de resolução de nomes (**-n**):

```
[root@curso6:~]# netstat -tuln

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::1:631                 :::*                    LISTEN
tcp6       0      0 :::1:25                   :::*                    LISTEN
udp        0      0 0.0.0.0:5353             0.0.0.0:*               LISTEN
udp        0      0 127.0.0.1:323           0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:43626           0.0.0.0:*               LISTEN
udp6       0      0 :::177                   :::*                    LISTEN
udp6       0      0 :::5353                   :::*                    LISTEN
udp6       0      0 :::1:323                  :::*                    LISTEN
udp6       0      0 :::60224                  :::*                    LISTEN
```

O comando **netstat** também poderá ser usado para exibir a tabela de roteamento IP do kernel, bastando para isto utilizar a opção **-r** (ou **--route**).

```
[root@curso6:~]# netstat -r

Kernel IP routing table
Destination      Gateway          Genmask          Flags        MSS Window  irttIface
192.168.0.0      0.0.0.0          255.255.255.0    U            0 0        0 eth0
```

As estatísticas relacionadas às interfaces também poderão ser exibidas através do uso dos comandos **ip**, **ifconfig** e **netstat**. Acompanhe:

```
[root@curso6:~]#netstat -i
```

Kernel Interface table

Iface	MTU	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Flg
eth0	1500	545	0	0	0	418	0	0	0	BMRU
lo	65536	148	0	0	0	148	0	0	0	LRU

```
[root@curso6:~]#ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.32 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe5a:4e7a prefixlen 64 scopeid 0x20<link>
ether 08:00:27:5a:4e:7a txqueuelen 1000 (Ethernet)
RX packets 545 bytes 54032 (52.7 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 418 bytes 34090 (33.2 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>mtu 65536
inet127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 148 bytes 12392 (12.1 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 148 bytes 12392 (12.1 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[root@curso6:~]# ip -s link
```

```
1: lo: <LOOPBACK,UP,LOWER_UP>mtu 65536 qdiscnoqueue state UNKNOWN mode DEFAULT
group default qlen 1000
link/loopback 00:00:00:00:00:00brd 00:00:00:00:00:00
RX: bytes  packets  errors  dropped overrun mcast
12392      148      0       0       0       0
TX: bytes  packets  errors  dropped carrier collsns
12392      148      0       0       0       0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP mode
DEFAULT group default qlen 1000
link/ether 08:00:27:5a:4e:7a brdff:ff:ff:ff:ff:ff
RX: bytes  packets  errors  dropped overrun mcast
54032      545      0       0       0       168
TX: bytes  packets  errors  dropped carrier collsns
34090      418      0       0       0       0
```

Já o comando **ss**, outro importante utilitário para investigação de sockets e atuante como substituto ao **netstat**, ao ser utilizado sem quaisquer opções, é exibida uma lista de sockets em estado "não-LISTEN" e que possuam conexões estabelecidas. Observe um exemplo parcial:

```
[root@curso6:~]# ss
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
u_str	ESTAB	0	0	* 29070	* 29071
u_str	ESTAB	0	0	* 26747	* 26748
u_str	ESTAB	0	0	* 30982	* 30983
u_str	ESTAB	0	0	* 26274	* 26275
u_str	ESTAB	0	0	* 25983	* 25984
u_str	ESTAB	0	0	* 31011	* 31012
u_str	ESTAB	0	0	* 25771	* 25772

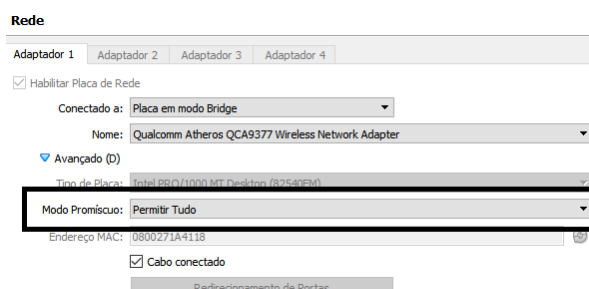
Seguindo a mesma lógica que vimos no **netstat**, se utilizarmos o comando **ss** junto às opções "lunt", serão exibidos os sockets **TCP** e **UDP** em modo **LISTEN** (além de não haver o processo de resolução de nomes). Além disso, também é possível acrescentar a opção **-p** (ou **--process**) para visualizar os **PIDs** dos processos relacionados a cada socket.

3.4 Escaneamento de Portas e Análise do Tráfego de Pacotes



O comando **tcpdump**, instalável a partir do pacote de mesmo nome, é responsável por analisar o tráfego geral de pacotes em um sistema.

Como estamos utilizando máquinas virtuais, será preciso ir até suas configurações e, nas interfaces de rede e, em Modo Promíscuo, selecionar "Permitir tudo". Com isso, torna-se possível identificar todos os pacotes em tráfego pelo host:



Quando nenhuma interface de rede é informada ao comando, o **tcpdump** escolhe aquela de menor "valor" para realizar suas operações (excluindo aquela de loopback). Ou seja, se existirem as interfaces "enp0s3" e "enp0s8", o padrão é que o comando utilize a primeira para "ouvir" o tráfego de pacotes no sistema:

```
[root@curso6:~]# tcpdump
```

```
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
```

```
16:02:21.929610 IP 192.168.0.52.64673 > 192.168.0.100.8009: Flags [.), seq
2987749718:2987749719, ack 529502624, win 510, length 1
16:02:21.934790 IP 192.168.0.100.8009 > 192.168.0.52.64673: Flags [.), ack 1, win
1419, options [nop,nop,sack 1 {0:1}], length 0
16:02:21.990399 IP 192.168.0.52.64906 > 192.168.0.100.8009: Flags [.), seq
1816658969:1816658970, ack 132698872, win 508, length 1
16:02:21.992726 IP 192.168.0.100.8009 > 192.168.0.52.64906: Flags [.), ack 1, win
1419, options [nop,nop,sack 1 {0:1}], length 0
16:02:23.947079 IP 192.168.0.52.64911 > 192.168.0.100.8009: Flags [P.), seq
234:351, ack 235, win 509, length 117
16:02:23.951950 IP 192.168.0.100.8009 > 192.168.0.52.64911: Flags [P.), seq
235:352, ack 351, win 1419, length 117
```

No exemplo a seguir, estamos especificando que a interface "enp0s8" será aquela responsável por "ouvir" o tráfego:

```
[root@curso6:~]#tcpdump -i enp0s8
```

Também é possível monitorar o tráfego relacionado a uma determinada porta, sem que haja (por exemplo) o processo de resolução de nomes. Observe:

```
root@curso6:~# tcpdump -n port 22
```

Neste caso, o comando irá tomar o shell corrente para monitorar todo o tráfego relacionado à porta 22 (**SSH**).

Se quisermos, também podemos registrar o monitoramento do comando em um arquivo, de forma que este seja aberto posteriormente por ferramentas especializadas em análise de pacotes, como o Wireshark, por exemplo. Para isso, utilizamos a opção **-w**:

```
[root@curso6:~]#tcpdump -n port 22 -w resultado.pcap
```

Para ler o conteúdo do arquivo, utilizamos o comando junto à opção **-r**:

```
[root@curso6:~]#tcpdump -r resultado.pcap
```

Já o comando **nmap** (instalável a partir do pacote de mesmo nome) é responsável por escanear portas abertas tanto no sistema local quanto em máquinas disponíveis na rede (ou a própria rede inteira) – bastante útil em processos de auditoria. Por exemplo, estando em um host, vamos escanear as portas abertas em outro (a opção **-v** poderá ser utilizada para exibir mais detalhes):

```
[root@curso6:~]# nmap 192.168.0.123 -v
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2019-11-05 17:25 -03
Initiating ARP Ping Scan at 17:25
Scanning 192.168.0.123 [1 port]
Completed ARP Ping Scan at 17:25, 0.20s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:25
```

```
Completed Parallel DNS resolution of 1 host. at 17:25, 0.01s elapsed
Initiating SYN Stealth Scan at 17:25
Scanning 192.168.0.123 [1000 ports]
Discovered open port 22/tcp on 192.168.0.123
Discovered open port 23/tcp on 192.168.0.123
Discovered open port 21/tcp on 192.168.0.123
Discovered open port 80/tcp on 192.168.0.123
Discovered open port 111/tcp on 192.168.0.123
Completed SYN Stealth Scan at 17:25, 2.65s elapsed (1000 total ports)
Nmap scan report for 192.168.0.123
Host is up (0.00033s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  openssh
23/tcp    open  telnet
80/tcp    open  http
111/tcp   open  rpcbind
MAC Address: 08:00:27:53:46:33 (Oracle VirtualBox virtual NIC)
```

```
Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.91 seconds
Raw packets sent: 1008 (44.320KB) | Rcvd: 1007 (40.288KB)
```

Para escanear todos os hosts da rede 192.168.0.0/24, basta executarmos da seguinte forma:

```
[root@curso6:~]#nmap 192.168.0.0/24
```

Para excluir um host do comando, basta utilizar a opção "--exclude":

```
[root@curso6:~]#nmap 192.168.0.0/24 --exclude 192.168.0.23
```

Também podemos especificar um range de hosts a serem analisados. Por exemplo, o comando a seguir realiza o escaneamento pelos hosts 192.168.0.1 até 192.168.0.50:

```
[root@curso6:~]#nmap 192.168.0.1-50
```

Para fazer com que o comando trabalhe com endereços **IP** contidos em um arquivo, as opções **-iL** poderão ser utilizadas. Acompanhe:

```
[root@curso6:~]#cat enderecos.txt
192.168.0.118
192.168.0.111
```

```
[root@curso6:~]#nmap -iL enderecos.txt
```

Para descobrir outras informações importantes, como o tipo de sistema operacional utilizado pelo host (incluindo a versão do kernel), podemos lançar a opção **-A**:

```
[root@curso6:~]# nmap -A 192.168.0.111
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2019-11-05 17:38 -03
Nmap scan report for 192.168.0.111
Host is up (0.00044s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
6000/tcp  open  X11      (access denied)
MAC Address: 08:00:27:A1:85:BE (Oracle VirtualBox virtual NIC)
Device type: general purpose
```



```
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.6
Network Distance: 1 hop
Service Info: OS: Unix
```

TRACEROUTE

HOP	RTT	ADDRESS
1	0.44 ms	192.168.0.111

OS and Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .
Nmap done: 1 IP address (1 host up) scanned in 12.27 seconds

4 Troubleshooting para Problemas de Redes

4.1 Introdução

Subtópico: 205.3 Peso: 4

Descrição do objetivo: Você deverá conseguir identificar e corrigir problemas comuns relacionados a configuração de redes, incluindo o conhecimento sobre as localizações de arquivos básicos e comandos.

Áreas-chave:

- Localização e conteúdo de arquivos de restrição de acesso.
- Utilitários para configurar e manipular interfaces de rede ethernet.
- Utilitários para gerenciar tabelas de roteamento.
- Utilitários para listar os estados da rede.
- Utilitários para obter informações sobre as configurações de rede.
- Métodos de informação sobre os dispositivos de hardware reconhecidos e utilizados.
- Arquivos de inicialização do sistema e seus conteúdos (processo init do SysV).
- Ciência a respeito do NetworkManager e seu impacto sobre a configuração da rede.

Principais termos, arquivos e utilidades:

- ip
- ifconfig
- route
- ss
- netstat
- /etc/network/, /etc/sysconfig/network-scripts/
- ping, ping6
- traceroute, traceroute6
- mtr
- hostname
- Arquivos de log do sistema, como /var/log/syslog, /var/log/messages e o journal do systemd
- dmesg

- /etc/resolv.conf
- /etc/hosts
- /etc/hostname, /etc/HOSTNAME
- /etc/hosts.allow, /etc/hosts.deny

4.2 Análise de Estatísticas e Depuração de Rotas



O Linux oferece ferramentas diversas que poderão ser utilizadas durante processos de análises e soluções de problemas relacionados ao subsistema de rede. Os comandos **ifconfig** e **ip** surgem como principais atores no sentido de entender como as interfaces disponíveis estão se comportando, através de estatísticas exibidas.

A seguir é ilustrada a execução do comando **ifconfig** em um sistema que possui duas placas de rede instaladas:

```
[root@curso6:~]# ifconfig

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.75 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe5c:ffab prefixlen 64 scopeid 0x20<link>
ether 08:00:27:5c:ff:ab txqueuelen 1000 (Ethernet)
RX packets 165 bytes 15597 (15.2 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 77 bytes 7950 (7.7 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>mtu 65536
inet127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1 (Local Loopback)
RX packets 48 bytes 3216 (3.1 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 48 bytes 3216 (3.1 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Apesar de duas interfaces encontrarem-se disponíveis neste sistema, somente uma é exibida ("enp0s3"). Não esqueça que, ao ser utilizado sem quaisquer opções, o comando **ifconfig** somente exibe aquelas interfaces ativas. Sendo assim, para também visualizar as inativas, a opção **-a** deverá ser utilizada:

```
[root@curso6:~]# ifconfig -a

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.75 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe5c:ffab prefixlen 64 scopeid 0x20<link>
ether 08:00:27:5c:ff:ab txqueuelen 1000 (Ethernet)
RX packets 165 bytes 15597 (15.2 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 77 bytes 7950 (7.7 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<BROADCAST,MULTICAST>mtu 1500
inet172.16.0.65 netmask 255.255.0.0 broadcast 172.16.0.255
inet6 fe80::a00:27ff:fe57:f297 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:57:f2:97 txqueuelen 1000 (Ethernet)
RX packets 147 bytes 11890 (11.6 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 73 bytes 7665 (7.4 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>mtu 65536
inet127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1 (Local Loopback)
RX packets 48 bytes 3216 (3.1 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 48 bytes 3216 (3.1 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Note que agora podemos visualizar que a interface "enp0s8" não encontra-se ativa ("UP") e nem em execução ("RUNNING"). Já o comando **ip**, quando utilizado com o objeto "link" e ação "show", irá apontar o estado corrente de cada uma das interfaces:

```
[root@curso6:~]# ip address show

1: lo: <LOOPBACK,UP,LOWER_UP>mtu 65536 qdiscnoqueue state UNKNOWN group default
qlen 1
link/loopback 00:00:00:00:00:00brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:5c:ff:ab brdff:ff:ff:ff:ff:ff
inet 192.168.0.75/24 brd 192.168.0.255 scope global enp0s3
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe5c:ffab/64 scope link
valid_lft forever preferred_lft forever

3: enp0s8: <BROADCAST,MULTICAST>mtu 1500 qdiscpfifo_fast state DOWN group default
qlen 1000
link/ether 08:00:27:57:f2:97 brdff:ff:ff:ff:ff:ff
```

```
inet 172.16.0.65/16 brd 172.16.0.255 scope global enp0s8
valid_lft forever preferred_lft forever
```

Veja que a interface "enp0s8" não apresenta as string "UP" (indicando que a interface está habilitada) e "LOWER_UP" (indicando que a interface está conectada à rede), conforme vistas na interface "enp0s3". Além disso, é exibida a string "DOWN" em sua saída, indicando a inoperabilidade da interface.

Para "levantar" a interface, ambos os comandos poderão ser usados:

```
[root@curso6:~]# ip link set enp0s8 up
```

```
[root@curso6:~]# ifconfig enp0s8 up
```

Outro fator importante a ser verificado em processos de troubleshooting são as estatísticas relacionadas a cada interface – ou seja, como elas estão se comportando durante os procedimentos de envio e recebimento de pacotes.

Por padrão, o comando **ifconfig** já exibe essas estatísticas em sua saída. Para visualizá-las através do comando **ip**, a opção **-s** (**-stats** ou **-statistics**) deverá ser usada junto ao objeto "link":

```
[root@curso6:~]# ip -s link
```

```
1: lo: <LOOPBACK,UP,LOWER_UP>mtu 65536 qdiscnoqueue state UNKNOWN mode DEFAULT
group default qlen 1
link/loopback 00:00:00:00:00:00brd 00:00:00:00:00:00
RX: bytes  packets  errors  dropped overrun mcast
3216      48        0       0        0        0
TX: bytes  packets  errors  dropped carrier collsns
3216      48        0       0        0        0

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
mode DEFAULT group default qlen 1000
link/ether 08:00:27:5c:ff:ab brdff:ff:ff:ff:ff:ff
RX: bytes  packets  errors  dropped overrun mcast
28064     238      0       0        0       101
TX: bytes  packets  errors  dropped carrier collsns
8426      82       0       0        0        0

3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
mode DEFAULT group default qlen 1000
link/ether 08:00:27:57:f2:97 brdff:ff:ff:ff:ff:ff
RX: bytes  packets  errors  dropped overrun mcast
14965     189      0       0        0       57
TX: bytes  packets  errors  dropped carrier collsns
10569     97       0       0        0        0
```

As estatísticas das interfaces são divididas em dois grupos: Transmissão (**TX**) e Recepção (**RX**). Para cada um deles, são exibidas informações importantes, como a quantidade de bytes e pacotes manipulados, a quantidade daqueles que apresentaram erros ("errors") durante os processos de recebimento ou transmissão, quantos foram descartados ("dropped"), quantos foram desconsiderados por questões de sobrecarga ("overrun") etc.

Sendo assim, como um ponto inicial do procedimento de troubleshooting, essas são informações preciosas que o administrador deverá sempre monitorar.

Outra questão importante é a rota que deverá ser tomada por um dado pacote. Vejamos o cenário corrente:

```
[root@curso6:~]# route -n
```

```
Kernel IP routingtable
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.0.0	0.0.0.0	255.255.0.0	U	0	0	0	enp0s8
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3

```
[root@curso6:~]# ip route show
```

```
172.16.0.0/16 dev enp0s8 proto kernel scope link src 172.16.0.65  
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.75
```

Note que a rota padrão ainda não encontra-se configurada. Se neste momento tentarmos enviar pacotes **ICMP** a algum host externo, a seguinte mensagem será apresentada:

```
[root@curso6:~]# ping -c 4 www.google.com
```

```
connect: Network isunreachable
```

Após determinar o endereço **IP** do default gateway, podemos utilizar ambos os comandos:

```
# Adicionando a rota padrão com o route
```

```
[root@curso6:~]# route add default gw 192.168.0.1
```

```
# Adicionando a rota padrão com o ip
```

```
[root@curso6:~]# ip route add default via 192.168.0.1
```

```
# Verificando a atualconfiguração
```

```
[root@curso6:~]# ip route show
```

```
default via 192.168.0.1 dev enp0s3  
172.16.0.0/16 dev enp0s8 proto kernel scope link src 172.16.0.65  
192.168.0.0/24 dev enp0s3 proto kernel scope link src 192.168.0.75
```

Com a rota padrão configurada, vamos utilizar o comando **ping** para enviar pacotes **ICMP** ao host externo:

```
[root@curso6:~]# ping -c 4 www.google.com

PING www.google.com (172.217.29.100) 56(84) bytes of data.
64 bytes from rio01s24-in-f4.1e100.net (172.217.29.100): icmp_seq=1 ttl=52
time=21.2 ms
64 bytes from rio01s24-in-f4.1e100.net (172.217.29.100): icmp_seq=2 ttl=52
time=53.3 ms
64 bytes from rio01s24-in-f4.1e100.net (172.217.29.100): icmp_seq=3 ttl=52
time=19.4 ms
64 bytes from rio01s24-in-f4.1e100.net (172.217.29.100): icmp_seq=4 ttl=52
time=17.8 ms

--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 17.809/27.954/53.305/14.688 ms
```

O comando **traceroute** também poderá ser bastante útil no sentido de determinar a rota que os pacotes estão tomando da origem até o destino, utilizando o campo **TTL** (Time To Live) no cabeçalho do **IP** para realizar suas operações. Este campo determina quantos "saltos" um pacote em particular irá tomar ao trafegar por uma rede:

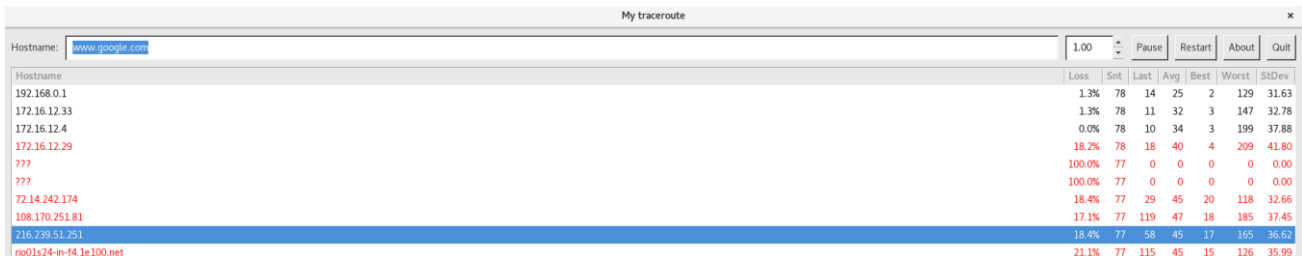
```
[root@curso6:~]# traceroute www.lipsum.com
traceroute to www.lipsum.com (184.173.177.52), 30 hops max, 60 byte packets
 1  192.168.0.1 (192.168.0.1)  2.175 ms  2.134 ms  2.104 ms
 2  172.16.12.33 (172.16.12.33)  6.537 ms  9.295 ms  13.533 ms
 3  172.16.12.4 (172.16.12.4)  13.528 ms  13.573 ms  13.570 ms
 4  172.16.12.29 (172.16.12.29)  13.852 ms  13.849 ms  14.062 ms
 5  * * *
 6  * * *
 7  tel-7.bbr01.tl01.nyc01.networklayer.com (198.32.160.27)  132.348 ms  137.855
ms  143.065 ms
 8  ae5.cbs01.tl01.nyc01.networklayer.com (50.97.17.38)  144.395 ms  144.761 ms
143.044 ms
 9  ae2.cbs01.cs01.wdc05.networklayer.com (169.45.18.188)  146.370 ms  151.125 ms
158.990 ms
10  * * *
11  b9.10.35a9.ip4.static.sl-reverse.com (169.53.16.185)  135.262 ms
bb.10.35a9.ip4.static.sl-reverse.com (169.53.16.187)  136.703 ms
b9.10.35a9.ip4.static.sl-reverse.com (169.53.16.185)  137.867 ms
12  c9.76.2bd0.ip4.static.sl-reverse.com (208.43.118.201)  146.575 ms
cb.76.2bd0.ip4.static.sl-reverse.com (208.43.118.203)  142.827 ms  139.745 ms
13  www.lipsum.com (184.173.177.52)  154.384 ms  144.548 ms  144.315 ms
```

No percurso, poderá ocorrer de algum roteador não oferecer resposta dentro de um certo período de tempo. Sendo assim, um asterisco será exibido para aquele salto:

Por padrão, o programa irá tentar resolver o nome que lhe fora passado, escolhendo automaticamente o protocolo apropriado (**IPv4** ou **IPv6**). Se o host destino retornar ambos, o comando irá utilizar o **IPv4**. Entretanto é possível forçá-lo a utilizar o **IPv4** (opção **-4**) ou **IPv6** (opção **-6**). A opção **-6** também possui um comando equivalente: o **traceroute6**.

Outro utilitário importante é o **mtr** (Mytraceroute), combinando funcionalidades oferecidas pelos comandos **traceroute** e **ping** (instalável a partir do pacote de mesmo nome):

```
[root@curso6:~]# mtr www.google.com
```



The screenshot shows the mtr utility interface. The Hostname is set to www.google.com. The table displays the following data:

Hostname	Loss	Snt	Last	Avg	Best	Worst	StDev
192.168.0.1	1.3%	78	14	25	2	129	31.63
172.16.12.33	1.3%	78	11	32	3	147	32.78
172.16.12.4	0.0%	78	10	34	3	199	37.88
172.16.12.29	18.2%	78	18	40	4	209	41.80
???	100.0%	77	0	0	0	0	0.00
???	100.0%	77	0	0	0	0	0.00
72.14.242.174	18.4%	77	29	45	20	118	32.66
108.170.251.81	17.1%	77	119	47	18	185	37.45
216.239.51.251	18.4%	77	58	45	17	165	36.62
ro01s24-in-f4.1e100.net	21.1%	77	115	45	15	126	35.99

Ao ser executado, o comando verifica a rota até o destino e vai exibindo em tempo real as estatísticas importantes relacionadas aos envios dos pacotes, como a porcentagem de perda, a média de velocidade dos "pings" etc.

4.3 Análise de Estado das Conexões de Rede

Os sockets de conexão do sistema também poderão ser investigados a partir de ferramentas diversas. Por exemplo, o comando **netstat**, como já estudamos, é de suma importância ao processo de troubleshooting, já que permite ao administrador visualizar aquelas conexões ativas (estabelecidas) no momento, por exemplo. Observe sua execução:

```
[root@curso6:~]# netstat -tun
```

Active Internet connections (w/o servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
-------	--------	--------	---------------	-----------------	-------

Por enquanto não existem conexões **TCP** ou **UDP** ativas no sistema. Porém, se utilizarmos outro sistema para conectar-se a este, esta conexão já poderá ser visualizada:

```
[root@curso6:~]# netstat -tun
```

Active Internet connections (w/o servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	192.168.0.118:22	192.168.0.126:41200	ESTABLISHED

Veja que o host remoto (identificado no campo "ForeignAddress") encontra-se conectado à porta 22 (**SSH**) do sistema local – a conexão encontra-se estabelecida (campo "State").

Alguns outros estados também poderão ser exibidos neste campo, como "CLOSE_WAIT" (sistema remoto finalizou a conexão e o socket está sendo fechado), "TIME_WAIT" (o socket foi fechado, mas está manipulando pacotes que ainda encontram-se na rede) e "LISTEN" (o socket está aguardando por conexões).

Para exibir somente aqueles sockets prontos para receberem conexões (ou seja, em estado "LISTEN"), utilizamos a opção "l":

```
[root@curso6:~]# netstat -tlun
```

Active Internet connections (only servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:23	0.0.0.0:*	LISTEN
tcp6	0	0	:::21	:::*	LISTEN
tcp6	0	0	:::22	:::*	LISTEN
tcp6	0	0	:::1:631	:::*	LISTEN
udp	0	0	127.0.0.53:53	0.0.0.0:*	
udp	0	0	0.0.0.0:631	0.0.0.0:*	
udp	0	0	0.0.0.0:50385	0.0.0.0:*	
udp	0	0	0.0.0.0:5353	0.0.0.0:*	
udp6	0	0	:::5353	:::*	
udp6	0	0	:::39750	:::*	

Se a intenção for exibir tanto os sockets em modo "LISTEN" quanto aqueles que já possuam conexões ativas, utilizamos a opção -a:

```
[root@curso6:~]# netstat -talun
```

Active Internet connections (only servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:23	0.0.0.0:*	LISTEN
tcp	0	0	192.168.0.118:22	192.168.0.126:41200	ESTABLISHED
tcp6	0	0	:::21	:::*	LISTEN
tcp6	0	0	:::22	:::*	LISTEN
tcp6	0	0	:::1:631	:::*	LISTEN
udp	0	0	127.0.0.53:53	0.0.0.0:*	
udp	0	0	0.0.0.0:631	0.0.0.0:*	
udp	0	0	0.0.0.0:50385	0.0.0.0:*	
udp	0	0	0.0.0.0:5353	0.0.0.0:*	
udp6	0	0	:::5353	:::*	
udp6	0	0	:::39750	:::*	

Se abrirmos um navegador web, por exemplo, veremos que diversas outras conexões são abertas. Obviamente, essas deverão ser constantemente monitoradas pelo administrador:

```
[root@curso6:~]# netstat -nut
```

Active Internet connections (w/o servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	192.168.0.118:53352	192.16.58.8:80	ESTABLISHED
tcp	0	0	192.168.0.118:42678	200.149.59.83:80	ESTABLISHED
tcp	0	0	192.168.0.118:49444	200.149.59.81:80	ESTABLISHED
tcp	0	0	192.168.0.118:51380	35.160.208.145:443	ESTABLISHED
tcp	0	0	192.168.0.118:56262	54.186.106.198:443	ESTABLISHED
tcp	0	0	192.168.0.118:51160	208.80.154.240:443	ESTABLISHED
tcp	0	0	192.168.0.118:22	192.168.0.126:41200	ESTABLISHED
tcp	0	0	192.168.0.118:51228	208.80.154.224:443	ESTABLISHED

Repare que as portas 80 e 443 são aquelas utilizadas nos outros sistemas remotos, indicando os serviços de **HTTP** e **HTTPS**, respectivamente. Se fecharmos o navegador e executarmos imediatamente o comando anterior, note que os estados de cada uma das conexões que foram estabelecidas com o website são alterados:

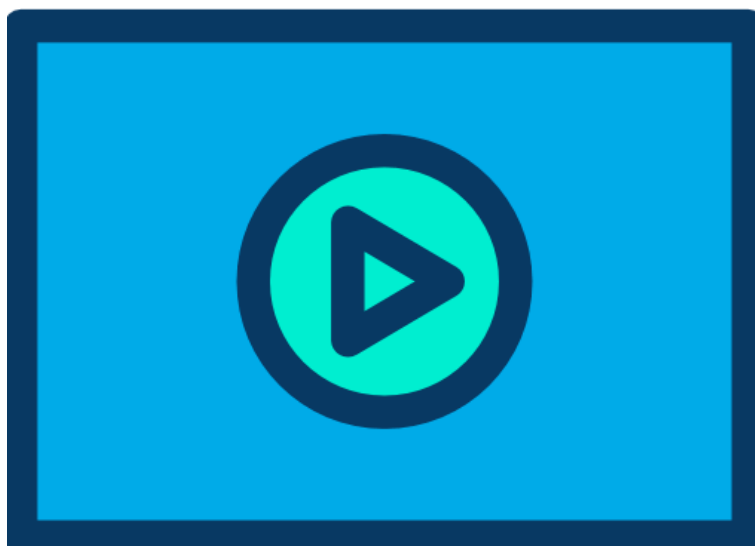
```
[root@curso6:~]# netstat -nut
```

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 192.168.0.118:42678    200.149.59.83:80       TIME_WAIT
tcp      0      0 192.168.0.118:49444    200.149.59.81:80       TIME_WAIT
tcp      0      0 192.168.0.118:22      192.168.0.126:41204    ESTABLISHED
tcp      0      0 192.168.0.118:34668    54.230.57.250:443      TIME_WAIT
tcp      0      0 192.168.0.118:51228    208.80.154.224:443     TIME_WAIT
tcp      0      0 192.168.0.118:58132    54.230.57.28:443       TIME_WAIT
tcp      0      0 192.168.0.118:50600    54.230.57.170:443      TIME_WAIT
```

Como já estudamos, o comando **ss** é também importante neste cenário, já que consegue visualizar diversas estatísticas relacionadas aos sockets de rede, funcionando de forma similar ao **netstat**, mas indo além e trazendo informações mais detalhadas. O próprio manual do comando **netstat** informa que este já se encontra praticamente obsoleto – já que outras ferramentas (incluindo o comando **ss**) poderão ser utilizados para os mesmos propósitos.

Entretanto, para o exame, é importante estar atento sobre o comportamento geral do **ss** ao ser utilizado nesses procedimentos de troubleshooting. Por exemplo, quando nenhuma opção é informada ao comando, ele exibe sockets que já possuem conexões estabelecidas – não importando o tipo de socket (TCP/UNIX/UDP). Além disso, as opções **-p** (ou **--process**) e **-m** (ou **--memory**) são importantes a serem memorizadas para o dia do exame. Enquanto a primeira é responsável por exibir o **PID** do processo relacionado ao socket, a segunda exibe a quantidade de memória a ele alocada.

4.4 Aplicação de Configurações Permanentes nas Interfaces



É importante não esquecer que as configurações de endereçamento **IP** e de rotas que aplicamos através dos comandos **ifconfig**, **ip** e **route** são de caráter temporário. Por isso, quando reiniciamos o sistema, por exemplo, essas configurações são perdidas. Para que se tornem permanentes, é necessário envolver arquivos de configuração.

Em distribuições Debian-like, o arquivo a ser utilizado é o **/etc/network/interfaces**. Por exemplo, vamos trabalhar com a interface "enp0s3" de um host. Acompanhe:

```
# Visualizando as atuais configurações de endereçamento
```

```
[root@curso6:~]#ifconfig enp0s3
```

```
enp0s3: flags=4098<BROADCAST,MULTICAST>mtu 1500
ether 08:00:27:78:e7:99 txqueuelen 1000 (Ethernet)
RX packets 118 bytes 38225 (38.2 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 136 bytes 13551 (13.5 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
# Aplicando as configurações da interface em /etc/network/interfaces
```

```
auto enp0s3
iface enp0s3 inet static
address 192.168.0.78
netmask 255.255.255.0
gateway 192.168.0.1
network 192.168.0.0
broadcast 192.168.0.255
up route add -net 172.16.30.0/24 gw 192.168.0.44 dev enp0s3
```

```
# Reiniciando o serviço de rede
```

```
[root@curso6:~]#systemctlrestart networking
```

```
# Visualizando as atuais configurações de endereçamento
```

```
[root@curso6:~]# ifconfig enp0s3
```

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.78 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::2884:17ee:ca57:a265 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:78:e7:99 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Através da instrução "auto" contida em **/etc/network/interfaces**, a interface em questão será sempre ativada quando o sistema for iniciado ou quando a rede for reiniciada.

Se as diretivas "network" ou "broadcast" não forem declaradas, o sistema irá calcular automaticamente essas configurações, baseando-se no **IP** e máscara de rede informadas.

Note também que configuramos a rota para a rede 172.16.30.0/24 no arquivo e, devido a isto, esta rota sempre estará disponível. Como podemos perceber, o gateway para esta rede é o host 192.168.0.44. Observe a tabela de roteamento:

```
[root@curso6:~]#ifconfig enp0s3

0.0.0.0          192.168.0.1      0.0.0.0          UG        0          0          0 enp0s3
172.16.30.0      192.168.0.44     255.255.255.0    UG        0          0          0 enp0s3
192.168.0.0      0.0.0.0          255.255.255.0    U         0          0          0 enp0s3
```

Agora, vamos aplicar uma configuração de forma que a interface "enp0s3" obtenha sempre o seu endereço via **DHCP** (de forma dinâmica). Como já possui configurações relacionadas em **/etc/network/interfaces**, podemos lançar o comando **ifdown** para derrubar a interface. Acompanhe:

```
[root@curso6:~]#ifdown enp0s3

# Aplicando as configurações da interface em /etc/network/interfaces

auto enp0s3
iface enp0s3 inetdhcp

# Aplicando o comando ifup para levantar a interface (já obtendo o seu endereço
# via DHCP)

[root@curso6:~]#ifup enp0s3

Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/enp0s3/08:00:27:78:e7:99
Sending on   LPF/enp0s3/08:00:27:78:e7:99
Sending on   Socket/fallback
DHCPDISCOVER on enp0s3 to 255.255.255.255 port 67 interval 3 (xid=0xf47e9e72)
DHCPDISCOVER on enp0s3 to 255.255.255.255 port 67 interval 3 (xid=0xf47e9e72)
DHCPREQUEST of 192.168.0.119 on enp0s3 to 255.255.255.255 port 67
(xid=0x729e7ef4)
DHCPOFFER of 192.168.0.119 from 192.168.0.1
DHCPACK of 192.168.0.119 from 192.168.0.1
bound to 192.168.0.119 -- renewal in 3364 seconds.

# Verificando as configurações

[root@curso6:~]# ifconfig enp0s3

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.119 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::a00:27ff:fe78:e799 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:78:e7:99 txqueuelen 1000 (Ethernet)
RX packets 325 bytes 120385 (120.3 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 302 bytes 35220 (35.2 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Já em sistemas RedHat-like, o processo é um pouco diferente. Nesses sistemas, cada interface poderá possuir o seu próprio arquivo de configurações permanentes no diretório **/etc/sysconfig/network-scripts**, respeitando uma nomenclatura.

Por exemplo, o arquivo relacionado à interface "enp0s3" será identificado como "ifcfg-enp0s3"; o da interface "enp0s8" será o "ifcfg-enp0s8" e assim por diante. Observe a seguir exemplos de diretivas que são utilizadas para aplicar configurações estáticas à interface "enp0s3":

```
DEVICE=enp0s3
BOOTPROTO=static
IPADDR=192.168.0.90
NETMASK=255.255.255.0
ONBOOT=yes
GATEWAY=192.168.0.1

# Reiniciando o serviço de rede

[root@curso6:~]# systemctl restart network

# Verificando as configurações

[root@curso6:~]# ip address show enp0s3

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:27:56:9e brdff:ff:ff:ff:ff:ff
inet 192.168.0.90/24 brd 192.168.0.255 scope global noprefixroute enp0s3
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe27:569e/64 scope link
valid_lft forever preferred_lft forever
```

A instrução "BOOTPROTO" é utilizada para definirmos a forma de utilização do endereçamento – neste caso, será estático ("static"). Já a instrução "ONBOOT" permite especificar se esta interface sempre será inicializada no boot do sistema ou quando a rede for reiniciada. Observe a tabela de rotas do kernel:

```
[root@curso6:~]#route -n
```

Tabela de Roteamento IP do Kernel

Destino	Roteador	MáscaraGen.	Opções	Métrica	Ref	Uso	Iface
0.0.0.0	192.168.0.1	0.0.0.0	UG	100	0	0	enp0s3
192.168.0.0	0.0.0.0	255.255.255.0	U	100	0	0	enp0s3

Em relação a configuração do gateway, o arquivo **/etc/sysconfig/network** poderá ser utilizado:

```
# Comentando a instrução GATEWAY de /etc/sysconfig/network-scripts/ifcfg-enp0s3

DEVICE=enp0s3
BOOTPROTO=static
IPADDR=192.168.0.90
NETMASK=255.255.255.0
ONBOOT=yes
#GATEWAY=192.168.0.1

# Visualizando a tabela de rotas do kernel

[root@curso6:~]#route -n
```

Tabela de Roteamento IP do Kernel

Destino	Roteador	MáscaraGen.	Opções	Métrica	Ref	Uso	Iface
192.168.0.0	0.0.0.0	255.255.255.0	U	100	0	0	enp0s3

```
# Editando o arquivo /etc/sysconfig/network e inserindo o seguinte conteúdo
```

```
GATEWAY=192.168.0.1
```

```
# Após salvar o arquivo, vamos reiniciar o serviço de rede
```

```
[root@curso6:~]#systemctlrestart network
```

```
# Visualizando a tabela de rotas do kernel
```

```
[root@curso6:~]#route -n
```

Tabela de Roteamento IP do Kernel

Destino	Roteador	MáscaraGen.	Opções	Métrica	Ref	Uso	Iface
0.0.0.0	192.168.0.1	0.0.0.0	UG	100	0	0	enp0s3
192.168.0.0	0.0.0.0	255.255.255.0	U	100	0	0	enp0s3

Para a configuração permanente de outras rotas relacionadas a esta interface, utilizamos agora outro padrão de arquivo: "route-enp0s3", também podendo ser criado em **/etc/sysconfig/network-scripts**.

Se desejarmos configurar uma rota permanente à rede 172.16.30.0/24, após criarmos o arquivo em questão, podemos inserir o conteúdo a seguir. Acompanhe:

```
# Derrubando a interface enp0s3
```

```
[root@curso6:~]#ifdown enp0s3
```

```
# Editando o arquivo /etc/sysconfig/network-scripts/route-enp0s3
```

```
172.16.30.0/24 via 192.168.0.44 dev enp0s3
```

```
# Iniciando a interface enp0s3
```

```
[root@curso6:~]#ifup enp0s3
```

```
# Visualizando as rotas
```

```
[root@curso6:~]#route -n
```

Tabela de Roteamento IP do Kernel

Destino	Roteador	MáscaraGen.	Opções	Métrica	Ref	Uso	Iface
0.0.0.0	192.168.0.1	0.0.0.0	UG	102	0	0	enp0s3
172.16.30.0	192.168.0.44	255.255.255.0	UG	102	0	0	enp0s3
192.168.0.0	0.0.0.0	255.255.255.0	U	102	0	0	enp0s3

Apenas como um teste, vamos agora configurar a interface para que esta receba as suas configurações de endereçamento via **DHCP**. Acompanhe:

```
# Derrubando a interface
```

```
[root@curso6:~]#ifdown enp0s3
```

```
Dispositivo "enp0s3" desconectado com sucesso.
```

```
# Alterando o arquivo /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

```
DEVICE=enp0s3
ONBOOT=yes
BOOTPROTO=dhcp
```

```
# Levantando a interface
```

```
[root@curso6:~]# ifup enp0s3
Conexão          ativada      com          sucesso      (caminho      D-Bus      ativo:
/org/freedesktop/NetworkManager/ActiveConnection/3)
```

```
# Verificando as configurações
```

```
[root@curso6:~]# ip address show enp0s3
```

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:27:56:9e brdfff:fff:fff:fff:fff:ff
inet 192.168.0.119/24 brd 192.168.0.255 scope global noprefixroute dynamic enp0s3
valid_lft 7194sec preferred_lft 7194sec
inet6 fe80::a00:27ff:fe27:569e/64 scope link
valid_lft forever preferred_lft forever
```

Note que agora o comando exibe a string "dynamic" (ao contrário de "global"), indicando que este endereço foi atribuído de forma dinâmica.

4.5 Determinação das Configurações de DNS

Muitos dos problemas de indisponibilidade de comunicação em rede são ocasionados por configurações incorretas relacionadas ao serviço de **DNS** (Domain Name Service). Sendo assim, é importante que o administrador esteja bastante atento aos arquivos e comandos relacionados a este assunto, de forma que o processo de troubleshooting seja realizado de forma eficiente.

A primeira coisa a ser destacada sobre isto é saber como um determinado host é conhecido na rede. Para isto, o arquivo **/etc/hostname** (ou **/etc/HOSTNAME**) poderá ser consultado:

```
[root@curso6:~]# cat /etc/hostname
dltec-laptop
```

Veja que o host é identificado como "dltec-laptop". Esta mesma informação poderá ser obtida de formas diferentes. Acompanhe:

```
# Comando hostname
```

```
[root@curso6:~]# hostname
dltec-laptop
```

```
# Conteúdo do arquivo /proc/sys/kernel/hostname
```

```
[root@curso6:~]# cat /proc/sys/kernel/hostname
dltec-laptop
```

```
# Usando o comando o comando sysctl
```

```
[root@curso6:~]# sysctlkernel.hostname
kernel.hostname = dltec-laptop
```

Caso seja necessário alterar este nome, o comando **hostname** poderá ser utilizado:

```
[root@curso6:~]#hostname centos7
```

```
# Verificando a configuração
```

```
[root@curso6:~]#hostname  
centos7
```

Perceba que o comando **hostname** já exibe o novo nome de host que foi aplicado. Entretanto, esta alteração é temporária. Isto significa que, caso o sistema seja reiniciado, o host volta a ser identificado como "dltec-laptop" – já que o arquivo **/etc/hostname** (ou **/etc/HOSTNAME**) não foi alterado:

```
[root@curso6:~]#cat /etc/hostname  
dltec-laptop
```

Sendo assim, se for necessário alterar de forma definitiva, o administrador poderá editar esses arquivos de forma manual. Mas é importante destacarmos que, em sistemas que utilizam o método de inicialização e gestão de serviços **systemd**, o comando **hostnamectl** é disponibilizado para fazer com que esta configuração seja também aplicada ao arquivo **/etc/hostname** (ou **/etc/HOSTNAME**). Para isso, utilizamos a opção "set-hostname":

```
[root@curso6:~]#hostnamectl set-hostname centos7
```

```
[root@curso6:~]#cat /etc/hostname  
centos7
```

E é também importante mencionarmos que em sistemas mais antigos baseados no RedHat, como o CentOS 5, por exemplo, a configuração do hostname é realizada no arquivo **/etc/sysconfig/network**: 0 116

```
[root@curso6:~]# cat /etc/sysconfig/network
```

```
NETWORKING=yes  
NETWORKING_IPV6=yes  
HOSTNAME=centos5
```

Outro arquivo de suma importância é o **/etc/hosts**, responsável por conter os mapeamentos entre nomes e endereços **IP** – permitindo que a resolução desses nomes sejam feitas de forma local (não necessitando assim utilizar um servidor de **DNS** externo). Observe o seu conteúdo:

```
[root@curso6:~]#cat /etc/hosts
```

```
127.0.0.1          dltec-laptop  
::1               localhost
```

Veja que o **IP** de loopback (127.0.0.1) ainda está mapeado para o antigo nome de host que estava configurado nesta máquina. Desta forma, vamos editá-lo:

```
127.0.0.1          centos  
::1               localhost
```


Vamos criar agora um novo mapeamento no arquivo para o host identificado pelo **IP** 192.168.0.15. Neste exemplo, este host encontra-se localizado na sala1 (utilizado pelo usuário Marcos):

```
127.0.0.1      centos
::1           localhost
192.168.0.15   sala1      marcos
```

Veja que criamos o mapeamento ao endereço **IP** utilizando dois nomes de identificação. Sendo assim, podemos (por exemplo) lançar o comando **ping** para testar a conectividade com o host do Marcos utilizando ambas as strings de identificação:

```
[root@curso6:~]# ping -c 1 sala1

PING sala1 (192.168.0.116) 56(84) bytes of data.
64 bytes from sala1 (192.168.0.116): icmp_seq=1 ttl=64 time=1.73 ms

--- sala1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.734/1.734/1.734/0.000 ms

[root@curso6:~]# ping -c 1 marcos

PING sala1 (192.168.0.116) 56(84) bytes of data.
64 bytes from sala1 (192.168.0.116): icmp_seq=1 ttl=64 time=0.345 ms

--- sala1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.345/0.345/0.345/0.000 ms
```

Já o arquivo **/etc/nsswitch.conf** entra em cena para determinar quais arquivos deverão ser consultados para se obter informações diversas – inclusive aquelas relacionadas ao processo de resolução de nomes (além de também estabelecer a ordem em que esta busca se dará). Observe a configuração aplicada à instrução "hosts":

```
[root@curso6:~]#grep hosts /etc/nsswitch.conf

hosts:      files      dns
```

A instrução "files", como é a primeira a estar listada, indica que o processo de resolução de nomes utilizará primeiramente o conteúdo do arquivo **/etc/hosts**. Caso o nome a ser resolvido não conste neste arquivo, será utilizado então o conteúdo do arquivo **/etc/resolv.conf**, indicado pela string "dns".

Este último é responsável por manter registradas as configurações cliente de **DNS** do sistema. Dito isto, o administrador poderá utilizá-lo para gravar os endereços de hosts (servidores) a serem utilizados no processo de resolução de nomes.

Quando lançamos o comando **ping** junto à string "marcos", por exemplo, o processo de resolução de nomes se deu com sucesso – já que este nome encontra-se mapeado no arquivo **/etc/hosts**. Ou seja, **/etc/resolv.conf** não precisou ser consultado. Partindo do princípio que o arquivo **/etc/resolv.conf** esteja vazio (sem nenhuma instrução válida), vamos tentar utilizar o comando **ping** para o domínio "google.com":

```
[root@curso6:~]# ping -c 1 google.com
ping: google.com: Falha temporária na resolução de nome
```

Veja que o sistema não consegue realizar com sucesso o comando **ping**. Como "google.com" não encontra-se em **/etc/hosts**, o arquivo **/etc/resolv.conf** é consultado. Como não existe qualquer configuração válida ainda nele (neste exemplo), o processo de resolução de nomes é finalizado sem êxito.

Apenas como um exemplo, se incluirmos o seguinte mapeamento em **/etc/hosts**, o processo será realizado com sucesso. Acompanhe:

```
# Inclusão da seguinte linha em /etc/hosts
172.217.30.14 google.com

# Verificação da existência da rota padrão (0.0.0.0)

[root@curso6:~]#route -n

Tabela de Roteamento IP do Kernel
Destino      Roteador      MáscaraGen.    Opções Métrica Ref    Uso Iface
0.0.0.0      192.168.0.1    0.0.0.0        UG      100     0        0 enp0s3
192.168.0.0  0.0.0.0        255.255.255.0  U       100     0        0 enp0s3

# Teste de conectividade

[root@curso6:~]# ping -c 1 google.com

PING google.com (172.217.30.14) 56(84) bytes of data.
64 bytes from google.com (172.217.30.14): icmp_seq=1 ttl=52 time=25.6 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 25.658/25.658/25.658/0.000 ms
```

Vamos então remover o mapeamento que acabamos de criar do arquivo **/etc/hosts** e incluirmos as seguintes linhas no arquivo **/etc/resolv.conf**:

Observe o conteúdo do arquivo presente neste sistema utilizado como exemplo:

```
nameserver 208.67.222.222
nameserver 8.8.8.8
nameserver 8.8.4.4
```

As instruções "nameserver" são utilizadas para especificar os endereços de servidores de **DNS** a serem utilizados no procedimento de resolução de nomes. Atualmente é possível configurar até 3 instruções "nameserver" no arquivo.

Outras instruções também poderão estar contidas neste arquivo, como "domain". Esta é responsável por indicar um nome de domínio que deverá ser acrescentado ("append") a um nome de host para ser utilizado no processo de resolução. Observe:

```
[root@curso6:~]#ping -c 1 ftp
ping: ftp: Nome ou serviço desconhecido
```

Note que o processo de resolução de nomes não foi realizado com sucesso – este host ("ftp") não pôde ser reconhecido. Apenas como um exemplo, vamos incluir a instrução "domain" ao arquivo:

```
nameserver 208.67.222.222
nameserver 8.8.8.8
nameserver 8.8.4.4
domain dltec.com.br
```

Em seguida, podemos novamente executar o comando **ping** junto ao hostname "ftp" e verificar que, agora sim, o processo de comunicação foi estabelecido:

```
[root@curso6:~]# ping -c 1 ftp

PING ftp.dltec.com.br (198.57.234.87) 56(84) bytes of data.
64 bytes from 198-57-234-87.unifiedlayer.com (198.57.234.87): icmp_seq=1 ttl=49
time=223 ms

--- ftp.dltec.com.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 223.920/223.920/223.920/0.000 ms
```

Outra instrução muito comum é a "search", onde é possível determinar uma lista de domínios a serem utilizados no processo de resolução de nomes:

```
nameserver 208.67.222.222
nameserver 8.8.8.8
nameserver 8.8.4.4
search google.com dltec.com.br
```

Como o domínio "google.com" é listado primeiro, o hostname "ftp" é mapeado como "ftp.google.com". Como não é encontrado, o próximo domínio ("dltec.com.br") é consultado e o resultado é exibido pelo comando ping:

```
[root@curso6:~]# ping -c 1 ftp

PING ftp.dltec.com.br (198.57.234.87) 56(84) bytes of data.
64 bytes from 198-57-234-87.unifiedlayer.com (198.57.234.87): icmp_seq=1 ttl=49
time=223 ms

--- ftp.dltec.com.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 223.920/223.920/223.920/0.000 ms
```

Diferentes comandos poderão ser utilizados em procedimentos de troubleshooting relacionado ao serviço de **DNS**. Por exemplo, o comando **host** é normalmente utilizado para converter nomes de domínio em endereços **IP** (e vice versa). Por padrão, o comando os servidores configurados no arquivo **/etc/resolv.conf**:

```
# Exibindo detalhes de endereçamento IP do domínio terra.com.br

[root@curso6:~]# host terra.com.br

terra.com.br has address 208.84.244.116
terra.com.br has IPv6 address 2604:600:0:aaaa:208:84:244:116
terra.com.br mail is handled by 10 vip-us-br-mx.terra.com.
```

Veja que o comando também exibe o registro referente ao **MX** (Mail Exchange), utilizado para indicar o servidor de e-mail utilizado pelo domínio.

Caso desejássemos visualizar apenas esta informação, basta utilizar a opção **-t** e, em seguida, informar a string "mx":

```
[root@curso6:~]# host -t mx lipsum.com
terra.com.br mail is handled by 10 vip-us-br-mx.terra.com.
```

Para exibir detalhes do domínio referente a um determinado **IP**, basta informá-lo junto ao comando:

```
[root@curso6:~]# host 208.84.244.116
116.244.84.208.in-addr.arpa domain name pointer www.terra.com.br.
```

Para descobrir os servidores de **DNS** utilizados por um domínio, basta utilizar a opção **-t** e, em seguida, informar a string "ns":

```
[root@curso6:~]# host -t ns terra.com.br
terra.com.br name server d.dns.terra.com.br.
terra.com.br name server c.dns.terra.com.
terra.com.br name server b.dns.terra.com.br.
terra.com.br name server a.dns.terra.com.
```

Para utilizar um servidor de **DNS** diferente daqueles posicionados no arquivo **/etc/resolv.conf**, basta informá-lo junto ao comando:

```
[root@curso6:~]# host terra.com.br 208.67.220.220

Using domain server:
Name: 208.67.220.220
Address: 208.67.220.220#53
Aliases:

terra.com.br has address 208.84.244.116
terra.com.br has IPv6 address 2604:600:0:aaaa:208:84:244:116
terra.com.br mail is handled by 10 vip-us-br-mx.terra.com.
```

O comando **dig** também poderá ser utilizado neste mesmo sentido. Este comando também utiliza os servidores de **DNS** configurados no arquivo **/etc/resolv.conf**:

```
[root@curso6:~]#dig terra.com.br

; <<>>DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> terra.com.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23566
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:;udp: 512
;; QUESTION SECTION:
;terra.com.br.                IN      A

;; ANSWER SECTION:
terra.com.br.                 500     IN      A      208.84.244.116

;; Query time: 16 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
```

```
;; WHEN: Ter Nov 26 18:52:13 -03 2019
;; MSG SIZE rcvd: 57
```

As linhas iniciadas com o caractere ";" representam comentários – não fazendo parte assim da informação propriamente dita. O comando também exibe estatísticas, como o tempo de duração deste procedimento (campo "Query time") e o servidor de **DNS** utilizado para a consulta (campo "SERVER"). Caso não seja desejado que essas estatísticas sejam exibidas, basta utilizar a opção "+nostats":

```
[root@curso6:~]#dig terra.com.br +nostats

; <<>>DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> terra.com.br +nostats
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58854
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;udp: 512
;; QUESTION SECTION:
;terra.com.br.                IN      A

;; ANSWER SECTION:
terra.com.br.                574     IN      A      208.84.244.116
```

Para visualizar apenas o endereço **IP** relacionado ao domínio, utilizamos a opção "+short":

```
[root@curso6:~]# dig terra.com.br +short
208.84.244.116
```

Para visualizar informações relacionadas ao servidor de e-mail utilizado pelo domínio, utilizamos a opção "MX":

```
[root@curso6:~]#dig terra.com.br MX

; <<>>DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> terra.com.br MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1539
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;udp: 512
;; QUESTION SECTION:
;terra.com.br.                IN      MX

;; ANSWER SECTION:
terra.com.br.                247     IN      MX      10 vip-us-br-mx.terra.com.

;; Query time: 15 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Ter Nov 26 18:51:29 -03 2019
;; MSG SIZE rcvd: 79
```

Se quisermos utilizar um servidor de **DNS** em especial, basta especificar o seu **IP** após o caractere "@":

```
[root@curso6:~]#dig terra.com.br @199.85.126.10

; <<>>DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> terra.com.br @199.85.126.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41355
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;udp: 4096
;; QUESTION SECTION:
;terra.com.br.                IN      A

;; ANSWER SECTION:
terra.com.br.                132     IN      A      208.84.244.116

;; Query time: 278 msec
;; SERVER: 199.85.126.10#53(199.85.126.10)
;; WHEN: Ter Nov 26 18:51:11 -03 2019
;; MSG SIZE rcvd: 57
```

Para realizar uma consulta de **DNS** reversa, utilizamos a opção **-x** e, em seguida, informamos o endereço **IP** a ser consultado:

```
[root@curso6:~]# dig -x 208.84.244.116

; <<>>DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <<>> -x 208.84.244.116
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50688
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;udp: 512
;; QUESTION SECTION:
;116.244.84.208.in-addr.arpa.  IN      PTR

;; ANSWER SECTION:
116.244.84.208.in-addr.arpa. 476 IN      PTR      www.terra.com.br.

;; Query time: 31 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Ter Nov 26 18:50:48 -03 2019
;; MSG SIZE rcvd: 86
```

O comando **nslookup** também poderá ser utilizado neste mesmo sentido:

```
[root@curso6:~]# nslookup lipsum.com

Server:      8.8.8.8
Address:     8.8.8.8#53

Non-authoritative answer:
Name:   terra.com.br
Address: 208.84.244.116
Name:   terra.com.br
Address: 2604:600:0:aaaa:208:84:244:116
```

4.6 NetworkManager



Para finalizar esta seção do capítulo, também é importante que você tenha ciência sobre o impacto que poderá ser exercido nas configurações de rede do sistema pelo serviço **NetworkManager**, inicialmente desenvolvido pela RedHat. Trata-se de um programa que realiza todo o processo de detecção e configurações automáticas aplicáveis ao subsistema de redes no Linux.

Já que oferece uma interface gráfica bastante intuitiva, este serviço tem sido muito utilizado em distribuições voltadas ao uso pessoal. Apenas como um exemplo, veja a seguir que o arquivo **/etc/network/interfaces** foi utilizado para aplicar configurações manuais à interface "enp0s3":

```
[root@curso6:~]# cat /etc/network/interfaces

auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet static
address 192.168.0.78
netmask 255.255.255.0
gateway 192.168.0.1
network 192.168.0.0
broadcast 192.168.0.255
up route add -net 172.16.30.0/24 gw 192.168.0.44 dev enp0s3
```

Vamos executar o comando **ip** para visualizar a devida aplicação da configuração estabelecida no arquivo:

```
[root@curso6:~]# ip address show

1: lo: <LOOPBACK,UP,LOWER_UP>mtu 65536 qdiscnoqueue state UNKNOWN group default qlen 1
link/loopback 00:00:00:00:00:00brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP>mtu 1500 qdiscpfifo_fast state UP
group default qlen 1000
link/ether 08:00:27:5c:ff:ab brd ff:ff:ff:ff:ff:ff
inet 192.168.0.78/24 brd 192.168.0.255 scope global enp0s3
valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe5c:ffab/64 scope link
valid_lft forever preferred_lft forever
```

Para visualizarmos na prática o funcionamento do serviço **NetworkManager**, vamos primeiramente derrubar a interface "enp0s3":

```
[root@curso6:~]# ifdown enp0s3
```

Em seguida, vamos comentar as configurações aplicáveis a interface "enp0s3" no arquivo **/etc/network/interfaces**. Feito isto, lançamos o comando **ip** para removermos quaisquer configurações de endereçamento **IP** atribuídos à interface:

```
[root@curso6:~]# ip address flush enp0s3
```

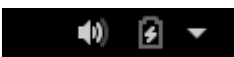
Agora, vamos visualizar o estado corrente do serviço:

```
[root@curso6:~]# systemctl status NetworkManager

• NetworkManager.service - Network Manager
  Loaded: loaded (/lib/systemd/system/NetworkManager.service; disabled; vendor
  Active: inactive (dead)
  Docs: man:NetworkManager(8)
```

```
Nov 22 13:01:09 debian9 systemd[1]: Network Manager is not active.
```

Note que, no momento, o serviço ainda não encontra-se em execução. Se observarmos a barra superior do ambiente **GNOME**, por exemplo, notamos que nenhum ícone associado ao serviço é apresentado:



Desta forma, vamos iniciar o **NetworkManager**:

```
[root@curso6:~]# systemctl start NetworkManager
```

Agora, seu ícone é então posicionado na barra superior do **GNOME**. Como no exemplo estamos utilizando uma interface de rede "cabeada", o ícone é apresentado da seguinte maneira:



Se visualizarmos novamente o estado do serviço, vemos que este já ativou a interface "enp0s3" e, inclusive, já configurou o seu endereço **IP** – obtido via **DHCP**:

```
[root@curso6:~]# systemctl status NetworkManager

NetworkManager.service - Network Manager
  Loaded: loaded (/lib/systemd/system/NetworkManager.service; disabled; vendor preset: enabled)
  Active: active (running) since Fri 2019-11-22 13:36:49 -03; 2min 58s ago
  Docs: man:NetworkManager(8)
  Main PID: 2045 (NetworkManager)
  Tasks: 4 (limit: 4915)
  CGroup: /system.slice/NetworkManager.service
          └─2045 /usr/sbin/NetworkManager --no-daemon
             └─2051 /sbin/dhclient -d -q -sf /usr/lib/NetworkManager/nm-dhcp-helper
               -pf /var/run/dhclient-enp0s3.pid -lf /var/lib

Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.1611] dhcp4
(enp0s3): state changed unknown -> bound
Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.1741] device
(enp0s3): state change: ip-config ->ip-check (r
Nov 22 13:36:50 debian9 dhclient[2051]: bound to 192.168.0.112 -- renewal in 2757
seconds.
Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.2278] device
(enp0s3): state change: ip-check -> secondaries
Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.2586] device
(enp0s3): state change: secondaries -> activated
Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.2612] manager:
NetworkManager state is now CONNECTED_LOCAL
Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.2658] manager:
NetworkManager state is now CONNECTED_GLOBAL
Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.2659] policy:
set 'Wired connection 1' (enp0s3) as default fo
Nov 22 13:36:50 debian9 NetworkManager[2045]: <info> [1574440610.2661] device
(enp0s3): Activation: successful, device activat
Nov 22 13:36:55 debian9 NetworkManager[2045]: <info> [1574440615.1846] manager:
startup complete
```

```
[root@curso6:~]# ifconfig enp0s3

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu 1500
inet192.168.0.112 netmask 255.255.255.0 broadcast 192.168.0.255
inet6 fe80::c637:8ebb:9323:6cec prefixlen 64 scopeid 0x20<link>
ether 08:00:27:5c:ff:ab txqueuelen 1000 (Ethernet)
RX packets 19 bytes 2702 (2.6 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 89 bytes 10189 (9.9 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Ou seja, todo o processo foi realizado de forma automática. Aliás, não somente o endereço **IP** foi configurado, mas também a rota padrão, o servidor de **DNS** etc – o usuário não precisa realizar nenhum procedimento extra. Observe:

```
# Verificação do arquivo /etc/resolv.conf
```

```
[root@curso6:~]# cat /etc/resolv.conf
```

```
# Generated by NetworkManager
```

```
nameserver 192.168.0.1

# Verificação as propriedades do arquivo /etc/resolv.conf

[root@curso6:~]#ls -l /etc/resolv.conf
lrwxrwxrwx 1 root root 35 Nov 22 15:14 /etc/resolv.conf ->
/var/run/NetworkManager/resolv.conf

# Verificação da tabela de roteamento IP do kernel

[root@curso6:~]# route -n

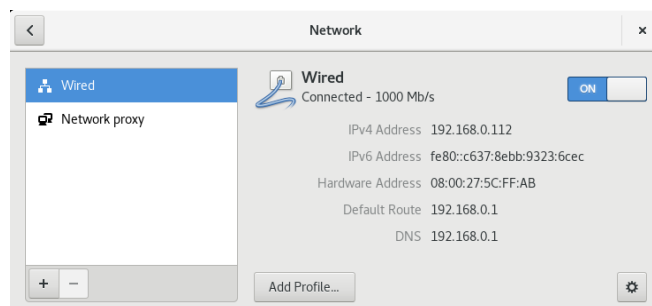
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.0.1    0.0.0.0         UG    100    0      0 enp0s3
192.168.0.0      0.0.0.0        255.255.255.0   U     100    0      0 enp0s3
```

Para fazer com que este serviço seja sempre iniciado no target padrão configurado, basta utilizar o comando **systemctl** junto à opção "enable":

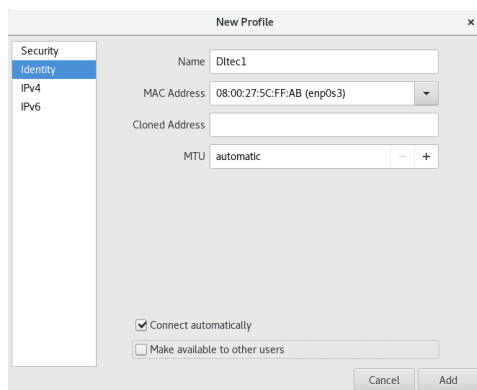
```
[root@curso6:~]# systemctl enable NetworkManager

Created symlink                               /etc/systemd/system/multi-
user.target.wants/NetworkManager.service    →
/lib/systemd/system/NetworkManager.service.
Created symlink /etc/systemd/system/dbus-org.freedesktop.nm-dispatcher.service →
/lib/systemd/system/NetworkManager-dispatcher.service.
```

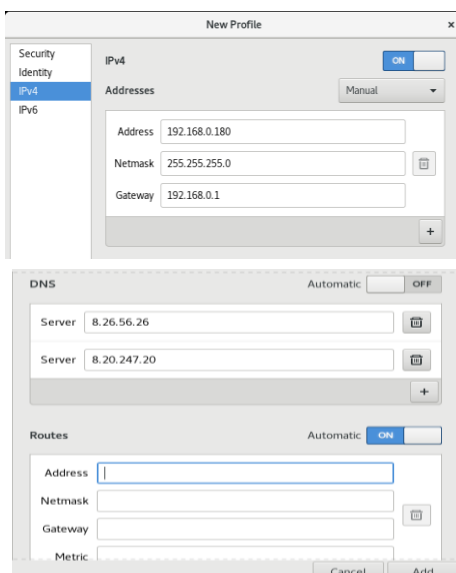
Vamos agora clicar sobre o ícone do serviço no painel do GNOME e, após clicar sobre "WiredConnected", clique também sobre a opção "Wired Settings":



Note que a conexão "Wired" foi automaticamente criada pelo sistema, de forma a conter as configurações atuais aplicáveis à interface "enp0s3". Vamos agora adicionar um novo perfil de conexão, de forma que as configurações nele contidas sejam automaticamente aplicadas à interface quando o sistema for iniciado. Para isto, clicamos no botão "Add Profile..." e, após selecionar a opção "Identity", configuraremos da seguinte maneira:



Em seguida, vamos clicar na opção **IPv4** e deixar configurado da seguinte forma:



Em "Routes" e na opção **IPv6** vamos deixar tudo de forma automática. Para finalizar, vamos clicar no botão "Add".

Neste momento, é criado o arquivo "Dltec1" em **/etc/NetworkManager/system-connections/** contendo todos os detalhes referentes a esta conexão recém-criada:

```
[root@curso6:~]# ls -l /etc/NetworkManager/system-connections
```

```
total 4
-rw----- 1 root root 388 Nov 22 16:44 Dltec1
```

```
[root@curso6:~]# cat /etc/NetworkManager/system-connections
```

```
[connection]
id=Dltec1
uuid=2b37e499-85b3-4948-b158-787781e51343
type=ethernet
```

```
permissions=user:dltec;;

[ethernet]
cloned-mac-address=08:00:27:5C:FF:AB
mac-address=08:00:27:5C:FF:AB
mac-address-blacklist=

[ipv4]
address1=192.168.0.180/24,192.168.0.1
dns=8.26.56.26;8.20.247.20;
dns-search=
ignore-auto-dns=true
method=manual

[ipv6]
addr-gen-mode=stable-privacy
dns-search=
method=auto
```

O **NetworkManager** também poderá ser operado através da linha de comandos. Para isto, utilizamos o comando **nmcli**. Por exemplo, para verificar o estado das interfaces, utilizamos o objeto "dev" seguido pela string "status":

```
[root@curso6:~]# nmcli dev status
```

DEVICE	TYPE	STATE	CONNECTION
enp0s3	ethernet	connected	Dltec1
lo	loopback	unmanaged	--

Se a intenção for exibir informações sobre todas as conexões, executamos:

```
[root@curso6:~]# nmcli connection show
```

NAME	UUID	TYPE	DEVICE
Dltec1	2b37e499-85b3-4948-b158-787781e51343	802-3-ethernet	enp0s3

Para compor os próximos exemplos, vamos disponibilizar mais uma interface de rede ao sistema ("enp0s8") e criar um novo perfil de conexão para ela utilizando o comando **nmcli**. Acompanhe:

```
# Derrubando a nova interface

[root@curso6:~]# ifconfig enp0s8 down

# Limpando as configurações de endereçamento IP

[root@curso6:~]# ip address flush enp0s8

# Verificando o estado das interfaces

[root@curso6:~]# nmcli dev status
```

DEVICE	TYPE	STATE	CONNECTION
enp0s3	ethernet	connected	Dltec1
enp0s8	ethernet	unavailable	--
lo	loopback	unmanaged	--

```
# Criando um novo perfil de conexão denominado "Dltec2" e associando-o à
# interface enp0s8
```

```
[root@curso6:~]# nmcli con add type ethernet con-name dltec2 ifname enp0s8 ip4 192.168.0.77/24 gw4 192.168.0.1 connection.autoconnect yes
```

```
Connection 'dltec2' (bc6c3364-e445-4e8e-9efc-4a5075f46342) successfully added.
```

```
# Verificando a criação do arquivo em /etc/NetworkManager/system-connections/
```

```
[root@curso6:~]# ls -l /etc/NetworkManager/system-connections
```

```
total 8
```

```
-rw----- 1 root root 388 Nov 22 16:44 Dltec1
```

```
-rw----- 1 root root 281 Nov 22 17:18 dltec2
```

```
# Subindo a nova conexão
```

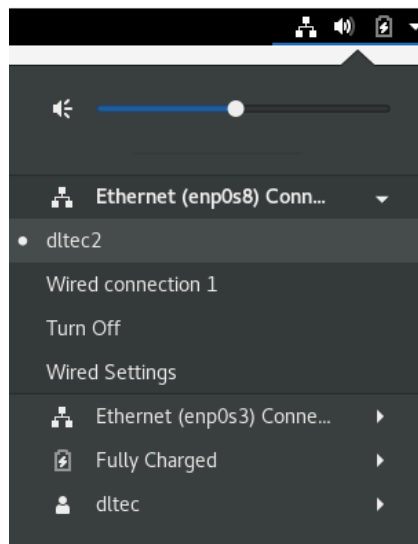
```
[root@curso6:~]# nmcli con up dltec2
```

```
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
```

```
# Verificando o estado das interfaces
```

```
[root@curso6:~]# nmcli dev status
```

DEVICE	TYPE	STATE	CONNECTION
enp0s3	ethernet	connected	Dltec1
enp0s8	ethernet	connected	dltec2
lo	loopback	unmanaged	--



4.7 Investigações em Arquivos de Log e Uso do TCP Wrapper



Os diversos registros de log presentes no sistema também agem como fontes importantes em procedimentos de troubleshooting e problemas relacionados a rede.

O comando **dmesg**, por exemplo, como é responsável por exibir o conteúdo presente no anel de buffer do kernel (ring buffer), é possível identificar mensagens diversas registradas desde o momento da inicialização do sistema.

Dessa forma, caso alguma interface (por exemplo) esteja apresentando algum tipo de problema, registros relacionados poderão ser encontrados neste anel de buffer:

```
[root@curso6:~]# dmesg

[11351.860169] e1000: enp0s3 NIC Link is Down
[11353.876302] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
RX
[11358.659422] 10:16:22.665031 timesyncvgsvcTimeSyncWorker: Radical host time
change: 29 070 815 000 000ns (HostNow=1 573 121 782 665 000 000 ns HostLast=1 573
092 711 850 000 000 ns)
[11368.659892] 10:16:32.665460 timesyncvgsvcTimeSyncWorker: Radical guest time
change: 29 070 814 112 000ns (GuestNow=1 573 121 792 665 433 000 ns GuestLast=1
573 092 721 851 321 000 ns fSetTimeLastLoop=true )
[19145.749852] e1000: enp0s3 NIC Link is Down
[19149.070088] 13:06:57.265031 timesyncvgsvcTimeSyncWorker: Radical host time
change: 2 454 190 000 000ns (HostNow=1 573 132 017 265 000 000 ns HostLast=1 573
129 563 075 000 000 ns)
[19149.779894] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
RX
[19159.071064] 13:07:07.266023 timesyncvgsvcTimeSyncWorker: Radical guest time
change: 2 454 190 305 000ns (GuestNow=1 573 132 027 266 012 000 ns GuestLast=1
573 129 573 075 707 000 ns fSetTimeLastLoop=true )
```

Entretanto, outros arquivos também poderão ser consultados durante o processo de investigação de problemas.

Por exemplo, em distribuições Debian-like, o arquivo **/var/log/syslog** age como uma das principais fontes de registros do sistema (incluindo, claro, questões relacionadas a utilização do subsistema de rede).

Caso este sistema Debian-like utilize o gerenciador de logs **rsyslog**, por exemplo, podemos verificar o seu arquivo **/etc/rsyslog.conf** e perceber quais os registros que vão ser registrados no arquivo **/var/log/syslog**. Observe a configuração padrão:

```
*.*;auth,authpriv.none          -/var/log/syslog
```

Dessa forma, todas as prioridades relacionadas a todas as facilidades (exceto aquelas "auth" e "authpriv") são registradas no arquivo em questão (por isso a sua importância nos processos de troubleshooting).

Outro arquivo essencial é o **/var/log/messages** - também encontrado em distribuições RedHat-like.

Este segue a mesma idéia do anterior, variando o que irá ser registrado de acordo com os critérios administrativos. Observe a configuração padrão presente no arquivo **/etc/rsyslog.conf** disponível no CentOS:

```
*.info;mail.none;authpriv.none;cron.none          /var/log/messages
```

Como podemos visualizar, todas as facilidades (exceto "mail", "authpriv" e "cron") relacionadas às facilidades "info" ou acima (ficando somente "debug" de fora), são registradas em **/var/log/messages**.

É também importante lembrar que o **systemd** também oferece o seu próprio sistema de logs, conhecido como "journal" - manipulado pelo daemon **systemd-journald**. O conteúdo deste sistema poderá ser exibido a partir do comando **journalctl**.

Quando utilizado sem qualquer opção, o comando mostra o conteúdo completo do "journal", iniciando com o registro mais antigo:

```
[root@curso6:~]# journalctl

-- Logs begin at Wed 2019-11-06 18:40:44 -03, end at Thu 2019-11-07 11:46:44 -03.
--
nov 06 18:40:44 dltec-lpic2 kernel: Linux version 4.9.0-11-amd64 (debian-
kernel@lists.debian.org) (gcc version 6
nov 06 18:40:44 dltec-lpic2 kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-
11-amd64 root=UUID=11435ef9-e53
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87
floating point registers'
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE
registers'
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX
registers'
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: xstate_offset[2]: 576,
xstate_sizes[2]: 256
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Enabled xstate features 0x7, context
size is 832 bytes, using 'stan
nov 06 18:40:44 dltec-lpic2 kernel: e820: BIOS-provided physical RAM map:
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x0000000000000000-
0x0000000000009fbff] usable
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x0000000000009fc00-
0x0000000000009ffff] reserved
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x000000000000f0000-
0x000000000000ffffff] reserved
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x0000000000100000-
0x00000000007ffefffff] usable
```

```
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x000000007fff0000-0x000000007fffffff] ACPI data
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
nov 06 18:40:44 dltec-lpic2 kernel: NX (Execute Disable) protection: active
nov 06 18:40:44 dltec-lpic2 kernel: SMBIOS 2.5 present.
nov 06 18:40:44 dltec-lpic2 kernel: DMI: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
nov 06 18:40:44 dltec-lpic2 kernel: Hypervisor detected: KVM
```

Também é possível a utilização de filtros diversos. Por exemplo, para visualizar registros relacionados ao **NetworkManager**, basta utilizarmos a opção **-u** seguida pelo nome do serviço:

```
[root@curso6:~]# journalctl -u NetworkManager
```

```
nov 06 18:40:52 dltec-lpic2 systemd[1]: Network Manager is not active.
nov 07 11:49:36 lab2.estudoslpic2.com.br systemd[1]: Starting Network Manager...
nov 07 11:49:37 lab2.estudoslpic2.com.br NetworkManager[2420]: <info> [1573138177.2013] NetworkManager (version
nov 07 11:49:37 lab2.estudoslpic2.com.br NetworkManager[2420]: <info> [1573138177.2019] Read config: /etc/Netwo
nov 07 11:49:37 lab2.estudoslpic2.com.br NetworkManager[2420]: <info> [1573138177.2286] manager[0x557fab02b040]
nov 07 11:49:37 lab2.estudoslpic2.com.br NetworkManager[2420]: <info> [1573138177.2287] monitoring ifupdownsta
nov 07 11:49:37 lab2.estudoslpic2.com.br NetworkManager[2420]: <info> [1573138177.2336] dns-mgr[0x557fab020180]
nov 07 11:49:37 lab2.estudoslpic2.com.br systemd[1]: Started Network Manager.
nov 07 11:49:37 lab2.estudoslpic2.com.br NetworkManager[2420]: <error> [1573138177.2600] dispatcher: could not g
nov 07 11:49:37 lab2.estudoslpic2.com.br NetworkManager[2420]: <info> [1573138177.2833] init!
```

Se for desejado visualizar todos os registros de log desde "ontem", por exemplo, podemos utilizar a opção **--since**:

```
[root@curso6:~]# journalctl --since yesterday
```

```
-- Logs begin at Wed 2019-11-06 18:40:44 -03, end at Thu 2019-11-07 11:49:37 -03.
--
nov 06 18:40:44 dltec-lpic2 kernel: Linux version 4.9.0-11-amd64 (debian-kernel@lists.debian.org) (gcc version 6
nov 06 18:40:44 dltec-lpic2 kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-11-amd64 root=UUID=11435ef9-e53
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
nov 06 18:40:44 dltec-lpic2 kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'stan
```



```
nov 06 18:40:44 dltec-lpic2 kernel: e820: BIOS-provided physical RAM map:
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x0000000000000000-0x0000000000009fbff] usable
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x0000000000009fc00-0x0000000000009ffff] reserved
nov 06 18:40:44 dltec-lpic2 kernel: BIOS-e820: [mem 0x000000000000f0000-0x000000000000ffffff] reserved
```

Para finalizarmos, é preciso que o administrador esteja bastante atento sobre quem poderá ou não utilizar os diferentes serviços que poderão ser oferecidos na rede. Apenas como um exemplo, vamos utilizar um host Debian para listar os sockets que estejam em modo LISTEN – prontos para receberem conexões:

```
[root@curso6:~]#netstat -ntpl
```

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      508/sshd
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      747/exim4
tcp6       0      0 :::22                  :::*                     LISTEN      508/sshd
tcp6       0      0 :::22                  :::*                     LISTEN      747/exim4
```

Veja, por exemplo, que um dos serviços oferecidos encontra-se ouvindo à porta 22 – tratando-se do servidor de **SSH**.

Vamos utilizar outro host (CentOS) para realizarmos uma conexão com o Debian em sua porta 22:

```
[root@curso6:~]#ssh dltec@192.168.0.122
```

Voltando ao host Debian, vamos novamente lançar o comando **netstat**, exibindo apenas as conexões já estabelecidas:

```
root@curso6:~# netstat -ntp
```

```
Conexões Internet Ativas (sem os servidores)
Proto Recv-Q Send-Q Endereço Local          Endereço Remoto          Estado      PID/Program name
tcp        0      0 192.168.0.122:22        192.168.0.90:59108      ESTABELECIDA 3850/sshd: dltec [p
```

Note que a conexão foi realizada com sucesso, a partir do host CentOS – representado na coluna "ForeignAddress".

Se, por exemplo, o administrador decidir que este host (ou qualquer outro) não deva utilizar este serviço, ele poderá utilizar os arquivos **/etc/hosts.allow** e **/etc/hosts.deny**, integrantes das configurações da biblioteca **TCP Wrapper** (também conhecido como **tcpd**).

Esses arquivos poderão ser utilizados para representar aqueles hosts que poderão ("allow") ou não ("deny") acessar os serviços que são protegidos pela camada lógica oferecida pela biblioteca.

Para visualizarmos isto na prática, vamos a um exemplo. O serviço de **SSH** já utiliza a camada lógica protetora oferecida pelo **TCP Wrapper**, já que utiliza a biblioteca "libwrap.so.0". Podemos perceber isto ao lançar o comando **ldd**, responsável por exibir as bibliotecas compartilhadas/dinâmicas requeridas pelos programas:

```
# Saída parcial do comando
```

```
[root@curso6:~]# ldd /usr/sbin/sshd | grep libwrap
```

```
libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007fcfd83dc000)
```

No Debian, por exemplo, ambos os arquivos de controle de acesso oferecidos pelo **TCP Wrapper** encontram-se disponíveis, mas sem qualquer configuração válida – apenas contendo comentários sobre a forma de utilizá-los. Dessa forma, como não existem ainda configurações, o padrão é que os serviços sejam liberados para todos.

Observe a seguinte configuração aplicada ao arquivo de recusa (**/etc/hosts.deny**):

```
ALL : ALL
```

Neste momento, nenhum usuário conseguirá acessar o serviço de **SSH** do host Debian. Observe a tentativa de conexão via **SSH** a partir do CentOS:

```
[root@curso6:~]# ssh dltec@192.168.0.122
```

```
ssh_exchange_identification: read: Connection reset by peer
```

Inicialmente, vamos somente liberar acesso ao serviço de **SSH** para outro host presente na rede: um OpenSUSE (192.168.0.78). Se lançarmos o comando de conexão a partir deste, a mesma mensagem de recusa de conexão será exibida. Para isto, no Debian, inserimos a seguinte linha em **/etc/hosts.allow**:

```
sshd : 192.168.0.78
```

Sendo assim, estando novamente no OpenSUSE, ao tentarmos realizar a conexão, o processo irá funcionar normalmente, já que encontra-se explicitamente autorizado no arquivo de permissão:

```
[root@curso6:~]# ssh dltec@192.168.0.122
```

```
The authenticity of host '192.168.0.122 (192.168.0.122)' can't be established.  
ECDSA key fingerprint is SHA256:SB0z8CgAO/RZKcg91c3l5gU+ELYBFe40zZH2cSQ8HE4.  
Are you sure you want to continue connecting (yes/no)? <<yes  
dltec@192.168.0.122's password: << senha do usuário dltec
```

```
Linux lab2.estudoslpic2.com.br 4.9.0-11-amd64 #1 SMP Debian 4.9.189-3+deb9u1  
(2019-09-20) x86_64
```

```
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.
```

```
Last login: Thu Nov 7 17:16:03 2019 from 192.168.0.90
```

Verificando o estabelecimento da conexão no Debian

```
[root@curso6:~]#ssh dltec@192.168.0.122 22
```

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 192.168.0.122:22        192.168.0.78:56666     ESTABLISHED 3967/sshd: dltec [p
```

Apesar do host OpenSUSE conseguir realizar a conexão, como não encontra-se explicitamente listado no arquivo **/etc/hosts.allow**, o host CentOS permanece sem conseguir acessar o serviço.

5 Conclusão e Certificado

5.1 Conclusão e Certificado

Parabéns por ter chegado ao final do curso **CONFIGURAÇÃO DE REDE!**

Tenha certeza de que compreendeu todos os conceitos aqui mostrados.

Não esqueça que você deve ler e assistir tudo para obter o seu certificado de conclusão do curso.

Ficamos por aqui e nos encontramos nos próximos cursos!!!!

