UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
ESPECIALIZAÇÃO EM SOFTWARE LIVRE APLICADO À TELEMÁTICA

ANDRÉ DE MACEDO PORTUGAL LOBATO

# IMPLEMENTAÇÃO DE UM ROTEADOR EM UM COMPUTADOR EXECUTANDO O SISTEMA OPERACIONAL LINUX

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2012

ANDRÉ DE MACEDO PORTUGAL LOBATO

# IMPLEMENTAÇÃO DE UM ROTEADOR EM UM COMPUTADOR EXECUTANDO O SISTEMA OPERACIONAL LINUX

Monografia apresentada como requisito parcial para a obtenção do grau de Especialista em Software Livre Aplicado à Telemática, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná. Área de Concentração: Telecomunicações.

Orientador: Prof. Dr. Kleber K. H. Nabas

CURITIBA

2012

# RESUMO

LOBATO, André de M. P. **Implementação de um roteador em um computador executando o sistema operacional Linux**. 2012. 44 f. Monografia (Especialização em Software Livre Aplicado à Telemática) – Programa de Pós-Graduação em Tecnologia, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

Este trabalho descreve a implementação de um roteador em um computador composto apenas de placa-mãe, processador, memória RAM e interfaces de rede, sem unidades de armazenamento locais, inicializado e configurado através da rede. Este computador executa a distribuição Tiny Core Linux do sistema operacional Linux e o Quagga, um conjunto de programas que implementam protocolos de roteamento. São apresentados o Tiny Core Linux e o Quagga. São descritos os procedimentos de personalização do arquivo de imagem do Tiny Core Linux, para incluir o Quagga e outras extensões necessárias, e de configuração de um servidor, de onde o computador com função de roteador é inicializado e configurado. Também é descrito o teste um computador com função de roteador, em uma rede contendo, além deste computador e o servidor, um *modem* ADSL com roteador integrado e o computador de um usuário. O teste mostra que o Quagga pode se comunicar com outro roteador e que o computador com função de roteador pode possuir pouca memória RAM.

**Palavras-chave:** Roteador. Quagga. Linux. Tiny Core Linux.

## ABSTRACT

LOBATO, André de M. P. **Implementation of a router on a computer running Linux operating system**. 2012. 44 f. Monografia (Especialização em Software Livre Aplicado à Telemática) – Programa de Pós-Graduação em Tecnologia, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

This work describes the implementation of a router on a computer comprised of only a motherboard, a processor, RAM memory and network interfaces, with no local storage units, booted and configured from the network. This computer runs Tiny Core Linux distribution of Linux operating system and Quagga, a software suite that implements routing protocols. Tiny Core Linux and Quagga are presented. The procedures of customizing the image file of Tiny Core Linux, to include Quagga and other necessary extensions, and of configuring a server, where the computer functioning as a router boots and is configured from, are described. The test of a computer functioning as a router, on a network with, beyond this computer and the server, an ADSL modem with an integrated router and a user's computer, is also described. The test shows that Quagga can communicate with another router and that the computer functioning as a router can have little RAM memory.



**Keywords:** Router. Quagga. Linux. Tiny Core Linux.

## LISTA DE FIGURAS

## LISTA DE SIGLAS

| | |
|---|---|
| ADSL | Asymmetric Digital Subscriber Line |
| API | Application Programming Interface |
| BGP | Border Gateway Protocol |
| DHCP | Dynamic Host Configuration Protocol |
| DSA | Digital Signature Algorithm |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| GPL | General Public License |
| IP | Internet Protocol |
| MD5 | Message Digest 5 |
| NFS | Network File System |
| OSPF | Open Shortest Path First |
| RAM | Random-access Memory |
| RIP | Routing Information Protocol |
| SSH | Secure Shell |
| TFTP | Trivial File Transfer Protocol |
| UDP | User Datagram Protocol |

**SUMÁRIO**

# 1 INTRODUÇÃO

O elemento central de uma rede de computadores é o roteador. Um roteador possui interfaces pertencentes a redes distintas, conectando, assim, redes diferentes umas às outras. Sua função principal é encaminhar pacotes de uma rede para a outra através do melhor caminho, conforme as rotas disponíveis em sua tabela de roteamento. (GRAZIANI, 2007).

A tabela de roteamento armazena informações sobre redes diretamente conectadas ao roteador e redes remotas – aquelas que não estão diretamente conectadas ao roteador e só podem ser alcançadas através de outro roteador. Estas redes estão associadas ao próximo salto, que é a interface de saída ou roteador para o qual um pacote deve ser encaminhado em direção ao seu destino final. (GRAZIANI, 2007).

As rotas da tabela de roteamento podem ser estáticas ou dinâmicas. Rotas estáticas são aquelas que foram configuradas manualmente pelo administrador do roteador, enquanto rotas dinâmicas são aquelas que foram configuradas automaticamente, através dos protocolos de roteamento. (GRAZIANI, 2007).

O objetivo deste trabalho é implementar um roteador em um computador composto apenas de placa-mãe, processador, memória de acesso aleatório (RAM) e interfaces de rede, sem quaisquer unidades de armazenamento local, a ser inicializado e configurado através da rede. Para tanto, este computador executará a distribuição Tiny Core Linux do sistema operacional Linux e o Quagga, um conjunto de programas que implementam os protocolos de roteamento Routing Information Protocol (RIP), Open Shortest Path First (OSPF) e Border Gateway Protocol (BGP).

A distribuição Tiny Core Linux foi escolhida em função do seu tamanho reduzido e da possibilidade de ser executada inteiramente a partir da memória RAM, características que permitem que o computador não possua unidades de armazenamento locais e seja executado inteiramente a partir da memória RAM.

Primeiramente, serão apresentados o Tiny Core Linux e o Quagga. Em seguida, serão descritos os procedimentos de criação de um arquivo de imagem personalizado do Tiny Core Linux, contendo o Quagga, e de configuração de um servidor, a partir do qual o computador que funcionará como roteador será inicializado e configurado. Por último, será demonstrada a operação de um computador funcionando como roteador, conectando-o a um *modem* com roteador integrado, que suporta o protocolo RIP.

## 2 O TINY CORE LINUX

O Tiny Core Linux – http://distro.ibiblio.org/tinycorelinux – é uma distribuição do sistema operacional Linux criada por Robert Shingledecker. Ao fazer o *download* do arquivo de imagem do Tiny Core Linux, disponível na página oficial da distribuição, já é possível observar uma das suas características: o tamanho. O arquivo da versão 4.7 da distribuição, utilizada neste trabalho, tem apenas 12.562.432 bytes, menos que 12 MB. Ao analisar o conteúdo do arquivo, observa-se a estrutura ilustrada na figura 1[1].

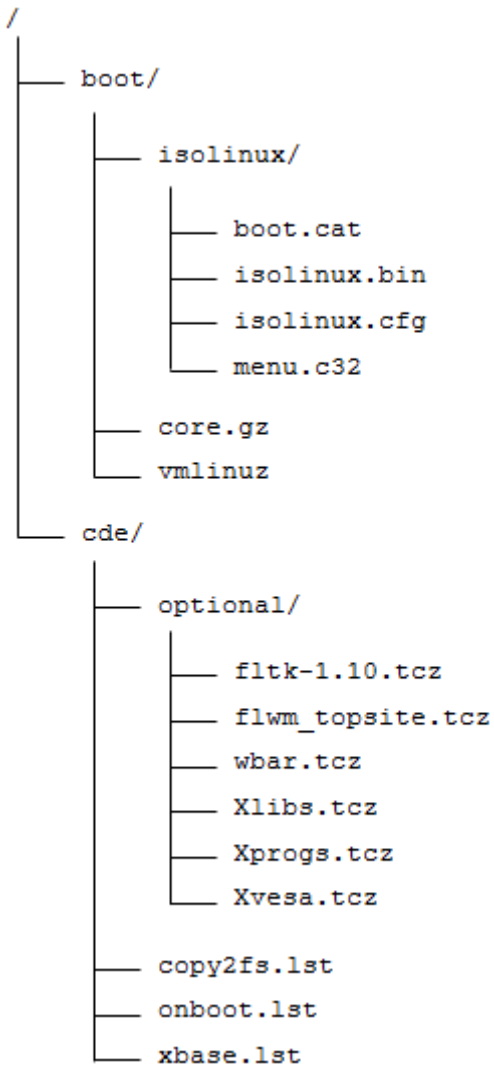


**Figura 1 – Estrutura do arquivo de imagem do Tiny Core Linux.**

[1] Nesta figura, foram omitidos os arquivos com as mensagens exibidas na lista de opções do gerenciador de inicialização ISOLINUX e os arquivos que contém os *hashes* Message Digest 5 (MD5) das extensões do Tiny Core Linux.

A partir desta estrutura, é possível descrever algumas características do Tiny Core Linux:

- a base do sistema operacional encontra-se em dois arquivos: um contendo o *kernel* – arquivo /boot/vmlinuz – e outro contendo o *initial ramdisk*[2] – arquivo /boot/core.gz –, que no Tiny Core Linux é o sistema de arquivos raiz principal;
- as aplicações, denominadas extensões, se encontram fora da base do sistema operacional, na forma de arquivos com a extensão tcz localizados no diretório /cde/optional/ – aquelas que se deseja que sejam carregadas na inicialização do sistema operacional devem ser relacionadas no arquivo /cde/onboot.lst.

Quando o computador é inicializado de uma mídia óptica com o arquivo de imagem do Tiny Core Linux gravado, a lista de opções do gerenciador de inicialização ISOLINUX, ilustrada na figura 2, é exibida. Nela, é possível escolher se o sistema operacional será carregado com sua interface gráfica ou apenas com a interface de linha de comando.

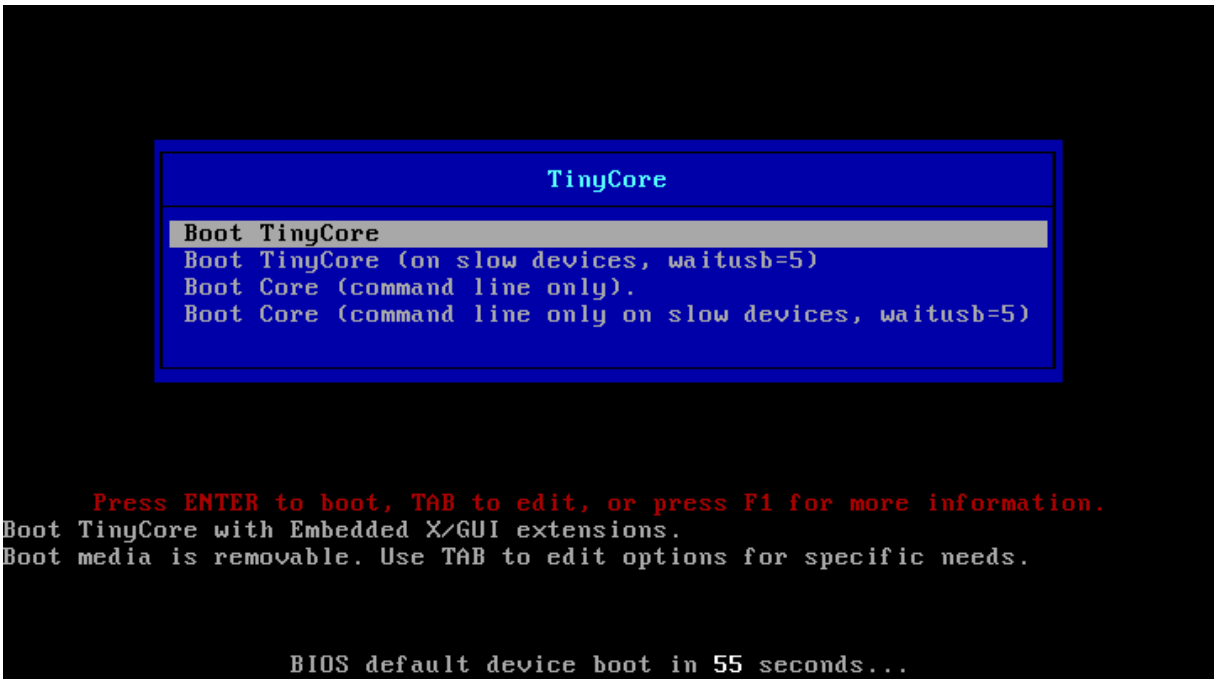


**Figura 2 – Lista de opções do gerenciador de inicialização ISOLINUX.**

[2] O *initial ramdisk* é um sistema de arquivos raiz temporário, carregado na memória RAM e "montado" durante a inicialização do *kernel*, que contém executáveis e módulos (*drivers*) que permitem que o *kernel* localize e "monte" o sistema de arquivos raiz principal, depois do que ele é "desmontado" e removido da memória RAM.

A opção “Boot TinyCore” é a opção padrão desta lista. Ao selecioná-la, além do ISOLINUX extrair o *kernel* para a memória RAM do computador, ele retorna para a linha de comando do *kernel* a localização do *initial ramdisk*, que também é extraído para a memória RAM, e das extensões. No Tiny Core Linux, uma extensão pode:

- ser associada a um dispositivo de *loop*, para o qual é definido um ponto de “montagem” no sistema de arquivos raiz principal, e então ser executada da mídia óptica – modo de operação denominado “Mount Mode”, pois a extensão é “montada” na memória RAM;
- ser copiada para a memória RAM e executada a partir dela, sem a definição de um ponto de “montagem” e sem a necessidade da mídia óptica permanecer no computador – modo de operação denominado “Copy Mode”, pois a extensão é copiada para a memória RAM.

Para que uma extensão seja copiada para a memória RAM do computador e seja executada dela, ela deve ser relacionada no arquivo /cde/copy2fs.lst. Se uma extensão não estiver relacionada neste arquivo, ela será “montada” na memória RAM e será executada da mídia óptica. Para que todas as extensões sejam copiadas para a memória RAM, ao invés de relacioná-las uma a uma no arquivo /cde/copy2fs.lst, pode-se substituir este arquivo pelo arquivo vazio /cde/copy2fs.flg.

O Tiny Core Linux vem acompanhado de seis extensões, responsáveis pela sua interface gráfica. Portanto, a escolha da opção padrão do ISOLINUX leva à área de trabalho do sistema operacional, ilustrada na figura 3. Estas extensões, conforme a figura 1, são as seguintes:

- fltk-1.10.tcz;
- flwm_topside.tcz;
- wbar.tcz;
- Xlibs.tcz;
- Xprogs.tcz;
- Xvesa.tcz.

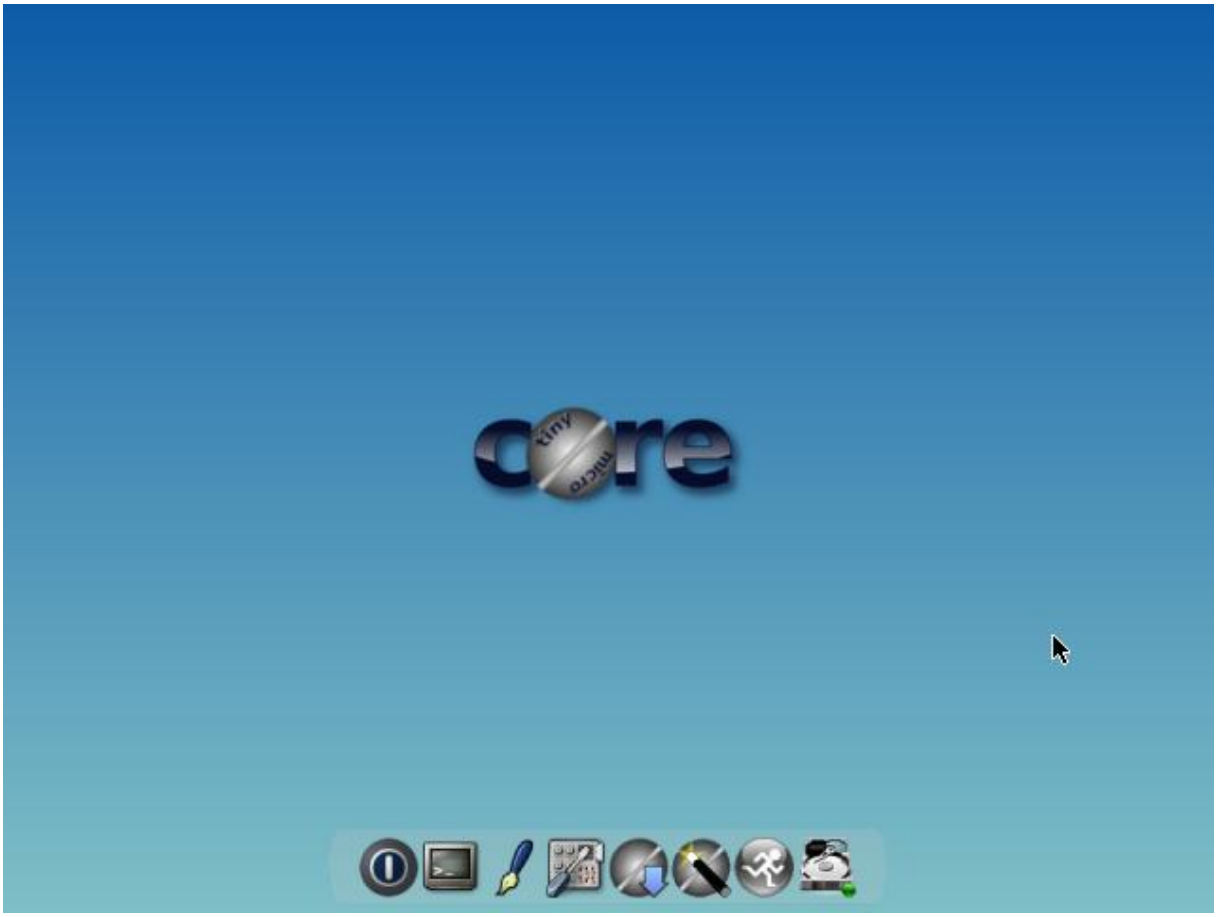

**Figura 3 – Área de trabalho do Tiny Core Linux.**

Todas estas extensões estão relacionadas no arquivo /cde/copy2fs.lst, o que permite que a mídia óptica em que o arquivo de imagem do Tiny Core Linux foi gravado seja removida do computador após a inicialização do sistema operacional. Caso elas não sejam carregadas, o Tiny Core Linux não terá uma interface gráfica, apenas uma interface de linha de comando. Isto pode ser feito selecionando-se a opção "Boot Core (command line only)" na lista de opções ilustrada na figura 2, pois nela o ISOLINUX não retorna para a linha de comando do *kernel* a localização das extensões.

Caso o Tiny Core Linux seja instalado em uma unidade de armazenamento que permite operações de escrita – o que pode ser feito através da extensão tc-install.tcz –, como um disco rígido ou *pen drive*, seu funcionamento será igual ao descrito anteriormente: a partir da memória RAM do computador, com as extensões localizadas fora do sistema de arquivos raiz principal e copiadas ou "montadas" para a memória RAM, conforme especificado pelo usuário, a cada inicialização do sistema operacional. Nesta situação, o diretório cde/ passará a se chamar tce/ e o usuário poderá salvar os seus arquivos nele, ou seja, também fora do sistema de arquivos raiz principal.

Por padrão, os arquivos que o usuário cria ou modifica não estarão disponíveis após a reinicialização do sistema operacional. Para tanto, é necessário salvá-los no arquivo de *backup* do Tiny Core Linux, tce/mydata.tgz, que a cada inicialização do sistema operacional tem seus arquivos extraídos para suas respectivas localizações no sistema de arquivos raiz principal. Para salvar arquivos no arquivo de *backup*, deve-se relacionar os caminhos dos arquivos que se deseja salvar no arquivo /opt/.filetool.lst e executar um comando, que irá salvar os arquivos relacionados no arquivo /opt/.filetool.lst no arquivo de *backup*.

# 3 O QUAGGA

O Quagga é um conjunto de programas, derivado do GNU Zebra e sob a licença GNU General Public License (GPL), que implementa os protocolos de roteamento RIP, RIPv2, RIPng, OSPFv2, OSPFv3 e BGPv4 nos sistemas operacionais Linux, FreeBSD, NetBSD, OpenBSD e Solaris. Um computador que executa o Quagga funciona como um roteador, capaz de trocar informações com outros roteadores através dos protocolos supracitados, para manter a tabela de roteamento do *kernel* do sistema operacional atualizada. Além de manter esta tabela atualizada com rotas que foram aprendidas dinamicamente através dos protocolos, com o Quagga é possível definir rotas estáticas, configurar o endereço Internet Protocol (IP) das interfaces de rede, ligar e desligar as interfaces, entre outros.

O Quagga é composto por seis *daemons*. Um deles, o *daemon* central, gerencia a tabela de roteamento do *kernel* e fornece uma Application Programming Interface (API), chamada Zserv, para os outros *daemons*, clientes do Zserv, que implementam os protocolos de roteamento e transmitem ao *daemon* central as informações sobre o roteamento. Devido esta arquitetura multiprocessos, cada *daemon* possui arquivo de configuração e interface terminal próprios, sendo necessário que cada opção de configuração seja feita no arquivo apropriado. Os *daemons* do Quagga, suas funções, os nomes dos seus arquivos de configuração e os números das portas Transmission Control Protocol (TCP) em que eles aguardam conexões estão ilustrados no quadro 1.

Apesar de cada *daemon* possuir sua própria interface terminal, o Quagga fornece uma interface *shell* integrada, chamada vtysh, que se conecta a cada *daemon*, permitindo que todas as configurações sejam feitas através dela.

| ***Daemon*** | **Função/Protocolo Implementado** | **Arquivo de Configuração** | **Porta (TCP)** |
|---|---|---|---|
| zebra | *Daemon* central | zebra.conf | 2601 |
| ripd | RIPv1 e RIPv2 | ripd.conf | 2602 |
| ripng | RIPng | ripngd.conf | 2603 |
| ospfd | OSPFv2 | ospfd.conf | 2604 |
| bgpd | BGPv4 | bgpd.conf | 2605 |
| ospf6d | OSPFv3 | ospf6d.conf | 2606 |

**Quadro 1 – Função, arquivo de configuração e porta de conexão dos *daemons* do Quagga.**

# 4 PERSONALIZAÇÃO DO ARQUIVO DE IMAGEM DO TINY CORE LINUX

Nesta seção, será descrito o procedimento de criação de um arquivo de imagem personalizado do Tiny Core Linux. Deste arquivo, serão extraídos o *kernel*, o *initial ramdisk* e o diretório /cde/ para um servidor, a partir do qual o computador com função de roteador será inicializado, "montará" seu diretório tce/ e será configurado. Este arquivo de imagem personalizado terá as extensões:

- quagga.tcz: extensão do Quagga;
- openssh.tcz: extensão que possui um servidor Secure Shell (SSH), para que o computador com função de roteador possa ser acessado remotamente através de uma conexão segura;
- isc-dhcp.tcz: extensão que possui um Dynamic Host Configuration Protocol (DHCP) *relay agent*, para que requisições feitas ao servidor DHCP através do endereço IP de *broadcast* 255.255.255.255 sejam encaminhadas para o servidor DHCP em outra rede;
- kmaps.tcz: extensão que possibilita que teclados com *layout* diferente do americano possam ser utilizados.

## 4.1 *Download* das Extensões

O primeiro passo do procedimento de criação do arquivo de imagem personalizado do Tiny Core Linux é a realização do *download* das extensões que serão adicionadas ao sistema operacional e da extensão ezremaster.tcz, que fornece o programa ezremaster, que auxilia o processo de criação de arquivos de imagem do Tiny Core Linux. Para tanto, deve-se abrir o Terminal – segundo ícone da barra de ícones exibida na área de trabalho – e executar o comando

tce-load -wi quagga.tcz openssh.tcz isc-dhcp.tcz kmaps.tcz ezremaster.tcz,

em que a opção -w define que deve ser feito o *download* das extensões e a opção -i define que as extensões devem ser carregadas após o *download* e a cada inicialização do sistema operacional.

## 4.2 Preparação do Sistema Operacional para a Execução dos *Daemons* do Quagga

O segundo passo do procedimento é a preparação do sistema operacional para que os *daemons* do Quagga possam ser executados. Este passo envolve a criação do usuário quagga e dos arquivos de configuração dos *daemons*.

Para criar o usuário *quagga*, deve-se executar o comando

sudo adduser -DH quagga,

em que a opção -D faz com que não seja solicitada a criação de uma senha para o usuário e a opção -H faz com que não seja criado um diretório para o usuário no diretório /home/.

No Tiny Core Linux, a localização padrão dos arquivos de configuração dos *daemons* do Quagga é o diretório /usr/local/etc/. Para criar estes arquivos, deve-se executar o comando

sudo touch /usr/local/etc/zebra.conf /usr/local/etc/ripd.conf /usr/local/etc/ospfd.conf /usr/local/etc/vtysh.conf.

Uma medida interessante é tornar a inicialização dos *daemons* do Quagga automática, a ser realizada durante a inicialização do sistema operacional. Para tanto, deve-se adicionar ao arquivo /opt/bootlocal.sh, que contém comandos a serem executados durante a inicialização do Tiny Core Linux, as linhas:

- /usr/local/sbin/zebra -u root -d;
- /usr/local/sbin/ripd -u root -d;
- /usr/local/sbin/ospfd -u root -d.

Nestes comandos, a opção -u root define com qual usuário os *daemons* devem ser executados e a opção -d define que os arquivos binários devem ser executados como *daemons*.

### 4.3 Preparação do Sistema Operacional para a Execução do Servidor SSH

O terceiro passo do procedimento é a preparação do sistema operacional para que o servidor SSH possa ser executado e possa receber conexões remotas. Este passo envolve a criação de uma senha para o usuário tc, que é o usuário padrão do Tiny Core Linux, e do arquivo de configuração do servidor SSH, /usr/local/etc/ssh/sshd_config.

Para criar a senha para o usuário tc, deve-se executar o comando

passwd.

A extensão openssh.tcz possui um arquivo de configuração de exemplo, /usr/local/etc/ssh/sshd_config.example, com opções de configuração padrão, que será utilizado como o arquivo de configuração necessário. Para tanto, será feita uma cópia deste arquivo, com o nome do arquivo de destino sem a extensão example, através do comando

sudo cp /usr/local/etc/ssh/sshd_config.example /usr/local/etc/ssh/sshd_config.

Após a criação do arquivo de configuração, o servidor SSH será inicializado, para que as chaves de criptografia Digital Signature Algorithm (DSA), Elliptic Curve DSA (ECDSA) e RSA, públicas e privadas, sejam criadas e estejam presentes no arquivo de imagem personalizado do Tiny Core Linux. Para tanto, deve-se executar o comando

sudo /usr/local/etc/init.d/openssh start.

Da mesma maneira que a inicialização dos *daemons* do Quagga foi tornada automática, a ser realizada durante a iniciação do sistema operacional, a inicialização do servidor SSH também será. Para tanto, deve-se adicionar ao arquivo /opt/bootlocal.sh a linha

/usr/local/etc/init.d/openssh start.

### 4.4 Ativação do DHCP *Relay Agent*

Quando um computador com função de roteador for inicializar através da rede, ele tentará localizar um servidor DHCP, para então obter dele um endereço IP, enviando uma mensagem DHCP para o endereço IP de *broadcast* 255.255.255.255. Como requisições encaminhadas para o endereço IP de *broadcast* 255.255.255.255 são limitadas à rede local, caso um computador com função de roteador seja conectado a outro computador com a mesma função em uma rede diferente do servidor DHCP, ele não conseguirá localizar o servidor e obter um endereço IP. Assim, é necessário ativar nos computadores com função de roteador um DHCP *relay agent*, que irá reencaminhar requisições enviadas ao endereço IP de *broadcast* 255.255.255.255, feitas por outros computadores com a mesma função, ao servidor DHCP localizado em outra rede.

Após a inicialização do computador com função de roteador, apenas uma interface de rede terá um endereço IP configurado. Caso o DHCP *relay agent* seja inicializado durante a inicialização do Tiny Core Linux, o reencaminhamento de requisições ao servidor DHCP será ativado apenas nesta interface que possui um endereço IP configurado, e não nas demais. Portanto, a inicialização do DHCP *relay agent* será feita durante a configuração do roteador, através do comando

sudo dhcrelay 192.168.1.254,

em que 192.168.1.254 é o endereço IP do servidor DCHP. Desta maneira, as requisições encaminhadas ao endereço IP de *broadcast* 255.255.255.255 serão reencaminhadas para o servidor DHCP cujo endereço IP é 192.168.1.254.

### 4.5 *Backup* dos Arquivos Criados e Modificados

O quinto passo do procedimento é salvar os arquivos que foram modificados pela criação do usuário quagga, os arquivos de configuração dos *daemons* do Quagga e do servidor SSH e os arquivos que contém as chaves criadas durante a inicialização do servidor SSH no arquivo de *backup* do Tiny Core Linux. Durante a criação do arquivo de imagem

personalizado do Tiny Core Linux, os arquivos salvos no arquivo de *backup* serão integrados ao *initial ramdisk*, o que fará com que, além das extensões desejadas, o arquivo de imagem personalizado contenha os arquivos necessários para a execução das aplicações fornecidas por estas extensões.

Por padrão, o arquivo de *backup*, mydata.tgz, é gravado no diretório tce/, que localiza-se no diretório /tmp/ quando o Tiny Core Linux é inicializado a partir de uma mídia óptica. Como uma mídia óptica não permite operações de escrita, apenas de leitura, não é possível gravar o arquivo de *backup* nela, sendo necessário redefinir a localização do diretório tce/ para uma unidade de armazenamento que permita operações de escrita. Esta unidade será um *pen drive*, que ao ser conectado ao computador deve ter um ponto de "montagem" definido, através do comando

sudo mount /dev/sdb1 /mnt/sdb1.

A localização do diretório tce/ é dada pelo *link* simbólico /etc/sysconfig/tcedir, que primeiramente deve ser removido, através do comando

sudo rm /etc/sysconfig/tcedir,

e então ser recriado tendo como alvo o *pen drive*, através do comando

sudo ln -s /mnt/sdb1 /etc/sysconfig/tcedir,

que faz com que o diretório tce/ passe a ser a raiz do *pen drive*.

Após a redefinição da localização do diretório tce/, os caminhos dos arquivos que devem ser salvos no arquivo de *backup* devem ser adicionados ao arquivo /opt/.filetool.lst. Os caminhos dos arquivos são os seguintes:

- /etc/group;
- /etc/passwd;
- /etc/shadow;
- /usr/local/etc/zebra.conf;
- /usr/local/etc/ripd.conf;

- /usr/local/etc/ospfd.conf;
- /usr/local/etc/vtysh.conf;
- /usr/local/etc/ssh/sshd_config;
- /usr/local/etc/ssh/ssh_host_dsa_key;
- /usr/local/etc/ssh/ssh_host_dsa_key.pub;
- /usr/local/etc/ssh/ssh_host_ecdsa_key;
- /usr/local/etc/ssh/ssh_host_ecdsa_key.pub;
- /usr/local/etc/ssh/ssh_host_rsa_key;
- /usr/local/etc/ssh/ssh_host_rsa_key.pub.

Em seguida, deve-se executar o comando

filetool.sh -b,

que salvará os arquivos relacionados no arquivo /opt/.filetool.lst no arquivo de *backup* /mnt/sdb1/mydata.tgz. Como, por padrão, o diretório /opt/ está relacionado no arquivo /opt/filetool.lst, o arquivo /opt/bootlocal.sh, que foi modificado para tornar a inicialização dos *daemons* do Quagga e do servidor SSH automática, também será incluído no arquivo de *backup*.

Como o ezremaster irá procurar por extensões a serem incluídas no arquivo de imagem personalizado do Tiny Core Linux no diretório tce/ e as extensões descarregadas da Internet foram salvas no diretório /tmp/tce/, após a criação do arquivo de *backup* deve-se redefinir a localização do diretório tce/ para o diretório /tmp/. Para tanto, deve-se remover o *link* simbólico que define a localização atual do diretório tce/, através do comando

sudo rm /etc/sysconfig/tcedir,

e recriá-lo tendo como alvo o diretório /tmp/tce/, através do comando

sudo ln -s /tmp/tce /etc/sysconfig/tcedir.

### 4.6 Criação do Arquivo de Imagem Personalizado do Tiny Core Linux

O sexto e último passo do procedimento é a criação do arquivo de imagem personalizado do Tiny Core Linux através do ezremaster, executado através do comando

ezremaster.

Na primeira tela do programa, ilustrada na figura 4, são informadas a localização da mídia óptica do Tiny Core Linux, que servirá de base para a criação do arquivo de imagem personalizado, e o diretório que o programa deverá utilizar durante o processo de criação. Nesta tela, deve-se:

- selecionar a opção “Use Mounted CD”;
- preencher o campo “Path to Tiny Core / CorePlus ISO or mounted CD” com o ponto de “montagem” da mídia óptica do Tiny Core Linux, /mnt/sr0/;
- deixar o campo “Path to temporary work directory” preenchido com o caminho pré-definido, /tmp/ezremaster.

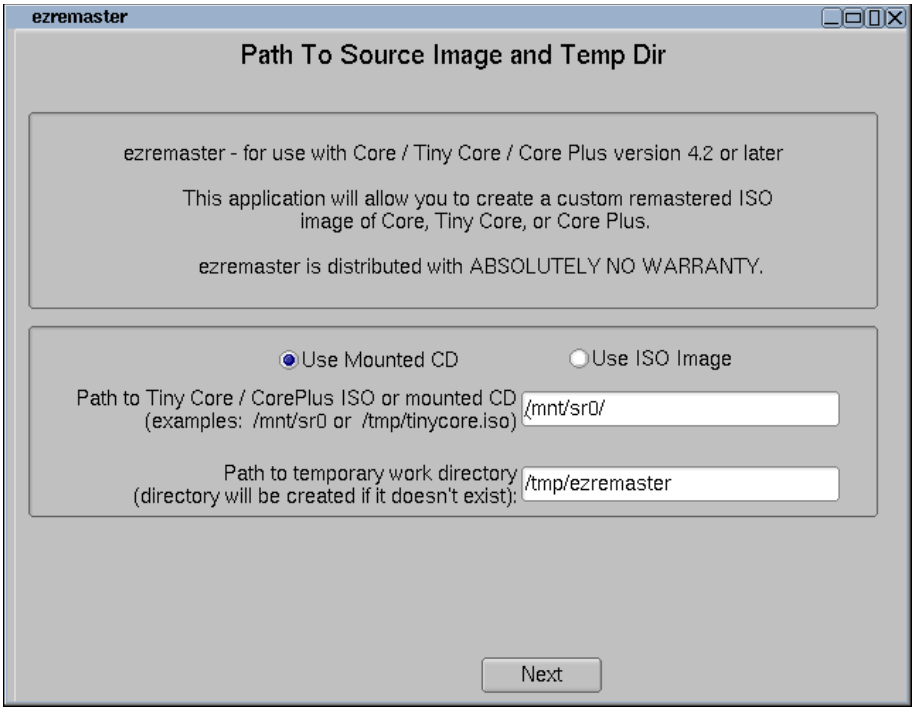


**Figura 4 – Seleção da base do arquivo de imagem personalizado do Tiny Core Linux.**

Na segunda tela do programa, ilustrada na figura 5, são informados os códigos de inicialização que devem ser incluídos no arquivo de configuração do gerenciador de inicialização ISOLINUX e a localização do arquivo de *backup*, cujos arquivos serão extraídos para o *initial ramdisk* contido no arquivo de imagem personalizado. Como o computador com função de roteador será inicializado através da rede, será utilizado o gerenciador de inicialização PXELINUX, que, assim como seu arquivo de configuração, estará presente no servidor, não sendo necessário informar nenhum código de inicialização nesta tela. Nela, deve-se apenas preencher o campo "Path to mydata.tgz to include in ISO" com o caminho do arquivo de *backup* criado anteriormente, /mnt/sdb1/mydata.tgz.

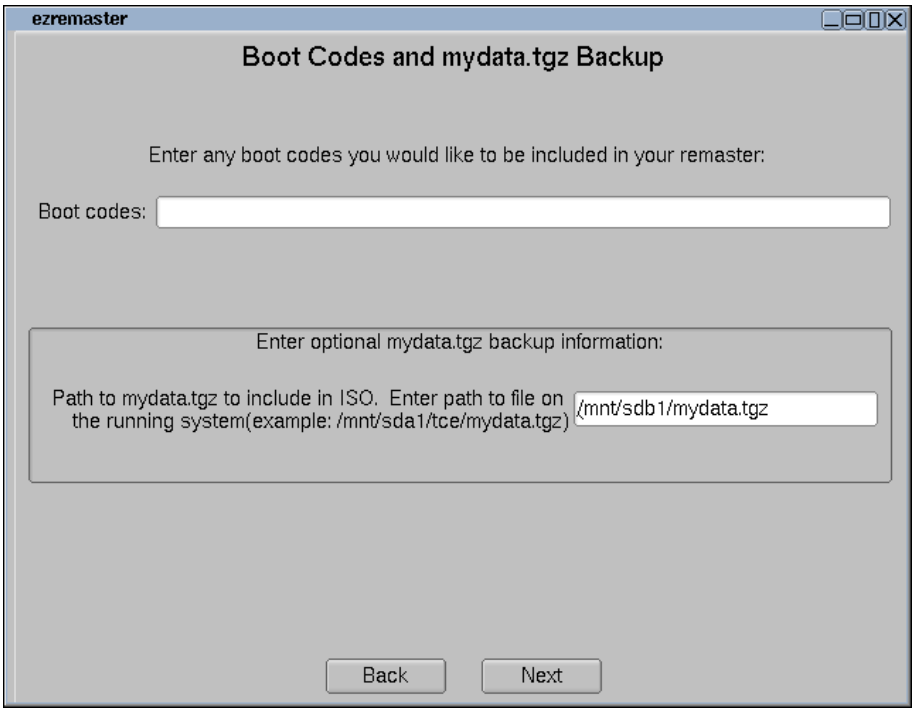


**Figura 5 – Seleção do arquivo de *backup* do arquivo de imagem personalizado.**

Na terceira tela do programa, ilustrada na figura 6, são informadas as extensões que deverão ser incluídas no arquivo de imagem personalizado. Nesta tela, deve-se:

- no campo "Outside initrd apps on boot", clicar no botão "Load" e remover a extensão ezremaster.tcz, deixando apenas as extensões quagga.tcz, openssh.tcz, isc-dhcp.tcz e kmaps.tcz;
- marcar a opção "Check to optionally set copy2fs.flg if using outside initrd apps", para que seja criado o arquivo /cde/copy2fs.flg e todas as extensões presentes no

arquivo de imagem personalizado sejam copiadas para a memória RAM do computador durante a iniciação do Tiny Core Linux.

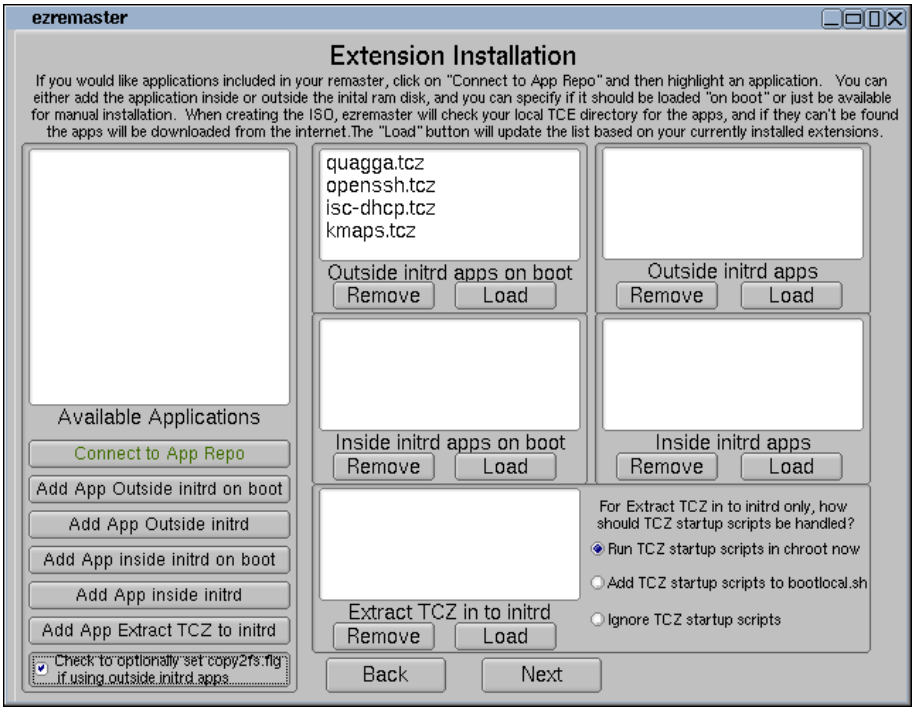


**Figura 6 – Seleção das extensões do arquivo de imagem personalizado.**

Na quarta e na quinta telas, ilustradas nas figuras 7 e 8, nenhuma ação é necessária.

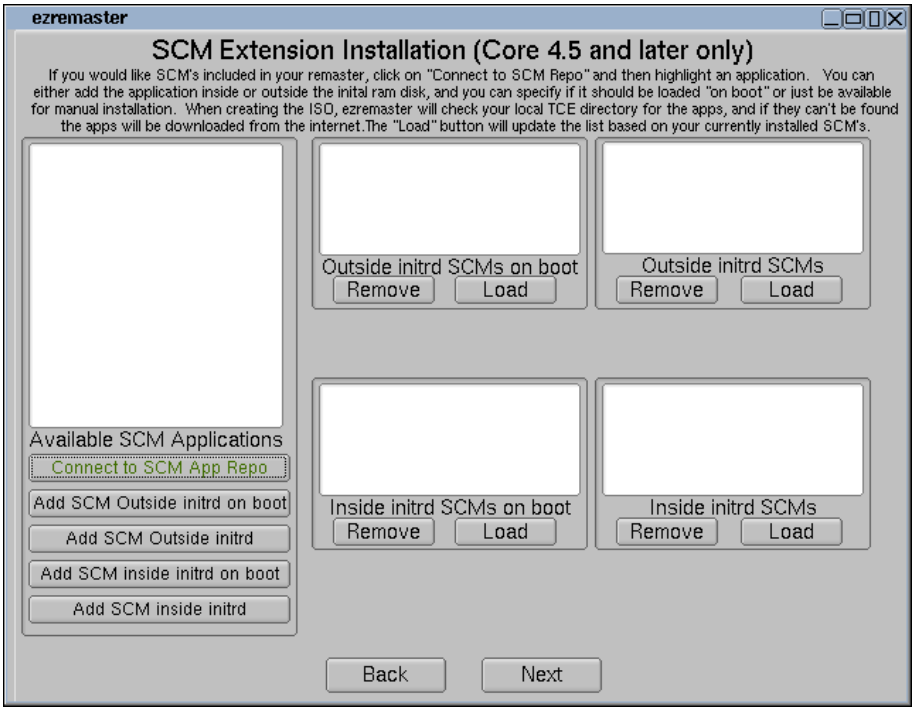


**Figura 7 – Quarta tela do ezremaster, onde nenhuma ação é necessária.**

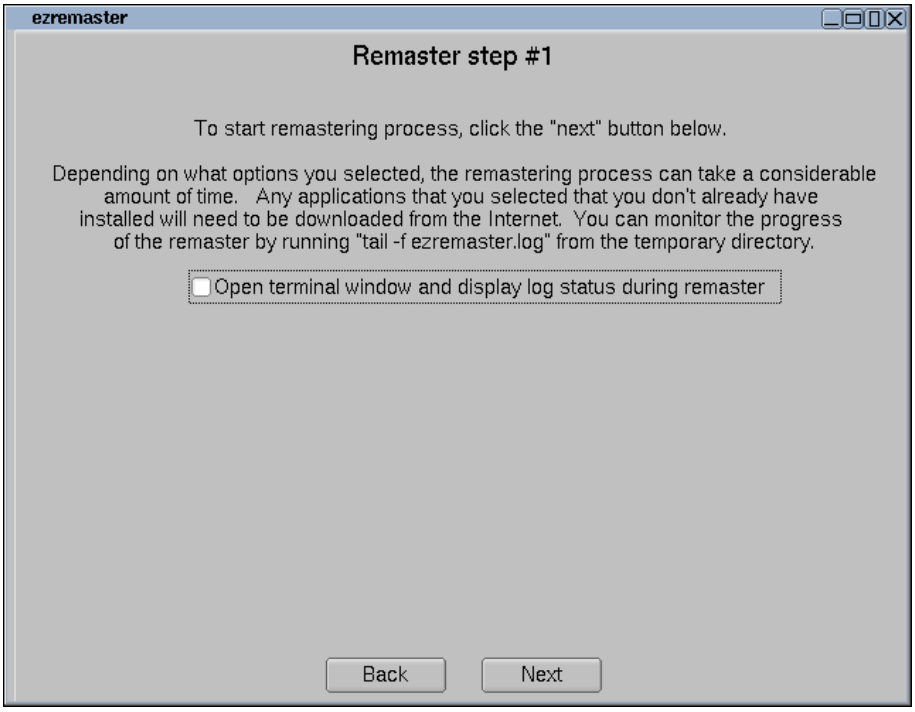


**Figura 8 – Quinta tela do ezremaster, onde nenhuma ação é necessária.**

Na sexta e última tela do programa, ilustrada na figura 9, o arquivo de imagem personalizado do Tiny Core Linux é criado. Porém, antes de criá-lo devem ser feitas alterações no diretório /cde/ e no sistema de arquivos raiz principal, o *initial ramdisk*.

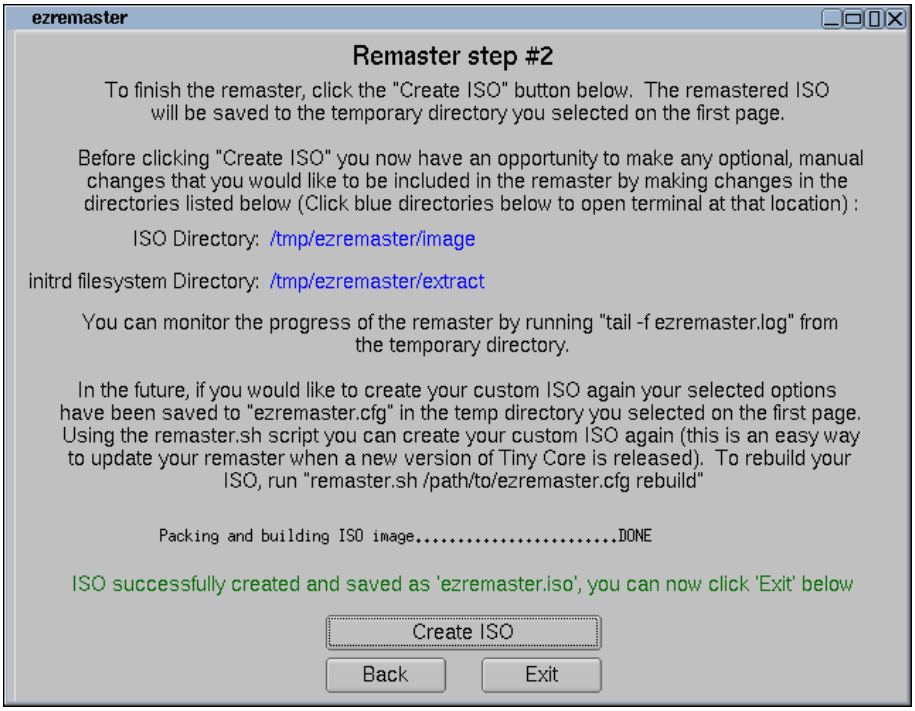


**Figura 9 – Criação do arquivo de imagem personalizado.**

Para alterar o diretório /cde/, na tela ilustrada na figura 9 deve-se clicar em /tmp/ezremaster/image, o que abrirá um novo terminal. Devido a criação do arquivo /cde/copy2fs.flg, para que todas as extensões sejam copiadas para a memória RAM do computador, o arquivo /cde/copy2fs.lst, que relaciona uma a uma as extensões que devem ser copiadas para a memória RAM, pode ser excluído, através do comando

sudo rm cde/copy2fs.lst.

Como o acesso aos computadores com função de roteador será feito através do protocolo SSH, utilizando apenas uma interface de linha de comando, as extensões fltk-1.10.tcz, flwm_topside.tcz, wbar.tcz, Xlibs.tcz, Xprogs.tcz e Xvesa.tcz, responsáveis pela interface gráfica do Tiny Core Linux, podem ser excluídas, juntamente com os arquivos que possuem seus *hashes* MD5. Para tanto, deve-se executar o comando

sudo rm cde/optional/f* cde/optional/w* cde/optional/X*.

Estas extensões estão relacionadas no arquivo /cde/onboot.lst, para que sejam carregadas durante a inicialização do Tiny Core Linux. Como elas foram removidas, este arquivo deve ser editado para ter estas extensões removidas de sua relação.

Por fim, o arquivo /cde/xbase.lst, que relaciona as extensões responsáveis pela interface gráfica, também pode ser excluído, através do comando

sudo rm cde/xbase.lst.

Para alterar o *initial ramdisk*, na tela ilustrada na figura 9 deve-se clicar em /tmp/ezremaster/extract, o que também abrirá um novo terminal. Como os arquivos que foram modificados pela criação do usuário *quagga*, o arquivo de configuração do servidor SSH e os arquivos que contém as chaves utilizadas por este servidor não serão modificados durante a configuração do roteador, eles podem ser removidos do arquivo /opt/.filetool.lst, onde ficarão relacionados apenas os arquivos de configuração do Quagga, que serão modificados durante a configuração do roteador. Assim, quando o comando para a realização do *backup* dos arquivos for executado, será feito o *backup* apenas dos arquivos de configuração do Quagga.

Após estas alterações, deve-se clicar no botão “Create ISO”, o que criará o arquivo de imagem personalizado do Tiny Core Linux, /tmp/ezremaster/ezremaster.iso.

Concluída a criação do arquivo de imagem personalizado do Tiny Core Linux, ele deve ser copiado para o *pen drive*, para que possa ser transferido para o servidor de onde o computador com função de roteador será inicializado e configurado. Para tanto, deve-se executar o comando

sudo cp /tmp/ezremaster/ezremaster.iso /mnt/sdb1/.

Em seguida, o ponto de "montagem" do *pen drive* pode ser desfeito, através do comando

sudo umount /dev/sdb1,

e o dispositivo pode ser removido do computador.

# 5 CONFIGURAÇÃO DO SERVIDOR

O servidor, que executará a distribuição Debian[3] do sistema operacional Linux, irá hospedar o *kernel* e o *initial ramdisk* do Tiny Core Linux, para que o computador com função de roteador possa inicializar através da rede, e também o diretório tce/, que será "montado" pelo computador com função de roteador. O servidor também será utilizado para acessar o roteador remotamente, através de uma conexão segura, para que o roteador possa ser configurado. Para tanto, é necessário que sejam instalados no servidor os pacotes seguintes:

- isc-dhcp-server: pacote que fornece um servidor DHCP, para a atribuição de um endereço IP ao computador com função de roteador;
- tftpd-hpa: pacote que fornece um servidor Trivial File Transfer Protocol (TFTP), para que o *kernel* e o *initial ramdisk* do Tiny Core Linux sejam transferidos para o computador com função de roteador;
- nfs-kernel-server: pacote que fornece um servidor Network File System (NFS), para que o diretório tce/ do Tiny Core Linux seja disponibilizado no servidor como um sistema de arquivos de rede e possa ser acessado pelo computador com função de roteador através da rede, como se fosse um diretório local;
- syslinux: pacote que fornece, entre outros, o gerenciador de inicialização PXELINUX, para o gerenciamento da inicialização dos computadores com função de roteador;

## 5.1 Instalação dos Pacotes

O primeiro passo da configuração do servidor é a instalação dos pacotes necessários. Para tanto, primeiramente é necessário editar o arquivo /etc/apt/sources.list, que contém a lista de fontes de pacotes Debian, para que contenha as fontes:

- deb http://ftp.br.debian.org/debian/ squeeze main;

[3] Foi utilizada a versão 6.0.5, de 32 bits, do Debian.

- deb-src http://ftp.br.debian.org/debian/ squeeze main;
- deb http://ftp.br.debian.org/debian-security/ squeeze/updates main;
- deb-src http://ftp.br.debian.org/debian-security/ squeeze/updates main.

Em seguida, deve-se executar os comandos

apt-get update,

para sincronizar os arquivos de índice de pacotes a partir das fontes relacionadas no arquivo /etc/apt/sources.list, e

apt-get install isc-dhcp-server tftpd-hpa nfs-kernel-server syslinux,

para instalar os pacotes necessários.

Durante a instalação do pacote tftpd-hpa, o usuário é solicitado a informar o diretório que será utilizado pelo servidor TFTP, que deve ser o diretório /tftpboot/, o que criará este diretório.

## 5.2. Configuração do Endereço IP do Servidor

O segundo passo da configuração do servidor é a configuração do seu endereço IP, que será 192.168.1.254 com máscara de sub-rede 255.255.255.0, ou seja, ele pertencerá à rede 192.168.1.0/24. Para alterar o endereço IP do servidor, deve-se executar o comando

ifconfig eth0 192.168.1.254 netmask 255.255.255.0.

Cada vez que o servidor for inicializado, o sistema operacional tentará obter um endereço IP por DHCP. Para que isto não seja feito e o servidor tenha um endereço IP fixo, é necessário editar o arquivo /etc/network/interfaces, que contém as configurações das interfaces de rede do servidor. Neste arquivo, nas configurações referentes à interface eth0, deve-se, conforme ilustrado na figura 10:

- substituir a opção allow-hotplug pela opção auto;
- substituir a opção dhcp pela opção static;
- adicionar opções que definem o endereço IP, a máscara de sub-rede e o endereço do *gateway*.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
            address 192.168.1.254
            netmask 255.255.255.0
            gatewak 192.168.1.1
```

**Figura 10 – Arquivo /etc/network/interfaces, com as configurações da interface de rede.**

## 5.3 Configuração do Servidor DHCP

O terceiro passo da configuração do servidor é a configuração do servidor DHCP, feita no arquivo /etc/dhcp/dhcpd.conf. As opções de configuração deste arquivo, ilustrado na figura 11, são as seguintes:

- subnet 192.168.1.0 netmask 255.255.255.0 { }: provê informações ao servidor DHCP que permitem que ele determine se um endereço IP pertence ou não à sub-rede declarada. Dentro deste campo, entre as chaves, são informados a faixa de endereços IP da sub-rede declarada que podem ser concedidos a clientes e outros parâmetros específicos desta sub-rede;
- range 192.168.1.2 192.168.1.10: define os endereços IP inferior e superior da faixa de endereços, pertencente à sub-rede declarada, que podem ser concedidos a clientes do servidor DHCP;
- filename “pxelinux.0”: define o nome do arquivo de inicialização que o cliente do servidor DHCP, que inicializará através da rede, deverá carregar;
- next-server 192.168.1.254: define o endereço do servidor de onde o arquivo de inicialização deve ser carregado.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
            range 192.168.1.2 192.168.1.10;
            filename "pxelinux.0";
            next-server 192.168.1.254;
}
```

**Figura 11 – Arquivo /etc/dhcp/dhcpd.conf, com as configurações do servidor DHCP.**

Após a configuração do servidor DHCP, ele pode ser inicializado, através do comando

/etc/init.d/isc-dhcp-server start.

## 5.4 Configuração do Servidor TFTP

O quarto passo da configuração do servidor é a configuração do servidor TFTP, feita no arquivo /etc/default/tftpd-hpa. As opções de configuração deste arquivo, ilustrado na figura 12, são as seguintes:

- TFTP_USERNAME="tftp": define o nome do usuário com o qual o servidor TFTP será executado;
- TFTP_DIRECTORY="/tftpboot": define o diretório do servidor TFTP;
- TFTP_ADDRESS="192.168.1.254:69": define o endereço IP e a porta User Datagram Protocol (UDP) em que o servidor TFTP aguardará conexões;
- TFTP_OPTIONS="--secure": permite que o cliente do servidor TFTP informe o caminho relativo do arquivo de inicialização a ser obtido do servidor, em relação ao diretório do servidor; assim, o cliente pode informar apenas o caminho relativo pxelinux.0, ao invés do caminho absoluto /tftpboot/pxelinux.0.

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/tftpboot"
TFTP_ADDRESS="192.168.1.254:69"
TFTP_OPTIONS="--secure"
```

**Figura 12 – Arquivo /etc/default/tftpd-hpa, com as configurações do servidor TFTP.**

Após a configuração do servidor TFTP, ele pode ser inicializado, através do comando

/etc/init.d/tftpd-hpa start.

No servidor TFTP, serão hospedados o *kernel* e o *initial ramdisk* do Tiny Core Linux, que encontram-se no arquivo de imagem personalizado criado anteriormente, localizado no *pen drive*. Ao conectar o *pen drive* ao computador, deve-se definir um ponto de "montagem" para ele. Para tanto, é necessário criar o diretório que servirá como ponto de "montagem", através do comando

mkdir /media/pendrive/,

e então definir o ponto de "montagem", através do comando

mount /dev/sdb1 /media/pendrive/.

Para que os arquivos do *kernel* e do *initial ramdisk* sejam obtidos do arquivo de imagem, também é necessário definir um ponto de "montagem" para ele. Para tanto, deve-se criar o diretório que será o ponto de "montagem", através do comando

mkdir /media/iso/,

e em seguida definir o ponto de "montagem", através do comando

mount -o loop /media/pendrive/ezremaster.iso /media/iso/.

Após a definição destes pontos de "montagem", o *kernel* e o *initial ramdisk* do Tiny Core Linux, bem como suas extensões, podem ser copiados para o diretório do servidor TFTP. Estes arquivos ficarão no diretório /tftpboot/tcl/, que deve ser criado através do comando

mkdir /tftpboot/tcl/.

Em seguida, os arquivos podem ser copiados para o diretório, através do comando

cp -r /media/iso/* /tftpboot/tcl/.

Como a inicialização será gerenciada pelo PXELINUX, o gerenciador de inicialização SYSLINUX, presente no arquivo de imagem personalizado, pode ser removido do servidor TFTP, através do comando

rm -r /tftpboot/tcl/boot/isolinux/.

## 5.5 Configuração do Diretório tce/

Como as extensões do Tiny Core Linux localizam-se fora do sistema de arquivos raiz principal, no diretório tce/, é necessário que o sistema operacional, em execução no computador com função de roteador, possa acessar este diretório, para que as extensões possam ser carregadas e os arquivos de configuração do Quagga possam ser salvos no arquivo de *backup* tce/mydata.tgz. Para tanto, o diretório /tftpboot/tcl/, onde encontra-se o diretório tce/, deve ser disponibilizado como um sistema de arquivos de rede, para que o Tiny Core Linux possa definir um ponto de "montagem" para ele e possa acessá-lo através da rede, como se fosse um diretório local.

Primeiramente, é necessário renomear o diretório das extensões do Tiny Core Linux, copiado do arquivo de imagem personalizado, de cde/ para tce/. Para tanto, deve-se executar o comando

mv /tftpboot/tcl/cde/ /tftpboot/tcl/tce/.

Em seguida, deve-se editar o arquivo /etc/exports, que possui os diretórios a serem disponibilizados como sistemas de arquivos de rede, e incluir nele a linha

/tftpboot/tcl *(rw,no_root_squash),

o que permitirá que qualquer cliente possa “montar” e acessar o diretório /tftpboot/tcl/, com permissão de leitura e escrita.

Finalmente, deve-se executar o comando

exportfs -a,

que irá incluir os diretórios relacionados no arquivo /etc/exports na lista de sistema de arquivos de rede, que podem ser “montados” e acessados através da rede.

## 5.6 Configuração do Gerenciador de Inicialização PXELINUX

Durante a inicialização através da rede, o computador com função de roteador irá procurar pelo gerenciador de inicialização no diretório /tftpboot/. Portanto, deve-se copiar para este diretório o arquivo de inicialização do PXELINUX, fornecido pelo pacote syslinux, através do comando

cp /usr/lib/syslinux/pxelinux.0 /tftpboot/.

O PXELINUX irá utilizar o arquivo de configuração /tftpboot/pxelinux.cfg/default. Antes de criar este arquivo, é necessário criar o diretório /tftpboot/pxelinux.cfg/, através do comando

mkdir /tftpboot/pxelinux.cfg/,

para então criar o arquivo, através do comando

touch /tftpboot/pxelinux.cfg/default.

O arquivo de configuração /tftpboot/pxelinux.cfg/default, ilustrado na figura 13[4], possui as opções de configuração seguintes:

[4] O parâmetro kmap=qwerty/br-abnt2 deve estar na mesma linha da opção APPEND.

- DEFAULT tcl/boot/vmlinuz: determina o *kernel* que deverá ser carregado pelo gerenciador de inicialização, sendo sua localização especificada em relação ao diretório /tftpboot/;
- APPEND initrd=tcl/boot/core.gz nfsmount=192.168.1.254:/tftpboot/tcl kmap=qwerty/br-abnt2: determina as opções que devem ser retornadas para a linha de comando do *kernel*, sendo que a opção initrd=tcl/boot/core.gz determina a localização do *initial ramdisk*, em relação ao diretório /tftpboot/, a opção nfsmount=192.168.1.254:/tftpboot/tcl determina a localização do diretório tce/, que será "montado" e acessado através da rede, e a opção kmap=qwerty/br-abnt2 determina o *layout* do teclado.

```
DEFAULT tcl/boot/vmlinuz
APPEND initrd=tcl/boot/core.gz nfsmount=192.168.1.254:/tftpboot/tcl
kmap=qwerty/br-abnt2
```

**Figura 13 – Arquivo /tftpboot/pxelinux.cfg/default, com as configurações do PXELINUX.**

# 6 TESTES DO COMPUTADOR COM FUNÇÃO DE ROTEADOR

Para testar o funcionamento de um computador funcionando como roteador, executando o Tiny Core Linux e o Quagga a partir da memória RAM, após ter inicializado através da rede, e configurado remotamente, foi montada a rede ilustrada na figura 14. Nesta rede, além deste computador funcionando como roteador – que possui 512 MB de memória RAM, dos quais 32 MB são dedicadas ao adaptador de vídeo *onboard* –, foram utilizados um *modem* Asymetric Digital Subscriber Line (ADSL), com roteador com suporte às versões 1 e 2 do protocolo RIP e *switch* de 4 portas integrados, um computador com função de servidor, a partir do qual o computador funcionando como roteador inicializou e foi configurado, e um computador de um usuário desejando acessar a Internet. O objetivo do teste foi configurar o protocolo RIP no computador com função de roteador e no roteador do *modem* ADSL e, após ambos terem trocado informações sobre as rotas para as redes conhecidas, verificar a conectividade do computador do usuário à Internet.

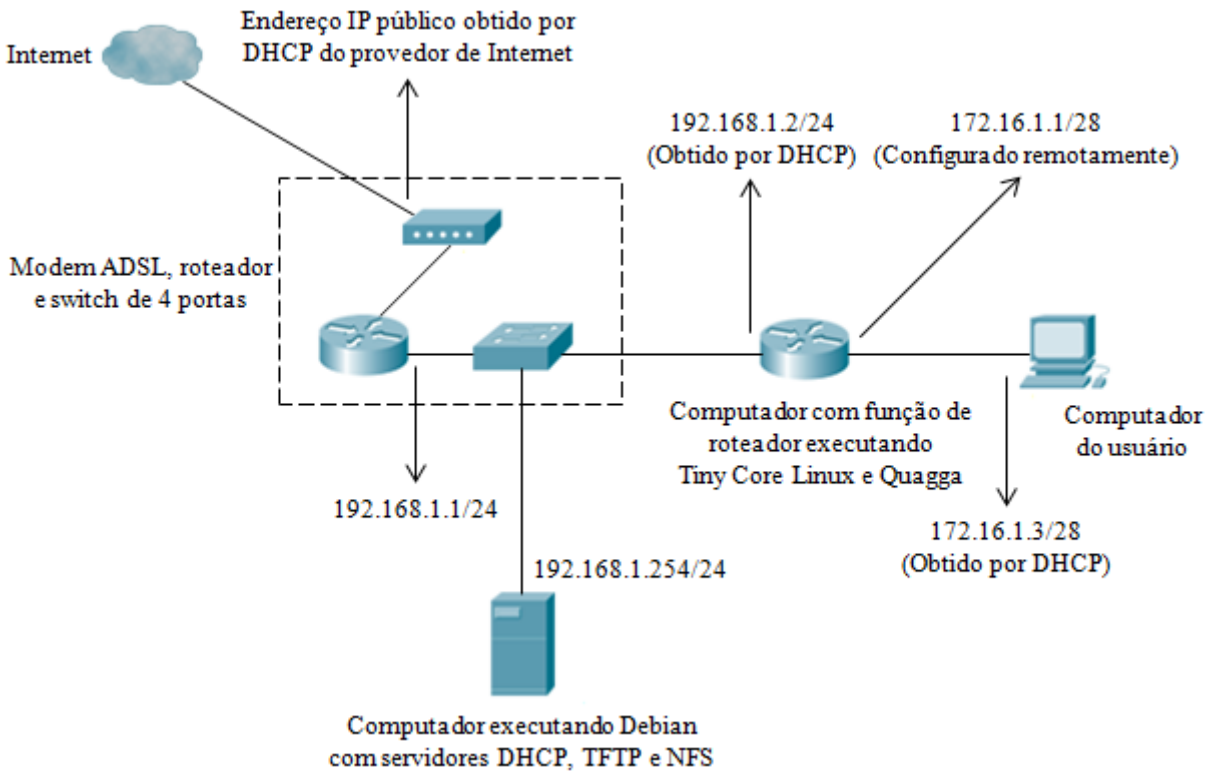


**Figura 14 – Rede utilizada para os testes de um computador com função de roteador.**

## 6.1 Configuração do Servidor

A configuração do computador com função de servidor ocorreu conforme o procedimento descrito no item 4, porém foi adicionado ao arquivo de configuração do servidor DHCP uma nova faixa de endereços IP, pertencente à rede 172.16.1.0/28, para ser atribuída aos computadores que se conectariam ao computador com função de roteador – neste caso, o computador do usuário desejando acessar a Internet – conforme ilustrado na figura 15. Após a configuração, foi verificado, através do comando nmap[5], que os servidores DHCP, TFTP e NFS estavam aguardando conexões nas portas UDP apropriadas, conforme ilustrado na figura 16.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
            range 192.168.1.2 192.168.1.10;
            filename "pxelinux.0";
            next-server 192.168.1.254;
}

subnet 172.16.1.0 netmask 255.255.255.240 {
            range 172.16.1.2 172.16.1.14;
            option routers 172.16.1.1;
}
```

**Figura 15 – Arquivo /etc/dhcp/dhcpd.conf, com as novas configurações do servidor DHCP.**

```
root@debian:~# nmap -sU 192.168.1.254

Starting Nmap 5.00 ( http://nmap.org ) at 2012-11-21 15:22 BRST
Interesting ports on 192.168.1.254:
Not shown: 996 closed ports
PORT     STATE         SERVICE
67/udp   open|filtered dhcps
69/udp   open|filtered tftp
111/udp  open|filtered rpcbind
2049/udp open|filtered nfs

Nmap done: 1 IP address (1 host up) scanned in 1.31 seconds
root@debian:~# _
```

**Figura 16 – Servidores DHCP, TFTP e NFS aguardando conexões em suas portas UDP.**

[5] O nmap é um comando do programa Nmap, que foi obtido através do comando apt-get install nmap.

## 6.2 Inicialização do Computador com Função de Roteador

Após a configuração do servidor, o computador com função de roteador foi ligado, tendo inicializado através da rede com sucesso. Neste processo, ele obteu do servidor DHCP o endereço IP 192.168.1.2, com máscara de sub-rede 255.255.255.0. A figura 17 ilustra que este computador estava com o servidor SSH, para poder ser acessado e configurado remotamente, e com os *daemons* do Quagga ativos e aguardando conexões, conforme relatado pelo comando nmap.

```
root@debian:~# nmap 192.168.1.2

Starting Nmap 5.00 ( http://nmap.org ) at 2012-11-21 15:23 BRST
Interesting ports on 192.168.1.2:
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
2601/tcp open  zebra
2602/tcp open  ripd
2604/tcp open  ospfd
MAC Address: 00:18:F3:A8:00:BD (Asustek Computer)

Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
root@debian:~# _
```

**Figura 17 – Servidor SSH e *daemons* do Quagga no roteador aguardando conexões.**

Em seguida, foi feita uma conexão remota ao computador com função de roteador, através do protocolo SSH, conforme ilustrado na figura 18, e foram obtidos o estado das suas interfaces de rede, a tabela de roteamento do *kernel*, o ponto de "montagem" do diretório tce/ e seu consumo de memória RAM, conforme ilustrado nas figuras 19 a 22, respectivamente. Nestas figuras, pode-se observar que:

- apenas a interface de rede eth1, conectada ao *switch* do *modem* ADSL, possui endereço IP, obtido do servidor DHCP; o endereço IP da outra interface, eth0, conectada ao computador do usuário, deve ser configurado manualmente;
- na tabela de roteamento do *kernel*, além do endereço IP da interface de *loopback*, há apenas a rede 192.168.1.0/24, conectada à interface eth1;
- o diretório /tftpboot/tcl/, onde encontra-se o diretório tce/ e que está localizado no servidor, teve um ponto de "montagem" definido no diretório /mnt/nfs/, podendo ser acessado como se fosse um diretório local;

- o Tiny Core Linux e todas as aplicações em execução, ambos sendo executados da memória RAM do computador, estão consumindo apenas 114992 kB de memória RAM, pouco mais que 112 MB.


**Figura 18 – Conexão ao computador com função de roteador através do protocolo SSH.**

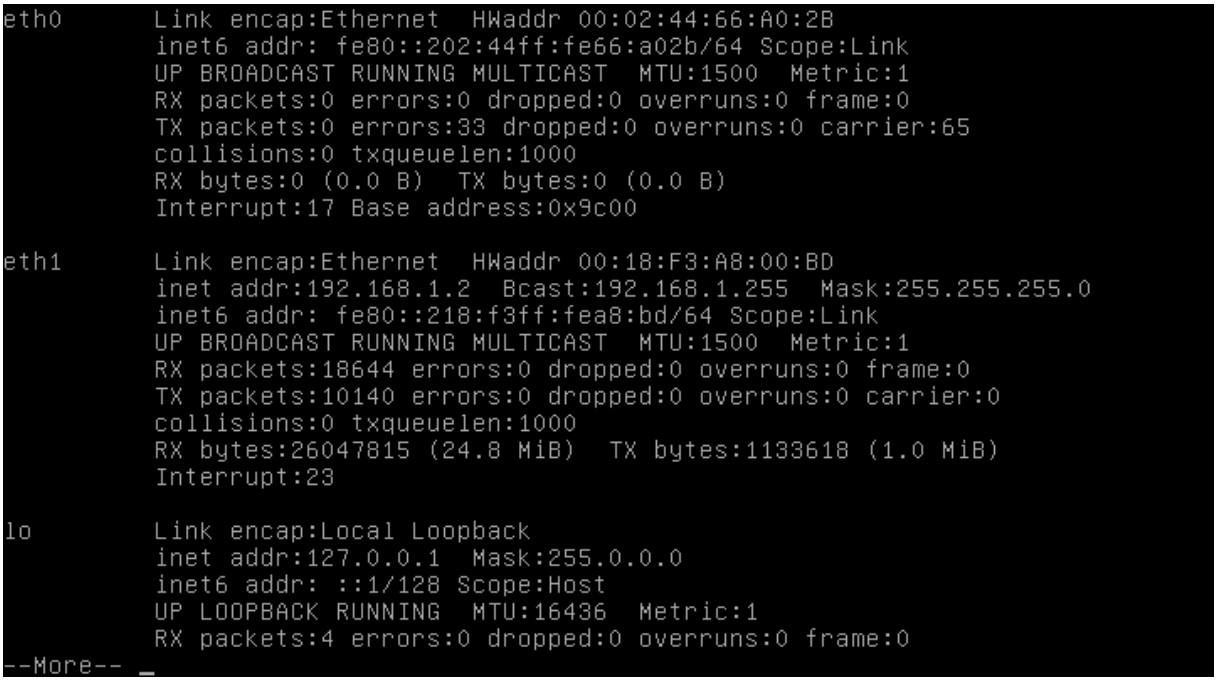


**Figura 19 – Estado das interfaces de rede do computador com função de roteador.**

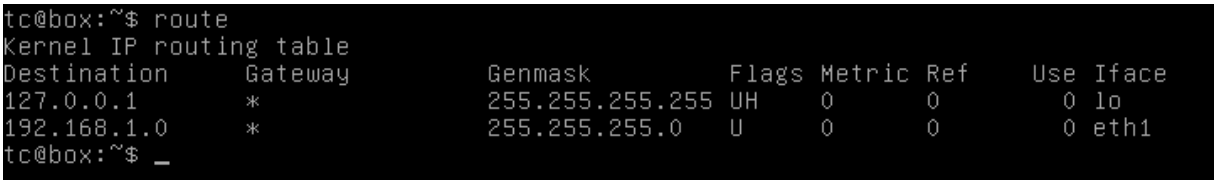


**Figura 20 – Tabela de roteamento do *kernel* do computador com função de roteador.**

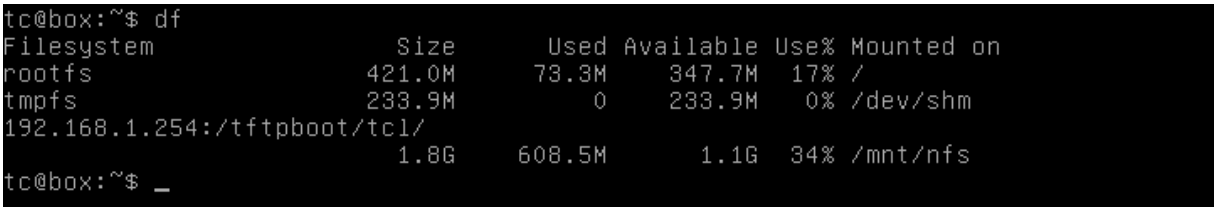


**Figura 21 – Ponto de "montagem" do diretório tce/ no computador com função de roteador.**


**Figura 22 – Consumo de memória RAM do computador com função de roteador.**

## 6.3 Configuração do Quagga

A configuração do Quagga foi feita através da sua interface *shell* integrada, vtysh. A conexão à esta interface e a exibição da tabela de roteamento através dela estão ilustradas na figura 23, onde observa-se que a tabela exibida difere da tabela exibida pelo *shell* do Tiny Core Linux apenas pela inclusão da rede de *loopback*, 127.0.0.0/8.

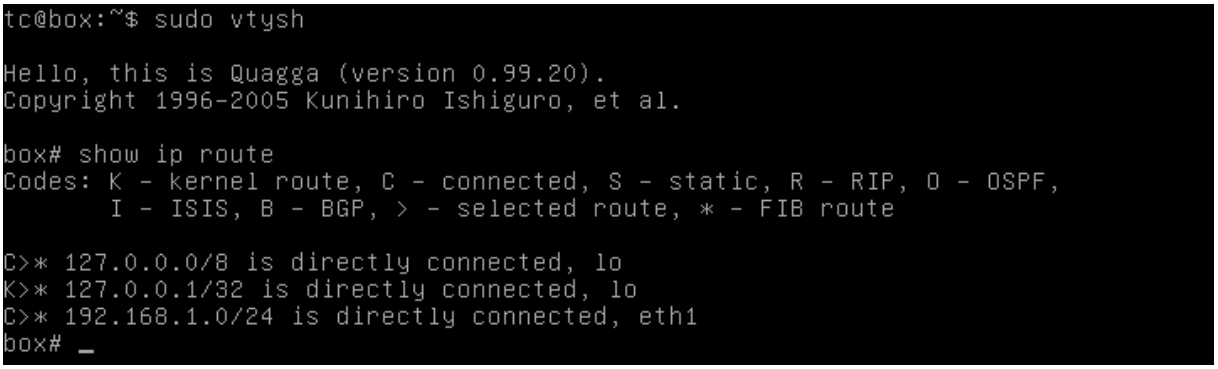


**Figura 23 – Tabela de roteamento exibida através da interface *shell* integrada do Quagga.**

Na figura 24, estão ilustradas as configurações feitas no roteador: atribuição de um endereço IP à interface de rede eth0 e ativação do protocolo RIP versão 2. Nesta figura, o comando ip forwarding habilita o roteamento no sistema operacional Linux, alterando o valor presente no arquivo /proc/sys/net/ipv4/ip_forward de 0 para 1.

```
box# configure terminal
box(config)# interface eth0
box(config-if)# ip address 172.16.1.1/28
box(config-if)# exit
box(config)#
box(config)# ip forwarding
box(config)#
box(config)# router rip
box(config-router)# version 2
box(config-router)# network 172.16.1.0/28
box(config-router)# network 192.168.1.0/24
box(config-router)# exit
box(config)# exit
box#
```

**Figura 24 – Configuração da interface de rede eth0 e do protocolo RIP versão 2.**

Após a configuração do computador com função de roteador, ele e o roteador integrado ao *modem* ADSL trocaram as rotas conhecidas através do protocolo RIP versão 2, conforme ilustrado nas figuras 25, da tabela de roteamento do computador com função de roteador, e 26, da tabela de roteamento do roteador integrado ao *modem* ADSL.

```
box# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

R>* 0.0.0.0/0 [120/2] via 192.168.1.1, eth1, 00:00:47
C>* 127.0.0.0/8 is directly connected, lo
K>* 127.0.0.1/32 is directly connected, lo
C>* 172.16.1.0/28 is directly connected, eth0
C>* 192.168.1.0/24 is directly connected, eth1
box#
```

**Figura 25 – Tabela de roteamento atualizada do computador com função de roteador.**


**Figura 26 – Tabela de roteamento do roteador integrado ao *modem* ADSL.**

Concluída a configuração do Quagga, a mesma pode ser salva, conforme ilustrado na figura 27, onde pode-se observar que cada opção de configuração é salva no arquivo de configuração do *daemon* apropriado. Em seguida, conforme ilustrado na figura 28, estes arquivos foram salvos no arquivo de *backup* do Tiny Core Linux, gravado no diretório tce/, localizado no servidor.

```
box# copy running-config startup-config
Building Configuration...
Configuration saved to /usr/local/etc/zebra.conf
Configuration saved to /usr/local/etc/ripd.conf
Configuration saved to /usr/local/etc/ospfd.conf
[OK]
box#
```

**Figura 27 – Cópia da configuração do Quagga.**

```
tc@box:~$ filetool.sh -b
Backing up files to /mnt/nfs/tce/mydata.tgztc@box:~$ ls -l /mnt/nfs/tce
total 16
-rw-rw-r--    1 tc       staff            0 Nov 21 15:45 copy2fs.flg
-rw-r--r--    1 root     root          3053 Nov 21  2012 mydata.tgz
-r--rw-r--    1 tc       staff           46 Nov 21  2012 onboot.lst
drwxrwxr-x    2 tc       staff         4096 Nov 21  2012 ondemand/
dr-xrwxr-x    2 tc       staff         4096 Nov 21 15:45 optional/
tc@box:~$ _
```

**Figura 28 – *Backup* dos arquivos de configuração do Quagga no servidor.**

A última ação a ser tomada no computador com função de roteador é a ativação do DHCP *relay agent*, para que o computador do usuário possa localizar o servidor DHCP e obter um endereço IP, conforme ilustrado na figura 29. Após a ativação deste serviço, o consumo de memória RAM do computador foi verificado novamente, tendo aumentado para 116176 kB de memória RAM, pouco mais que 113 MB.

```
tc@box:~$ sudo dhcrelay 192.168.1.254
Internet Systems Consortium DHCP Relay Agent 4.2.1-P1
Copyright 2004-2011 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Listening on LPF/eth1/00:18:f3:a8:00:bd
Sending on   LPF/eth1/00:18:f3:a8:00:bd
Listening on LPF/eth0/00:02:44:66:a0:2b
Sending on   LPF/eth0/00:02:44:66:a0:2b
Sending on   Socket/fallback
tc@box:~$ _
```

**Figura 29 – Ativação do DHCP *relay agent* no computador com função de roteador.**

## 6.4 Testes de Conexão

O computador do usuário pode localizar o servidor DHCP, do qual obteve o endereço IP 172.16.1.3 com máscara de sub-rede 255.255.255.240, com sucesso, conforme ilustrado na figura 30. Para testar seu acesso à Internet, conforme ilustrado na figura 31, foi testada a conectividade com o servidor *web* do portal Terra, cujo endereço IP é 200.154.56.80, e rastreada a rota até este servidor, através dos comandos ping e tracert.

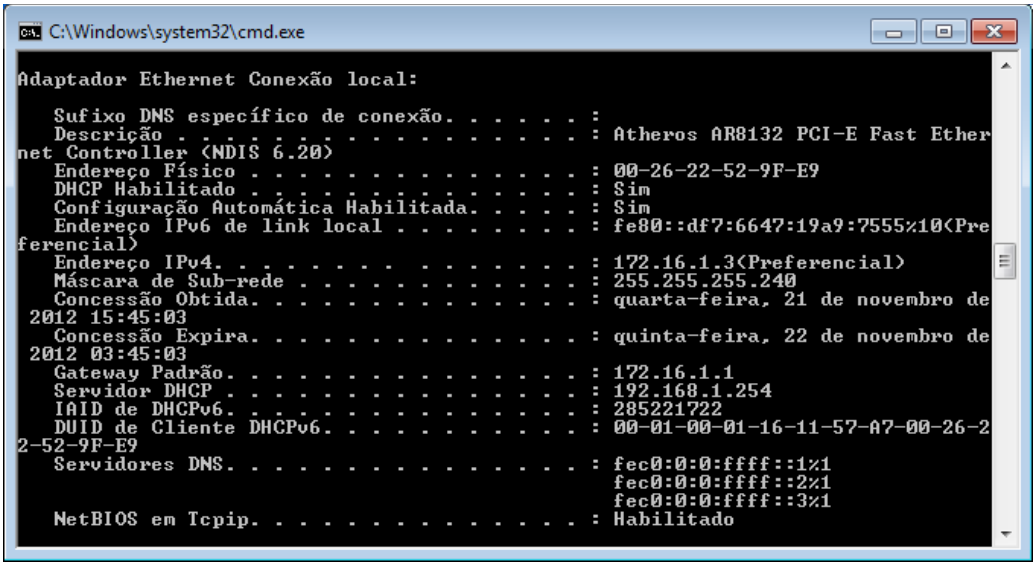


**Figura 30 – Endereço IP obtido pelo computador do usuário.**

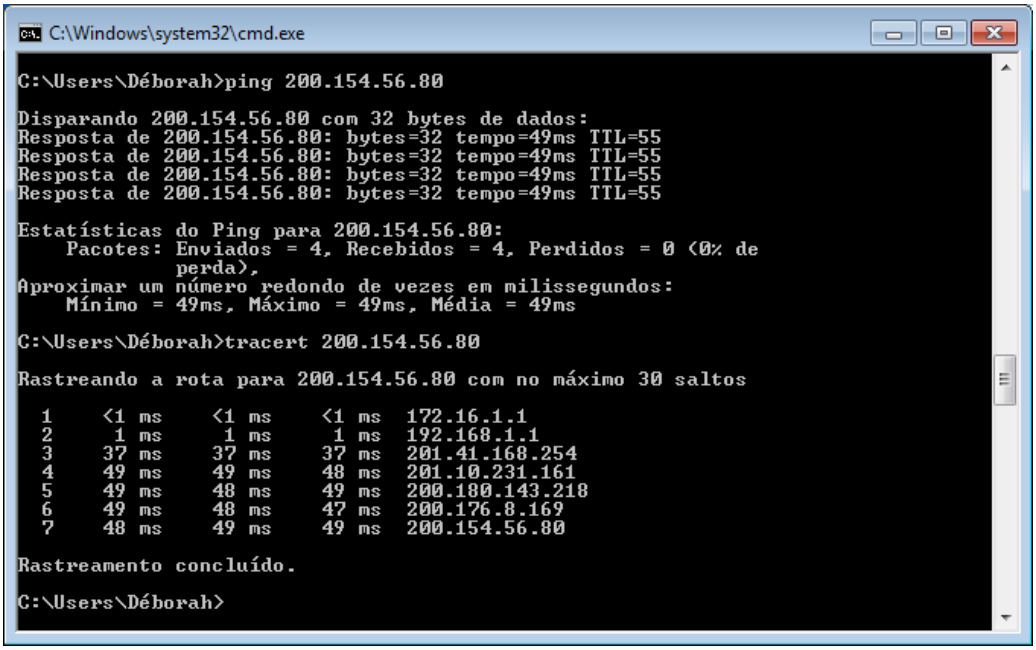


**Figura 31 – Conectividade e rastreamento da rota ao servidor 200.154.56.80.**

# 7 CONCLUSÃO

Os procedimentos e testes realizados neste trabalho mostraram que é possível utilizar um computador composto apenas de placa-mãe, processador, memória RAM e interfaces de rede, sem quaisquer unidades de armazenamento local, como um roteador, executado inteiramente a partir da memória RAM, inicializado e configurado através da rede. Para tanto, este computador executou a distribuição Tiny Core Linux do sistema operacional Linux, devido ao seu tamanho pequeno e a possibilidade de ser executada inteiramente a partir da memória RAM, o Quagga, um conjunto de programas que implementa protocolos de roteamento, e um servidor SSH, para que pudesse ser acessado remotamente através de uma conexão segura.

Para que este computador tenha sido inicializado e configurado, foi necessário que a rede possuísse um computador executando servidores DHCP, TFTP e NFS, além de um cliente SSH. Este segundo computador também foi utilizado para o *backup* dos arquivos de configuração do roteador.

A partir dos testes realizados, verificou-se que o Quagga pôde se comunicar com outro roteador – no caso, um roteador integrado a um *modem* ADSL, com suporte às verões 1 e 2 do protocolo RIP – e pode-se concluir que o computador com função de roteador pode possuir pouca memória RAM, visto que o Tiny Core Linux, o Quagga – *daemon* central e *daemons* que implementam os protocolos de roteamento RIP, RIPv2 e OSPFv2 –, o servidor SSH, o DHCP *relay agent* e a extensão que possibilita que teclados com *layout* diferente do americano sejam utilizados, executados inteiramente a partir da memória RAM, consumiram, apenas, pouco mais que 113 MB de memória RAM.

# REFERÊNCIAS

ABOUT Quagga. Disponível em < http://www.nongnu.org/quagga/index.html>.

BACKUP. Disponível em < http://wiki.tinycorelinux.net/wiki:backup>.

CORE Concepts. Disponível em <http://distro.ibiblio.org/tinycorelinux/concepts.html>.

FREQUENTLY Asked Questions. Disponível em <http://distro.ibiblio.org/tinycorelinux/faq.html>.

GRAZIANI, Rick; JOHNSON, Allan. **Routing Protocols and Concepts:** CCNA Exploration Companion Guide. 1 st. ed. Indiana: Cisco Press, 2007.

INSTALL Applications. Disponível em < http://wiki.tinycorelinux.net/wiki:install_apps>.

INTRODUCTION to Core. Disponível em <http://distro.ibiblio.org/tinycorelinux/intro.html>.

ISHIGURO, Kunihiro et al. **Quagga:** A routing software package for TCP/IP networks. Disponível em < http://www.nongnu.org/quagga/docs/quagga.pdf>.

JONES, M. T. **Linux initial RAM disk (initrd) overview:** Learn about its anatomy, creation, and use in the Linux boot process. Disponível em <http://www.ibm.com/developerworks/library/l-initrd/>.

NETBOOTING. Disponível em < http://wiki.tinycorelinux.net/wiki:netbooting>.

OVERVIEW. Disponível em <http://wiki.tinycorelinux.net/wiki:remastering_with_ezremaster>.

PERSISTENCE for Dummies, or "Why can't I find my apps and settings after booting?". Disponível em < http://wiki.tinycorelinux.net/wiki:persistence_for_dummies>.

PXELINUX. Disponível em <http://www.syslinux.org/wiki/index.php/PXELINUX>.

RUN Commands during Startup or Shutdown. Disponível em <http://wiki.tinycorelinux.net/wiki:bootlocal.sh_and_shutdown.sh>.

SYSLINUX. Disponível em <http://www.syslinux.org/wiki/index.php/SYSLINUX>.

VACHON, Bob; GRAZIANI, Rick. **Accessing the WAN:** CCNA Exploration Companion Guide. 1 st. ed. Indiana: Cisco Press, 2008.

WELCOME to The Core Project - Tiny Core Linux. Disponível em <http://distro.ibiblio.org/tinycorelinux/welcome.html>.