

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Willian Fernando de Oliveira

**Simulação do Protocolo OSPF em um Sistema
Autônomo Virtualizado utilizando *OpenWrt***

Uberlândia, Brasil

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Willian Fernando de Oliveira

**Simulação do Protocolo OSPF em um Sistema
Autônomo Virtualizado utilizando *OpenWrt***

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito parcial exigido à obtenção do grau
de Bacharel em Ciência da Computação.

Orientador: Luís Fernando Faina

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2018

Willian Fernando de Oliveira

Simulação do Protocolo OSPF em um Sistema Autônomo Virtualizado utilizando *OpenWrt*

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito parcial exigido à obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 18 de dezembro de 2018:

Luís Fernando Faina
Orientador

Anilton Joaquim da Silva

Cláudio Camargo Rodrigues

Uberlândia, Brasil
2018

Resumo

A possibilidade de conseguir reproduzir o comportamento de protocolos de roteamento em um sistema autônomo virtualizado é interessante em vários aspectos. Este projeto abrange todos os passos necessários, desde a instalação das máquinas virtuais até a configuração do protocolo de roteamento, tornando possível a montagem de um sistema autônomo utilizando um computador pessoal para analisar o comportamento do protocolo em questão, emulando falhas na transmissão de dados, capturando tráfego de pacotes nas interfaces de rede e analisando as rotas dinâmicas nos roteadores do sistema.

Palavras-chave: Sistema Autônomo, Protocolo de Roteamento, OSPF, *VirtualBox*, *OpenWrt*.

Lista de ilustrações

Figura 1 – Estrutura do AS Virtualizado para Estudo do Protocolo OSPF	10
Figura 2 – Arquitetura do Quagga	18
Figura 3 – VirtualBox - Tela Nome e Sistema Operacional	21
Figura 4 – VirtualBox - Tela Tamanho da Memória	21
Figura 5 – VirtualBox - Tela Disco Rígido	22
Figura 6 – VirtualBox - Configuração Rede: NAT	23
Figura 7 – OpenWrt - Tela Inicial	23
Figura 8 – OpenWrt - Tela após Instalação do Quagga	24
Figura 9 – VirtualBox - Tela Novo Nome da Máquina Virtual	24
Figura 10 – VirtualBox - Tela Tipo de Clone	25
Figura 11 – VirtualBox - Configuração Rede: Bridge	28
Figura 12 – OpenWrt - Arquivo de Configuração de Rede Padrão	29
Figura 13 – OpenWrt - Arquivo de Configuração de Rede: Exemplo de Host	29
Figura 14 – OpenWrt - Arquivo de Configuração de Rede: Exemplo de Roteador . .	30
Figura 15 – OpenWrt - Teste de Conectividade: Host3 para Roteador2	31
Figura 16 – OpenWrt - Teste de Conectividade: Roteador5 para Roteador3	31
Figura 17 – Quagga - Zebra: Tela Inicial	33
Figura 18 – Quagga - Zebra: Rotas do Roteador1 antes da Configuração	34
Figura 19 – Quagga - Zebra: Configuração	35
Figura 20 – Quagga - OSPFd: Tela Inicial	35
Figura 21 – Quagga - OSPFd: Rotas do Roteador1 antes da Configuração	36
Figura 22 – Quagga - OSPFd: Configuração do Protocolo no Roteador1	37
Figura 23 – Quagga - OSPFd: Rotas do Roteador1 após Configuração	37
Figura 24 – Quagga - OSPFd: Rotas do Roteador1 com Atualização Dinâmica . . .	37
Figura 25 – OpenWrt - Configuração da Rota Estática no Host1	38
Figura 26 – OpenWrt - Tabela de Rota do Host1 com Rota Estática	39
Figura 27 – Terminal - Vinculação dos Arquivos .pcap às Interfaces do VirtualBox .	41
Figura 28 – Terminal - Informação dos NICs da VM	42
Figura 29 – Wireshark - Tela Inicial	42
Figura 30 – Wireshark - Tela de Captura de Pacotes	43
Figura 31 – OpenWrt - Eleição do DR e do BDR no Roteador2	47
Figura 32 – OpenWrt - Eleição do DR e do BDR no Roteador5	47
Figura 33 – OpenWrt - Eleição do DR e do BDR no Roteador5	48
Figura 34 – OpenWrt - Ping Host3 para Roteador2	48
Figura 35 – OpenWrt - Ping: Roteador5 para Roteador3	48
Figura 36 – OpenWrt - Traceroute: Host3 para Roteador2	49

Figura 37 – OpenWrt - Traceroute: Roteador5 para Roteador3	49
Figura 38 – OpenWrt - Traceroute: Host1 para Host4	50
Figura 39 – OpenWrt - Traceroute: Host1 para Host4 após Recálculo de Rotas . . .	50
Figura 40 – Wireshark - Mensagem Hello Packet antes do ligamento do Roteador2 .	52
Figura 41 – Wireshark - Mensagem Hello Packet após o ligamento do Roteador2 . .	53
Figura 42 – Wireshark - Mensagem Database Description	53
Figura 43 – Wireshark - Mensagem LS Request	54
Figura 44 – Wireshark - Mensagem LS Update	54
Figura 45 – Wireshark - Mensagem LS Acknowledge	55

Lista de tabelas

Tabela 1 – Campos do Datagrama IPv4	12
Tabela 2 – Campos do Cabeçalho TCP	13
Tabela 3 – Tabela de Mapeamento entre Interfaces de Rede VirtualBox e OpenWrt	22
Tabela 4 – Atribuição de IPs às Interfaces dos Hosts	26
Tabela 5 – Atribuição de IPs às Interfaces do Roteador 1	26
Tabela 6 – Atribuição de IPs às Interfaces do Roteador 2	26
Tabela 7 – Atribuição de IPs às Interfaces do Roteador 3	27
Tabela 8 – Atribuição de IPs às Interfaces do Roteador 4	27
Tabela 9 – Atribuição de IPs às Interfaces do Roteador 5	27
Tabela 10 – Atribuição de IPs às Interfaces de Loopback dos Roteadores	27
Tabela 11 – Tabela de Portas VTY do Quagga	32
Tabela 12 – Tabela de Rota Padrão dos Hosts	38
Tabela 13 – Tabela de Mapeamento entre Nictrace e Interfaces de Rede	41
Tabela 14 – Tabela de Rotas do Host1	44
Tabela 15 – Tabela de Rotas do Host2	44
Tabela 16 – Tabela de Rotas do Host3	44
Tabela 17 – Tabela de Rotas do Host4	44
Tabela 18 – Tabela de Rotas do Host5	44
Tabela 19 – Tabela de Rotas do Roteador1	45
Tabela 20 – Tabela de Rotas do Roteador2	45
Tabela 21 – Tabela de Rotas do Roteador3	46
Tabela 22 – Tabela de Rotas do Roteador4	46
Tabela 23 – Tabela de Rotas do Roteador5	46
Tabela 24 – Tabela de Rotas do Roteador1 após Recálculo de Rotas	51
Tabela 25 – Tabela de Rotas do Roteador4 após Recálculo de Rotas	51

Lista de abreviaturas e siglas

AS	<i>Autonomous System</i>
BDR	<i>Backup Designated Router</i>
DR	<i>Designated Router</i>
IEFT	<i>Internet Engineering Task Force</i>
IGP	<i>Interior Gateway Protocol</i>
IP	<i>Internet Protocol</i>
IPv4	<i>Internet Protocol Version 4</i>
LSA	<i>Link-State Advertisement</i>
MAC	<i>Media Access Control</i>
MB	<i>Megabytes</i>
NAT	<i>Network Address Translation</i>
OPKG	<i>Open Package Management</i>
OSPF	<i>Open Shortest Path First</i>
RAM	<i>Random Access Memory</i>
RIP	<i>Routing Information Protocol</i>
SPF	<i>Shortest Path First</i>
TCP	<i>Transmission Control Protocol</i>
VM	<i>Virtual Machine</i>
VTY	<i>Virtual Teletype</i>
WRT	<i>Wireless Router</i>

Sumário

1	INTRODUÇÃO	10
1.1	Objetivos	10
1.2	Justificativa	11
2	REFERENCIAL TEÓRICO	12
2.1	Protocolo TCP/IP	12
2.2	Protocolo de Roteamento OSPF	14
2.3	<i>VirtualBox</i>	16
2.3.1	Adaptadores de rede	16
2.4	<i>OpenWrt</i>	17
2.4.1	Editor de arquivos	17
2.4.2	Gerenciador de pacotes	17
2.5	<i>Quagga</i>	17
2.5.1	<i>Zebra e OSPFd</i>	18
2.6	<i>Wireshark</i>	19
3	INSTALAÇÃO E CONFIGURAÇÃO DAS VMS	20
3.1	Conversão do formato de arquivo do <i>OpenWrt</i>	20
3.2	Instalação da VM no <i>VirtualBox</i>	20
3.3	Configuração do adaptador de rede da VM para <i>internet</i>	22
3.4	Instalação do <i>Quagga</i> no <i>OpenWrt</i>	23
3.5	Clonagem da VM	24
4	CONFIGURAÇÃO DO <i>OPENWRT</i>	26
4.1	Configuração dos adaptadores de rede da VM	27
4.2	Configuração das interfaces de rede no <i>OpenWrt</i>	28
4.3	Testes de conectividade	30
5	CONFIGURAÇÃO DAS ROTAS	32
5.1	Configuração das rotas dinâmicas: <i>Quagga: Zebra</i>	32
5.2	Configuração das rotas dinâmicas: <i>Quagga: OSPFd</i>	34
5.3	Configuração das rotas estáticas	38
6	CAPTURA DE TRÁFEGO E RESULTADOS	40
6.1	Vinculação do arquivo para receber dados dos pacotes	40
6.2	Inicialização das VM via terminal	41
6.3	Abertura do arquivo via <i>Wireshark</i>	42

6.4	Verificação das tabelas de rotas	43
6.4.1	Rotas nos <i>hosts</i>	43
6.4.2	Rotas nos roteadores	45
6.5	Verificação da eleição do DR e do BDR	47
6.6	Emulação de transmissão de pacotes	47
6.7	Emulação de falhas	49
6.8	Análise do tráfego de pacotes	51
	Conclusão	56
	REFERÊNCIAS	57

1 Introdução

Este projeto visa criar um ambiente de AS (*Autonomous System*) com *hosts* e roteadores virtualizados através do *VirtualBox*, utilizando o *OpenWrt* como sistema embarcado para emulação dos componentes e o *Quagga* como suíte de protocolos para executar o Protocolo OSPF. Desta forma, é possível realizar experimentos e analisar o comportamento deste protocolo e do funcionamento da rede através de um computador pessoal.

1.1 Objetivos

O objetivo geral deste trabalho é configurar um AS em ambiente virtual conforme a topologia apresentada na Figura 1 e conseguir analisar o comportamento do Protocolo OSPF como protocolo de roteamento dinâmico entre os roteadores.

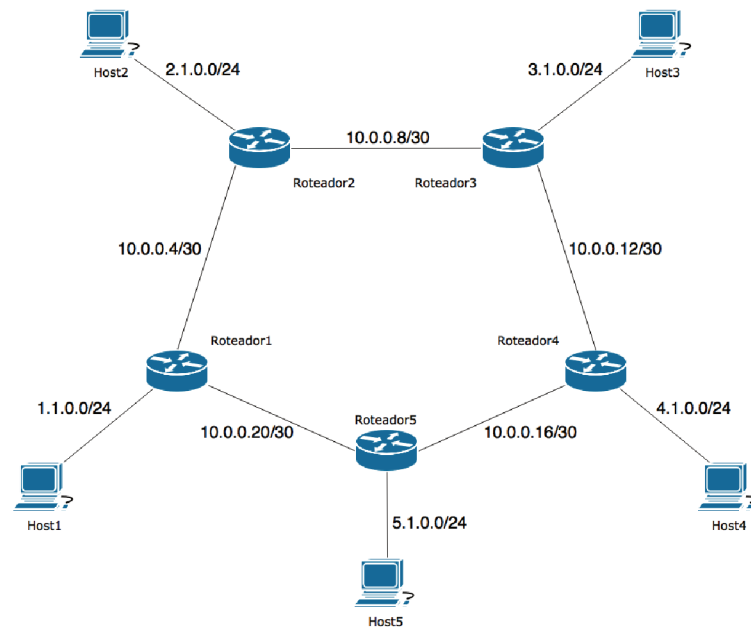


Figura 1 – Estrutura do AS Virtualizado para Estudo do Protocolo OSPF

Os principais objetivos específicos do trabalho são: a) instalação e configuração das VMs no *VirtualBox*; b) configuração do *OpenWrt* e *Quagga*; c) configuração das rotas dinâmicas nos roteadores que compõem o AS através do *Quagga*; d) configuração das rotas estáticas nos *hosts* para avaliação do Protocolo OSPF; e) captura do tráfego de pacotes utilizando *Wireshark* nas interfaces dos *hosts* e roteadores.

1.2 Justificativa

Durante a aprendizagem das disciplinas de redes de computadores, um dos grandes desafios é a possibilidade de reproduzir experimentos com os protocolos que são abordados. Seja pela dificuldade de encontrar equipamentos, como roteadores e *switches*, ou pela ausência de um laboratório, nem sempre é possível colocar na prática os fundamentos aprendidos. Seguindo os passos explicados neste projeto qualquer pessoa pode configurar seu próprio sistema autônomo e aprender diretamente através da montagem e configuração do ambiente e dos protocolos utilizando seu computador pessoal.

2 Referencial Teórico

Este capítulo apresenta os conceitos necessários para entender sobre protocolo TCP/IP, protocolo de roteamento OSPF e aborda também o básico sobre os *softwares* que serão utilizados neste projeto.

2.1 Protocolo TCP/IP

A *internet* é um grande aglomerado de redes de computadores interligados que trocam dados entre si. Para que estes computadores consigam enviar e receber pacotes é necessário que ambos estejam de acordo com as informações que trafegam entre eles. Neste sentido, o protocolo TCP/IP é um dos mais importantes da internet. Ele é uma suíte que corresponde ao protocolo da camada de rede IP juntamente com o protocolo da camada de transporte TCP. Juntos estes protocolos estabelecem os padrões que serão utilizados nas mensagens trocadas entre estes componentes.

O protocolo IP é responsável por prover o melhor esforço no serviço de transferência de dados de uma origem até um destino. Esses blocos de dados são chamados de datagramas, compostos por um cabeçalho e um corpo onde estão os dados, também chamado de *payload* (TANENBAUM; WETHERALL, 2011). Para este trabalho será utilizada a versão 4 (IPv4) e suas especificações podem ser encontradas na RFC 791. De acordo com Postel (1981a), o datagrama do IPv4 é representado como na Tabela 1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																												
Versão				Comprimento do cabeçalho				Tipo de serviço								Comprimento total (bytes)																																											
Identificador								Flags								Deslocamento de fragmentação																																											
Tempo de vida								Protocolo								Soma de verificação do cabeçalho																																											
Endereço IP da origem																																																											
Endereço IP do destino																																																											
Opções																																																											
Dados																																																											

Tabela 1 – Campos do Datagrama IPv4

- Versão (4 bits): versão do protocolo IP utilizado no datagrama, neste caso, é a versão 4;
- Comprimento do cabeçalho (4 bits): determina o tamanho do cabeçalho e onde os dados começam;
- Tipo de serviço (8 bits): determina os parâmetros de qualidade do serviço desejado;
- Comprimento total (16 bits): comprimento total do datagrama medido em bytes, com o cabeçalho e os dados;

- Identificador (16 bits): identificador atribuído pelo remetente para auxiliar na montagem do datagrama;
- *Flags* (3 bits): *flags* para controle variado;
- Deslocamento da fragmentação (13 bits): determina qual parte do datagrama este fragmento pertence;
- Tempo de vida (8 bits): determina o tempo de vida do datagrama, para que este não fique circulando pela rede indefinidamente;
- Protocolo (8 bits): identifica o protocolo utilizado na camada superior;
- Soma de verificação do cabeçalho (16 bits): auxilia na detecção de erros do cabeçalho;
- Endereço IP da origem (32 bits): indica o endereço IP de origem do datagrama;
- Endereço IP do destino (32 bits): indica o endereço IP de destino do datagrama;
- Opções (variável): pode aparecer ou não no datagrama e serve para estender o cabeçalho IP;
- Dados (variável): dados que serão trafegados na rede.

O protocolo TCP é orientado à conexão, uma vez que os processos que estão se comunicando precisam primeiro estabelecer parâmetros para a transferência de dados. Além disso ele provê transmissão confiável dos dados com checagem de erro, garantindo assim a entrega de pacotes na ordem correta sem perder a integridade das informações (KUROSE; ROSS, 2013). As especificações do protocolo podem ser encontradas na RFC 793. De acordo com Postel (1981b), o cabeçalho TCP é representado como na Tabela 2.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																																
Número da porta de origem																Número da porta de destino																																															
Número de sequência																																																															
Número de reconhecimento																																																															
Deslocamento dos dados				Reservado				Bits de controle								Tamanho da janela																																															
																Soma de verificação																Ponteiro Urgente																															
																Opções																																															
																Dados																																															

Tabela 2 – Campos do Cabeçalho TCP

- Número da porta de origem (16 bits): indica o número da porta de origem;
- Número da porta de destino (16 bits): indica o número da porta de destino;
- Número de sequência (32 bits): indica a sequência destas informações no fluxo de dados;

- Número de reconhecimento (32 bits): indica o próximo número de sequência que o destinatário está esperando receber;
- Deslocamento dos dados (4 bits): indica onde os dados começam, representa o tamanho do cabeçalho;
- Reservado (6 bits): reservado para uso futuro, deve ser zero;
- Bits de controle (6 bits): bits utilizados para controles diversos;
- Tamanho da janela (16 bits): indica o número de bytes que um destinatário está disposto a aceitar;
- Soma de verificação do cabeçalho (16 bits): auxilia na detecção de erros do cabeçalho;
- Ponteiro Urgente (16 bits): indica algum dado urgente dentro do segmento;
- Opções (variável): pode aparecer ou não no cabeçalho e serve para estender o cabeçalho TCP;
- Dados (variável): dados que serão trafegados na rede.

No protocolo TCP/IP o cabeçalho TCP é encapsulado no campo de dados do datagrama do IP. Desta forma, é oferecida as vantagens das duas abordagens tanto na camada de transporte quanto na camada de rede.

2.2 Protocolo de Roteamento OSPF

Um dos grandes desafios da *internet* é a escolha do caminho que as informações trocadas entre a origem e o destino devem percorrer. Protocolos de roteamento nasceram com o intuito de facilitar a comunicação entre os dispositivos responsáveis por transmitir esses dados entre uma ponta e outra. Esses dispositivos, chamados de roteadores ou comutadores da camada de enlace, são responsáveis por encaminhar os pacotes até seu destino final através de um caminho ou rota definidos através de um algoritmo. Este algoritmo deve escolher o caminho de menor custo entre o destino e a origem, que geralmente é aquele que possui o menor número de enlaces (KUROSE; ROSS, 2013).

Existem duas classes de algoritmos de roteamento: os algoritmos não-adaptivos e os algoritmos adaptivos. Os algoritmos não adaptivos utilizam rotas estáticas, pois não baseiam sua decisão em métricas ou estimativas da topologia de rede. Desta forma eles também não conseguem responder à falhas. Por outro lado, os algoritmos adaptivos mudam as decisões de rota de acordo com a topologia da rede e utilizam uma série de métricas para tomar esta decisão, formando assim rotas dinâmicas (TANENBAUM; WETHERALL, 2011). Neste projeto os algoritmos não-adaptivos serão representados pela

configuração que será feita nos *hosts* e os algoritmos adaptivos serão representados pelo OSPF que será configurado nos roteadores.

O *Open Shortest Path First* (OSPF) é um algoritmo de roteamento dinâmico desenvolvido para o protocolo IP por um grupo de pesquisadores da *Internet Engineering Task Force* (IETF). Ele é classificado como um *Internet Gateway Protocol* (IGP), ou seja, ele distribui informações das rotas dos roteadores que pertencem ao mesmo sistema autônomo. O OSPF é um protocolo de domínio público e suas especificações podem ser encontradas na RFC 2328.

O OSPF é considerado um algoritmo do estado do enlace, isto porque após dois roteadores vizinhos se descobrirem e estabelecerem um acordo para se comunicarem, são trocadas informações sobre o estado das suas interfaces e das outras já recebidos por vizinhos em sua adjacência. Este processo é feito através da troca de mensagens por inundação chamadas de *Link State Advertisement* (LSA) (TADIMETY, 2015). Sendo assim, cada roteador mantém um banco de dados da topologia do AS (chamado de *Link State Database*). Neste banco de dados existe a informação de custo para transmitir um pacote entre os enlaces do AS que é usada como métrica e através do algoritmo *Dijkstra* cada roteador pode construir uma árvore com o menor caminho para cada destino do AS tendo ele próprio como raiz (MOY, 1998b).

Um roteador pode estar rodando diversos protocolos, de forma que pode existir um roteador vizinho rodando um protocolo diferente do seu. Sendo assim, a adjacência entre roteadores só é válida se eles estão utilizando o mesmo protocolo, neste caso o Protocolo OSPF (DOYLE, 2005).

De acordo com Moy (1998a), os tipos de mensagens do OSPF são:

- *Hello Packet*: utilizado para descobrir e manter o relacionamento entre os vizinhos;
- *Database Description*: informações do banco de dados;
- *Link State Request*: requisita informações para o banco de dados vizinho;
- *Link State Update*: atualiza informações vindas do banco de dados vizinho;
- *Link State Ack*: inunda informações sobre reconhecimento.

Uma parte importante do OSPF é a eleição do DR (*designated router*) e do BDR (*backup designated router*) entre roteadores vizinhos. Em uma rede existe várias ligações entre os roteadores fazendo o número de vizinhos, muitas vezes em comum, serem muito repetitivos. Afim de evitar sobrecarrego do tráfego de informações devido às inundações de mensagens LSA, o protocolo elege um roteador que vai concentrar as informações e outro que serve de backup caso o primeiro falhe. Desta forma, aumenta-se a otimização e

performance do protocolo e garante-se a sincronização das bases de dados dos roteadores (MOY, 1998b) (TADIMETY, 2015).

A eleição é feita durante a troca de mensagens *Hello Packet*. Se nenhum roteador foi eleito ainda é escolhido o roteador com maior prioridade ou maior **router-id** para ser o DR, caso contrário o roteador já eleito é aceito. Após a eleição do DR, o BDR é eleito seguindo os mesmos critérios (MOY, 1998a).

2.3 VirtualBox

VirtualBox é um *software* grátis e *open source* de virtualização atualmente desenvolvido pela *Oracle*. Através dele é possível criar um ambiente virtual para a instalação de sistemas operacionais (sistemas visitantes) dentro de um outro sistema operacional (sistema hospedeiro) como computadores independentes, porém que compartilham o mesmo hardware. A versão atual, 5.2.20, pode ser obtida no site do *VirtualBox*¹. Uma vez instalada a VM que irá emular o componente, um aspecto que merece destaque é a configuração da interface de rede.

2.3.1 Adaptadores de rede

O *VirtualBox* permite que cada VM tenha até oito adaptadores de rede. Quatro deles são configuráveis pela interface do *software*, enquanto os quatro restantes podem ser configurados via linha de comando no terminal do sistema hospedeiro (ORACLE, 2018).

Cada adaptador de rede opera em um modo. Dentre as opções que o *VirtualBox* oferece serão utilizadas NAT e Bridge neste projeto, que segundo Oracle (2018) são:

- NAT: modo utilizado para que o sistema hospedeiro compartilhe *internet* com o sistema visitante de forma transparente. O *VirtualBox* traduz os pacotes enviados/-recebidos pelo sistema visitante como se fossem enviados/recebidos pelo sistema hospedeiro. Desta forma é como se o próprio sistema hospedeiro estivesse se comunicando diretamente com a *internet*;
- Placa em modo *Bridge*: modo que permite que o *VirtualBox* utilize o *driver* de um hardware de rede do sistema hospedeiro como uma nova interface. Desta forma é como existisse um cabo ligando a interface virtual do roteador *OpenWrt* no hardware da máquina que hospeda o *VirtualBox*. Será útil para que seja possível capturar pacotes do sistema visitante.

¹ <<https://www.virtualbox.org/wiki/Downloads>> Acesso em: 3 nov. 2018.

2.4 OpenWrt

O *OpenWrt* é um projeto *open source* baseado em *Linux* para sistemas embarcados, especificamente roteadores. Para este trabalho será utilizada a versão 15.01.1, chamada de *Chaos Calmer* e sua imagem pode ser baixada na seção de *download* do site do *OpenWrt*². A documentação do *OpenWrt* pode ser acessada no site estilo *wiki*³.

2.4.1 Editor de arquivos

O *OpenWrt* possui um editor de arquivos padrão que será utilizado para editar os arquivos de configuração de rede. Ele pode ser acionado através do comando `vi` seguido do caminho do arquivo.

O editor possui dois modos: comando e edição. Para entrar no modo de comando deve-se pressionar a tecla **ESC**. Neste modo, ao se posicionar em um linha, pressionando a tecla **D** ela é excluída. Para entrar no modo de edição deve-se pressionar a tecla **I** ou a tecla **A**. O `vi` começa no modo de comando.

Para sair do `vi` é necessário estar no modo de comando e executar algum dos comandos seguintes:

- `:w` para salvar o arquivo no disco, sobrescrevendo o arquivo antigo;
- `:q` para sair sem salvar ou `:q!` para sair forçadamente;
- `:wq` para salvar no disco e sair ou `:wq!` para salvar e sair de forma forçada.

2.4.2 Gerenciador de pacotes

Para baixar e instalar pacotes no *OpenWrt*, seja de repositórios na *internet* ou local, é necessário utilizar o gerenciador de pacotes que vem instalado, chamado OPKG. O OPKG é invocado através do comando `opkg` seguido do argumento, que pode ser para atualizar, para instalar, para remover, para listar ou para visualizar informações de um pacote. Existem vários pacotes disponíveis para o *OpenWrt*, porém este projeto necessitará apenas do *Quagga*, que servirá para implementar o OSPF nos roteadores.

2.5 Quagga

Quagga é uma suíte *open source* de roteamento para plataformas *Unix*. Ele permite gerenciar e implementar uma série de protocolos, como RIP, OSPF, BGP, dentre outros,

² <https://downloads.openwrt.org/chaos_calmer/15.05.1/x86/generic/openwrt-15.05.1-x86-generic-combined-ext4.img.gz> Acesso em: 3 nov. 2018.

³ <<https://wiki.openwrt.org/doc/start>> Acesso em: 3 nov. 2018.

tanto para IPv4 quanto para IPv6. Através do *Quagga* é possível trocar informações de roteamento com outros roteadores, atualizando a tabela de rotas através do protocolo utilizado (ISHIGURO et al., 2017). A arquitetura é apresentada na Figura 2

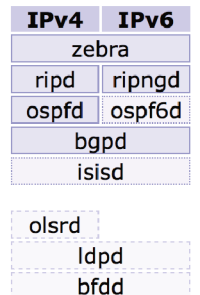


Figura 2 – Arquitetura do Quagga

Fonte: <http://www.nongnu.org/quagga>

2.5.1 Zebra e OSPFd

Zebra é um gerenciador de protocolos que funciona como uma camada de abstração entre o *daemon* do protocolo de roteamento e o sistema que está implementando o protocolo. Ele provê a atualização das tabelas de rotas, consulta de interfaces e redistribuição de rotas entre protocolos de roteamento diferentes. Já o *OSPFd* é o *daemon* responsável pelo protocolo OSPF versão 2 no *Quagga* (ISHIGURO et al., 2017).

Dentre os vários comandos descritos por Ishiguro et al. (2017) para o *Zebra*, destaca-se:

- `show ip route`: mostra a tabela de rotas com informação sobre os protocolos utilizados;
- `show interface`: mostra dados das interfaces de rede.

Para o *OSPFd*, destaca-se:

- `show ip ospf route`: mostra a tabela de rotas OSPF;
- `show ip ospf neighbor`: mostra dados dos roteadores vizinhos;
- `show ip ospf database`: mostra informações dos estados dos links;
- `show ip ospf interface`: mostra dados das interfaces, com informações complementares sobre OSPF;

2.6 Wireshark

O *Wireshark* é um analisador de pacotes *open source* utilizado para monitorar o tráfego de rede de um computador. Atualmente está na versão 2.6.4 que pode ser baixada na seção de download do site do *software*⁴.

Através dele é possível fazer captura em tempo real do tráfego de uma rede, mostrando informações detalhadas dos pacotes. Seus usos mais frequentes são para depurar implementações de protocolos, verificar problemas na rede, examinar problemas de segurança e aprender sobre os protocolos de rede (LAMPING; SHARP; WARNICKE, 2018).

⁴ <<https://www.wireshark.org/#download>> Acesso em: 3 nov. 2018.

3 Instalação e configuração das VMs

Neste capítulo serão apresentados os passos necessários para a instalação e configuração das VMs, divididos nas seguintes etapas:

- conversão do formato de arquivo do *OpenWrt*;
- instalação da VM no *VirtualBox*;
- configuração do adaptador de rede da VM para conexão com a *internet*;
- instalação do *Quagga* no *OpenWrt*;
- clonagem da VM.

3.1 Conversão do formato de arquivo do *OpenWrt*

Após baixar e descompactar a imagem através de um compactador de arquivos, é necessário convertê-la para um formato que o *VirtualBox* reconheça. Para isto, deve-se abrir o terminal do sistema operacional hospedeiro e navegar até a pasta onde a imagem está salva. Dentro da pasta, deve-se executar o comando seguinte para converter a imagem:

```
> VBoxManage convertfromraw --format VDI [nome da imagem baixada].img
[nome para a imagem convertida].vdi
```

Os trechos entre colchetes, nome da imagem baixada e nome da para a imagem convertida, devem ser substituídos pelos nomes correspondentes. Para este projeto o comando executado foi: `VBoxManage convertfromraw -format VDI openwrt-15.05.1-x86-generic-combined-ext4.img openwrt.vdi`

3.2 Instalação da VM no *VirtualBox*

Para instalar uma nova VM no *VirtualBox* pressione o botão Novo na barra de ferramentas da tela inicial. Na janela Nome e Sistema Operacional deve-se entrar com um nome, um tipo e uma versão para a VM. Para montar o AS do projeto os nomes utilizados para padronização e identificação das VMs serão: Host1, Host2, Host3, Host4, Host5, Roteador1, Roteador2, Roteador3, Roteador4 e Roteador5. Uma vez que as demais máquinas serão clonadas a partir da primeira, qualquer nome acima pode ser escolhido. Todas as máquinas deverão ser do tipo *Linux* e da versão 2.6 de 32 bits (Figura 3). Após preencher os dados deve-se pressionar Continuar.



Figura 3 – VirtualBox - Tela Nome e Sistema Operacional

Na janela **Tamanho da memória** deve-se escolher o tamanho da memória RAM que será utilizado pela VM. Como o *OpenWrt* é um sistema muito otimizado, para ser instalado em dispositivos com recursos limitados, o recomendado é que se utilize 128 MB (Figura 4). Pressionar **Continuar** após a escolha.



Figura 4 – VirtualBox - Tela Tamanho da Memória

Na janela **Disco Rígido** deve ser marcada a opção **Utilizar um disco virtual existente** e escolhido o arquivo convertido na etapa anterior (Figura 5). Ao clicar em **Criar** a VM será criada e apresentada na tela principal do *VirtualBox*.



Figura 5 – VirtualBox - Tela Disco Rígido

3.3 Configuração do adaptador de rede da VM para *internet*

Para instalar o *Quagga* é necessário conexão com a *internet*, que será compartilhada com a VM pelo sistema hospedeiro através do modo de rede NAT. O *VirtualBox* permite até quatro adaptadores de redes habilitados para uma VM na interface do *software* que são mapeados de acordo com cada interface de rede do *VirtualBox* de acordo com a Tabela 3.

Adaptador de Rede (VirtualBox)	Interface de Rede (OpenWrt)
Adaptador 1	Eth0
Adaptador 2	Eth1
Adaptador 3	Eth2
Adaptador 4	Eth3

Tabela 3 – Tabela de Mapeamento entre Interfaces de Rede VirtualBox e OpenWrt

O *OpenWrt* já vem por padrão com a interface **Eth1** configurada para funcionar através do modo NAT. Para que não seja necessário mudar as configurações de rede internamente no *OpenWrt*, é possível habilitar o **Adaptador 2** da VM e selecionar NAT como modo de rede através do botão **Configurações** localizado na tela inicial do *VirtualBox*.

Dentro da janela de **Configurações**, na aba **Rede**, deve-se escolher **Adaptador 2**, habilitar e selecionar NAT na opção **Conectado a** como mostrado na Figura 6. Neste momento não é necessário modificar os outros adaptadores de rede. A partir desta configuração a VM estará apta a conectar-se à *internet* e fazer o download do *Quagga*.

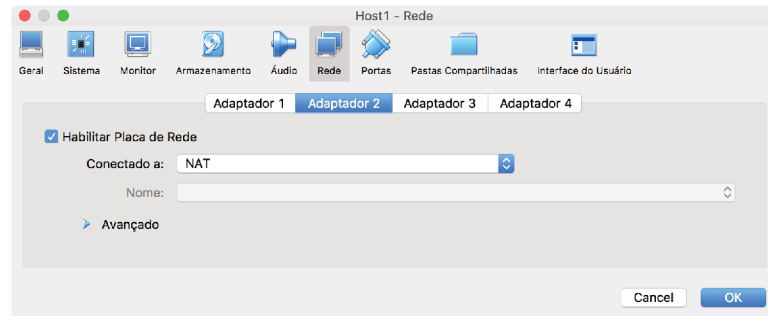


Figura 6 – VirtualBox - Configuração Rede: NAT

3.4 Instalação do *Quagga* no *OpenWrt*

Após iniciar a VM através do botão Iniciar na tela principal do *VirtualBox* o sistema será carregado e a tela inicial do *OpenWrt* será apresentada (Figura 7). Para fazer o download e instalar o *Quagga*, *Zebra* e *OSPFd*, os comandos seguintes devem ser executados no console do *OpenWrt*, um por vez:

```
> opkg update
> opkg install quagga
> opkg install quagga-zebra
> opkg install quagga-ospfd
```

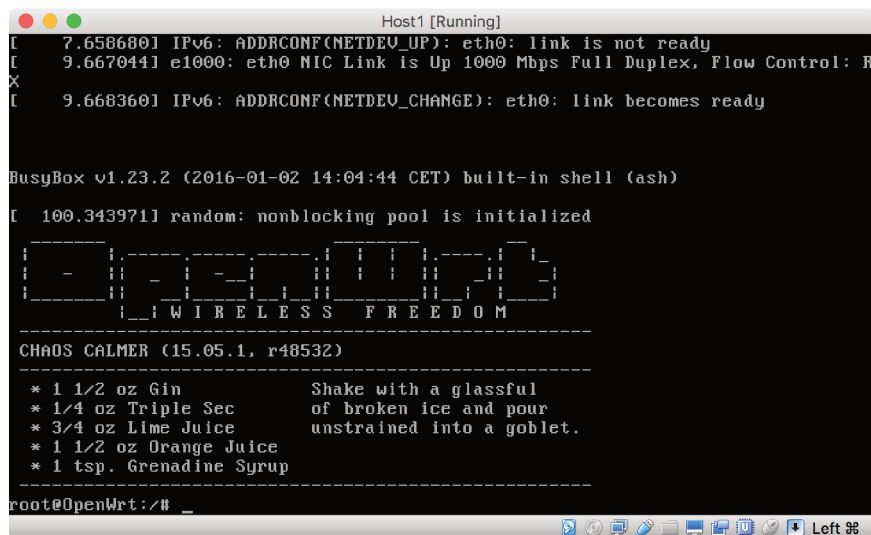
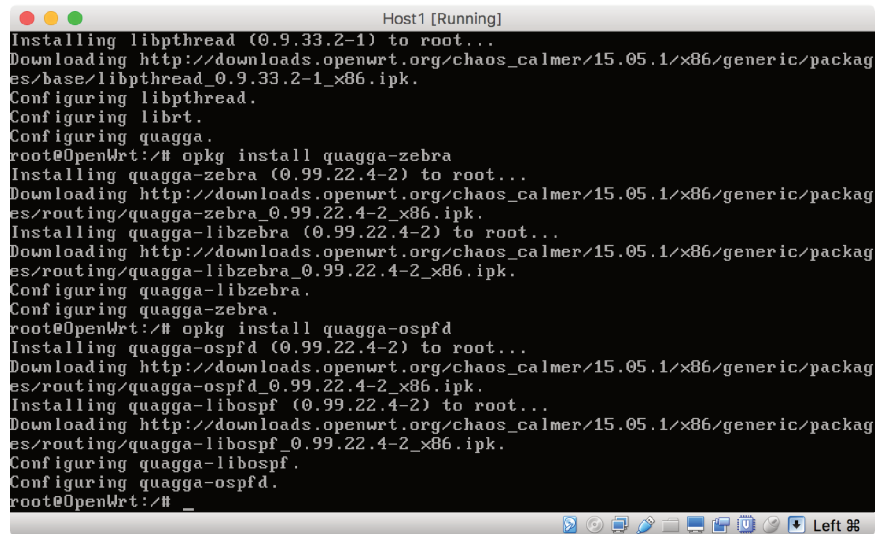


Figura 7 – OpenWrt - Tela Inicial

No final da execução dos quatro comandos, *Quagga*, *Zebra* e *OSPFd* estarão instalados (Figura 8). Para desligar a VM utilizar o comando `halt`.



```
Host1 [Running]
Installing libpthread (0.9.33.2-1) to root...
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/x86/generic/packages/base/libpthread_0.9.33.2-1_x86.ipk.
Configuring libpthread.
Configuring librt.
Configuring quagga.
root@OpenWrt:~# opkg install quagga-zebra
Installing quagga-zebra (0.99.22.4-2) to root...
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/x86/generic/packages/routing/quagga-zebra_0.99.22.4-2_x86.ipk.
Installing quagga-libzebra (0.99.22.4-2) to root...
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/x86/generic/packages/routing/quagga-libzebra_0.99.22.4-2_x86.ipk.
Configuring quagga-libzebra.
Configuring quagga-zebra.
root@OpenWrt:~# opkg install quagga-ospfd
Installing quagga-ospfd (0.99.22.4-2) to root...
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/x86/generic/packages/routing/quagga-ospfd_0.99.22.4-2_x86.ipk.
Installing quagga-libospf (0.99.22.4-2) to root...
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/x86/generic/packages/routing/quagga-libospf_0.99.22.4-2_x86.ipk.
Configuring quagga-libospf.
Configuring quagga-ospfd.
root@OpenWrt:~#
```

Figura 8 – OpenWrt - Tela após Instalação do Quagga

3.5 Clonagem da VM

Uma vez que as configurações padrões já foram concluídas pode-se clonar a VM nas demais que serão utilizadas para formar o AS. Para isso, deve-se clicar com o botão direito no ícone da VM, na tela principal do *VirtualBox*.

Na janela Novo nome da máquina virtual a opção Reinicialize o endereço MAC de todas as placas de rede deve ser marcada e a recomendação é que no campo nome seja inserido algum dos nomes que ainda não foram utilizados (Figura 9). Ao clicar

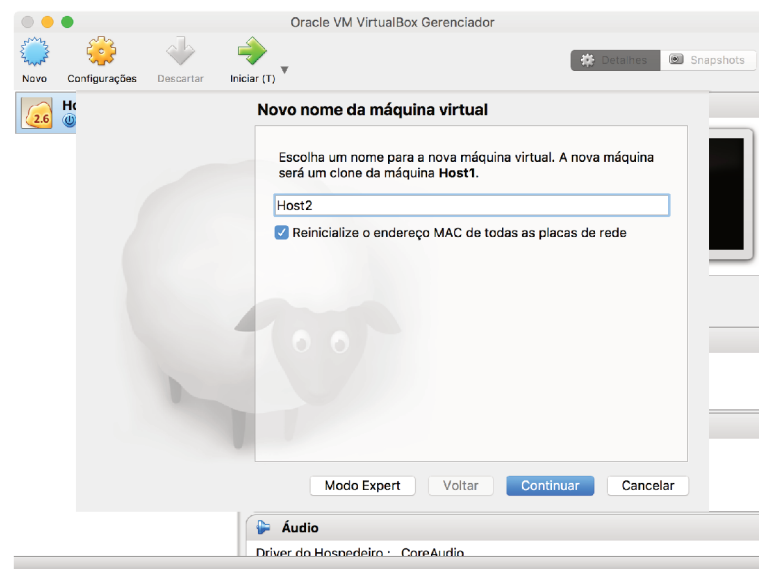


Figura 9 – VirtualBox - Tela Novo Nome da Máquina Virtual

em Continuar, a janela Tipo de Clone será apresentada e a opção marcada será a Clone completo. Ao clicar em Clonar a nova VM será criada (Figura 10).



Figura 10 – VirtualBox - Tela Tipo de Clone

O processo de clonagem deve ser repetido nove vezes ao todo, sempre escolhendo um nome que ainda não foi utilizado para facilitar na identificação. Após terminar todas as clonagens as dez máquinas virtuais devem aparecer na tela inicial do *VirtualBox*.

4 Configuração do *OpenWrt*

Neste capítulo serão apresentados os passos necessários para a configuração das interfaces de rede de todos os roteadores e *hosts* do AS, divididos nas seguintes etapas:

- configuração dos adaptadores de rede da VM;
- configuração das interfaces de rede no *OpenWrt*;
- testes de conectividade.

Para construir o AS da Figura 1 serão usados os endereços de IP e máscaras de rede descritos na Tabela 4 para os *hosts* e nas Tabelas 5, 6, 7, 8, 9 para cada um dos respectivos roteadores. Além disso, serão usados os endereços de *loopback* da Tabela 10 somente para os roteadores.

Host	Interface	IP	Máscara
Host1	Eth0	1.1.0.1	255.255.255.0
Host2	Eth0	2.1.0.1	255.255.255.0
Host3	Eth0	3.1.0.1	255.255.255.0
Host4	Eth0	4.1.0.1	255.255.255.0
Host5	Eth0	5.1.0.1	255.255.255.0

Tabela 4 – Atribuição de IPs às Interfaces dos Hosts

Roteador	Interface	IP	Máscara
Roteador1	Eth0	1.1.0.254	255.255.255.0
Roteador1	Eth1	10.0.0.6	255.255.255.252
Roteador1	Eth2	10.0.0.21	255.255.255.252

Tabela 5 – Atribuição de IPs às Interfaces do Roteador 1

Roteador	Interface	IP	Máscara
Roteador2	Eth0	2.1.0.254	255.255.255.0
Roteador2	Eth1	10.0.0.5	255.255.255.252
Roteador2	Eth2	10.0.0.10	255.255.255.252

Tabela 6 – Atribuição de IPs às Interfaces do Roteador 2

Roteador	Interface	IP	Máscara
Roteador3	Eth0	3.1.0.254	255.255.255.0
Roteador3	Eth1	10.0.0.14	255.255.255.252
Roteador3	Eth2	10.0.0.9	255.255.255.252

Tabela 7 – Atribuição de IPs às Interfaces do Roteador 3

Roteador	Interface	IP	Máscara
Roteador4	Eth0	4.1.0.254	255.255.255.0
Roteador4	Eth1	10.0.0.13	255.255.255.252
Roteador4	Eth2	10.0.0.18	255.255.255.252

Tabela 8 – Atribuição de IPs às Interfaces do Roteador 4

Roteador	Interface	IP	Máscara
Roteador5	Eth0	5.1.0.254	255.255.255.0
Roteador5	Eth1	10.0.0.22	255.255.255.252
Roteador5	Eth2	10.0.0.17	255.255.255.252

Tabela 9 – Atribuição de IPs às Interfaces do Roteador 5

Roteador	Interface	IP	Máscara
Roteador1	Loopback	1.1.1.1	255.255.255.255
Roteador2	Loopback	2.2.2.2	255.255.255.255
Roteador3	Loopback	3.3.3.3	255.255.255.255
Roteador4	Loopback	4.4.4.4	255.255.255.255
Roteador5	Loopback	5.5.5.5	255.255.255.255

Tabela 10 – Atribuição de IPs às Interfaces de Loopback dos Roteadores

4.1 Configuração dos adaptadores de rede da VM

Na Seção 3.3 foi apresentado como mudar o adaptador de rede da VM para possibilitar a conexão com a *internet*. Porém, para que o AS funcione corretamente e seja possível capturar os pacotes para análise no *Wireshark*, será utilizado outro modo de rede chamado **Bridge** para todas as interfaces de redes ativas.

Para isto deve-se abrir a aba de **Rede** no botão de **Configuração** da tela principal do *VirtualBox*. A configuração deverá ser feita como mostra na Figura 11. No campo **Nome** pode ser escolhida qualquer uma das opções apresentadas, contanto que seja a mesma para todas as interfaces. A recomendação é que seja escolhido o nome de uma placa que não esteja sendo utilizada. Por exemplo, se a máquina hospedeira está com o *Wi-Fi* ligado, é melhor utilizar o adaptador da placa de rede *Ethernet* para não capturar pacotes indesejados.

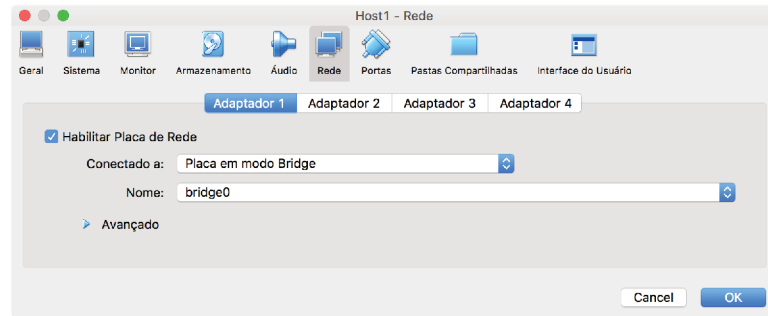


Figura 11 – VirtualBox - Configuração Rede: Bridge

Como é possível perceber através da Tabela 4, todos os *hosts* terão apenas uma interface de rede. Dessa forma, somente o adaptador de rede 1 será configurado na VM e o adaptador de rede 2, que foi ativado na Seção 3.3, será desativado.

Por outro lado, todos os roteadores terão 3 interfaces de rede habilitadas. Sendo assim, além de configurar os dois adaptadores que já estão ativos, ainda será necessário ativar o terceiro que corresponderá à interface Eth2, conforme a Tabela 3.

4.2 Configuração das interfaces de rede no OpenWrt

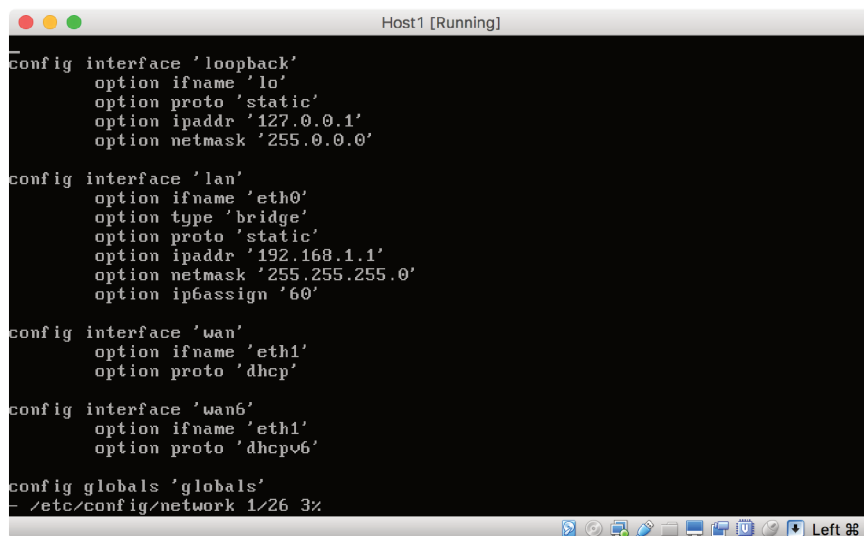
Após configurar todos os adaptadores de rede das VMs é necessário configurar também as interfaces de rede no *OpenWrt*. Para isto, deve-se iniciar as VMs e utilizar o editor de texto do *OpenWrt* para abrir o arquivo de configuração da rede localizado no caminho `/etc/config/network`, utilizando o comando:

```
> vi /etc/config/network
```

O arquivo será então carregado e mostrado como na Figura 12. Este arquivo deverá ser configurado de acordo com as tabelas de IPs e máscara de cada componente do AS, de acordo com o seguinte modelo:

```
config interface 'ethx'
    option ifname 'ethx'
    option proto 'static'
    option ipaddr 'a.b.c.d'
    option netmask 'e.f.g.h'
```

Deve-se trocar o x em `ethx` pelo número correspondente da interface e os valores em `a.b.c.d` pelo endereço IP e os valores em `e.f.g.h` pela máscara de rede. A ressalva é que não será mudado o valor de `config globals 'globals'` para nenhum componente e o valor de `config interface 'loopback'` somente será mudado nos roteadores, de acordo com a Tabela 10. As outras interfaces presentes no arquivo padrão devem ser excluídas.

A screenshot of a terminal window titled "Host1 [Running]". The terminal displays the configuration for the network interfaces and global settings. The configuration is as follows:

```
config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config interface 'lan'
    option ifname 'eth0'
    option type 'bridge'
    option proto 'static'
    option ipaddr '192.168.1.1'
    option netmask '255.255.255.0'
    option ip6assign '60'

config interface 'wan'
    option ifname 'eth1'
    option proto 'dhcp'

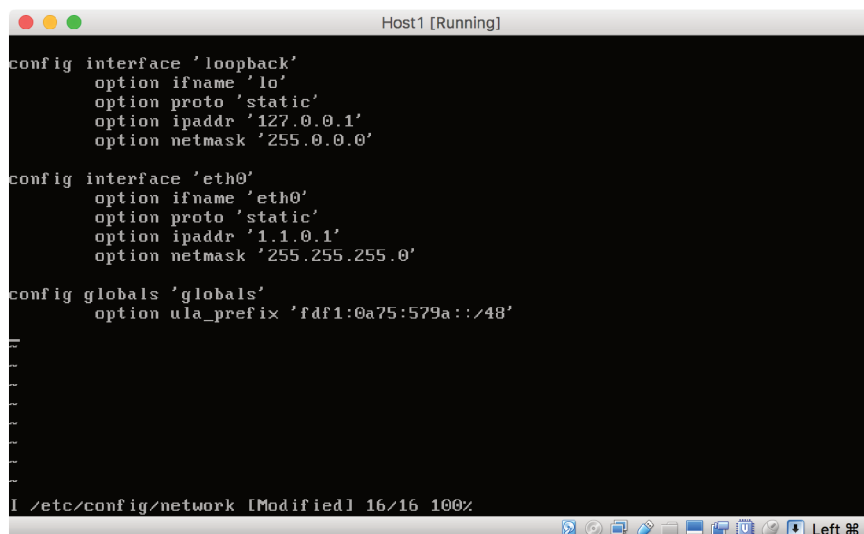
config interface 'wan6'
    option ifname 'eth1'
    option proto 'dhcpv6'

config globals 'globals'
    /etc/config/network 1/26 3%
```

The terminal window has a standard macOS-style title bar with red, yellow, and green buttons. At the bottom, there is a dock with various system icons and a status bar showing "Left %".

Figura 12 – OpenWrt - Arquivo de Configuração de Rede Padrão

Para entrar no modo de edição apertar a tecla **A** e para sair apertar a tecla **ESC**. Após editar o arquivo, os *hosts* devem ficar semelhante ao mostrado na Figura 13 e os roteadores ao mostrado Figura 14.

A screenshot of a terminal window titled "Host1 [Running]". The terminal displays the configuration for a host. The configuration is as follows:

```
config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config interface 'eth0'
    option ifname 'eth0'
    option proto 'static'
    option ipaddr '1.1.0.1'
    option netmask '255.255.255.0'

config globals 'globals'
    option ula_prefix 'fd11:0a75:579a::/48'
```

The terminal window has a standard macOS-style title bar with red, yellow, and green buttons. At the bottom, there is a dock with various system icons and a status bar showing "Left %".

Figura 13 – OpenWrt - Arquivo de Configuração de Rede: Exemplo de Host

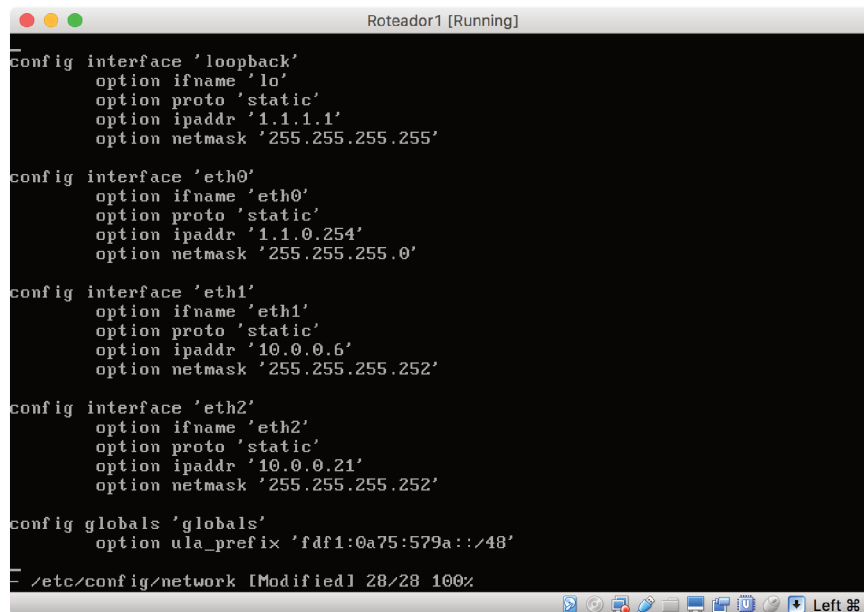
Após sair do modo de edição, digitar comando seguinte para salvar e sair do editor:

```
> :wq
```

Para que as alterações sejam carregadas é necessário reiniciar as interfaces de rede, através do comando:

```
> /etc/init.d/network restart
```

Todos os *hosts* e roteadores já estarão conectados através das redes configuradas após terminar esta etapa.



```
config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '1.1.1.1'
    option netmask '255.255.255.255'

config interface 'eth0'
    option ifname 'eth0'
    option proto 'static'
    option ipaddr '1.1.0.254'
    option netmask '255.255.255.0'

config interface 'eth1'
    option ifname 'eth1'
    option proto 'static'
    option ipaddr '10.0.0.6'
    option netmask '255.255.255.252'

config interface 'eth2'
    option ifname 'eth2'
    option proto 'static'
    option ipaddr '10.0.0.21'
    option netmask '255.255.255.252'

config globals 'globals'
    option ula_prefix 'fd1:0a75:579a::/48'

/etc/config/network [Modified] 28/28 100%
```

Figura 14 – OpenWrt - Arquivo de Configuração de Rede: Exemplo de Roteador

4.3 Testes de conectividade

Como nesta etapa ainda não foi configurada nenhuma rota estática nem dinâmica, é possível testar apenas a conectividade entre os componentes que estão diretamente conectados e na mesma rede. Para verificar a tabela de rotas do componente pode-se utilizar o comando:

```
> route -n
```

Para verificar a conectividade pode-se utilizar o comando:

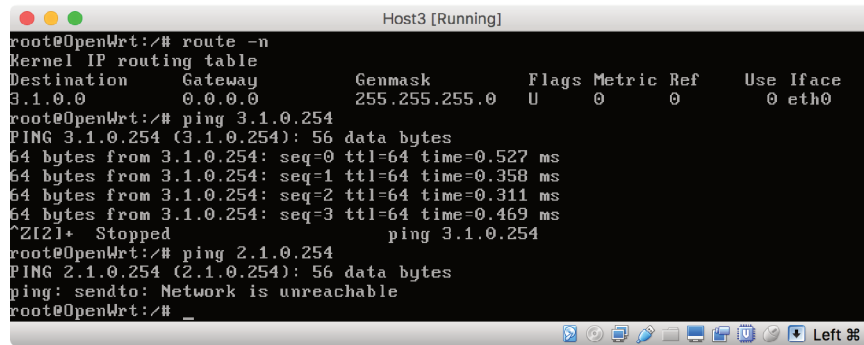
```
> ping [end. de IP]
```

Por exemplo, o **Host3** está conectado na rede **3.1.0.0/24**, sendo assim ele pode transmitir pacotes ao **Roteador3** que também está conectado nesta rede. Porém não consegue transmitir pacotes ao **Roteador2**, pois não estão conectados diretamente em nenhuma rede e não existe nenhuma rota ainda programada para que isto ocorra (Figura 15).

O **Roteador5** está conectado na rede **10.0.0.16/32**, então é possível mandar pacotes ao **Roteador4** pela interface que estão conectados, porém não é possível mandar ao **Roteador3**, pois não estão conectados diretamente por nenhuma interface nem existe rota programada (Figura 16).

Para que todos os componentes do AS consigam conversar em si é necessário que eles tenham informações de todas as redes atingíveis em sua tabela de rotas. Para isso pode-se utilizar rotas estáticas ou rotas dinâmicas.

No Capítulo 5 será apresentado como configurar rotas dinâmicas nos roteadores

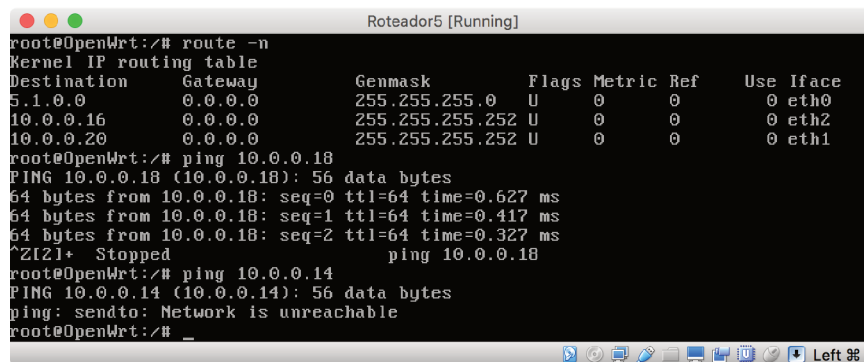


```

root@OpenWrt:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
3.1.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@OpenWrt:~# ping 3.1.0.254
PING 3.1.0.254 (3.1.0.254): 56 data bytes
64 bytes from 3.1.0.254: seq=0 ttl=64 time=0.527 ms
64 bytes from 3.1.0.254: seq=1 ttl=64 time=0.358 ms
64 bytes from 3.1.0.254: seq=2 ttl=64 time=0.311 ms
64 bytes from 3.1.0.254: seq=3 ttl=64 time=0.469 ms
^Z[21~ Stopped ping 3.1.0.254
root@OpenWrt:~# ping 2.1.0.254
PING 2.1.0.254 (2.1.0.254): 56 data bytes
ping: sendto: Network is unreachable
root@OpenWrt:~#

```

Figura 15 – OpenWrt - Teste de Conectividade: Host3 para Roteador2



```

root@OpenWrt:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
5.1.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.0.16 0.0.0.0 255.255.255.252 U 0 0 0 eth2
10.0.0.20 0.0.0.0 255.255.255.252 U 0 0 0 eth1
root@OpenWrt:~# ping 10.0.0.18
PING 10.0.0.18 (10.0.0.18): 56 data bytes
64 bytes from 10.0.0.18: seq=0 ttl=64 time=0.627 ms
64 bytes from 10.0.0.18: seq=1 ttl=64 time=0.417 ms
64 bytes from 10.0.0.18: seq=2 ttl=64 time=0.327 ms
^Z[21~ Stopped ping 10.0.0.18
root@OpenWrt:~# ping 10.0.0.14
PING 10.0.0.14 (10.0.0.14): 56 data bytes
ping: sendto: Network is unreachable
root@OpenWrt:~#

```

Figura 16 – OpenWrt - Teste de Conectividade: Roteador5 para Roteador3

utilizando o protocolo de roteamento OSPF, uma vez que estes precisaram trocar informações das tabelas de rotas para conseguir transmitir pacotes para todas as redes do AS. Será apresentado também como configurar rotas estáticas nos *hosts*, pois como os mesmos estão na borda da rede e só vão estar conectados aos roteadores não precisam implementar algoritmo de roteamento.

5 Configuração das rotas

Neste capítulo serão apresentados os passos necessários na configuração do *Quagga* para utilizar o protocolo de roteamento dinâmico OSPF nos roteadores e os passos necessários para configuração das rotas estáticas nos *hosts*, divididos nas seguintes etapas:

- configuração das rotas dinâmicas: *Quagga: Zebra*;
- configuração das rotas dinâmicas: *Quagga: OSPFd*.
- configuração das rotas estáticas.

Apesar do *Quagga* estar instalado em todos os componentes, ele será configurado somente nos roteadores, pois são eles que devem utilizar o roteamento dinâmico. Os *hosts* utilizarão uma rota estática que será configurada manualmente.

5.1 Configuração das rotas dinâmicas: *Quagga: Zebra*

Como explicado na Seção 2.5, o *Zebra* é o gerenciador de protocolos, então deve ser configurado primeiro. As configurações, tanto do *Zebra* quanto do *OSPFd*, são feitas através de um terminal VTY. Na Tabela 11 estão representados todos os serviços e daemons suportados pelo *Quagga* e as respectivas portas para conexão. Antes de conectar

Nome do serviço/daemon	Porta	Tipo	Descrição do serviço/daemon
zebrasrv	2600	TCP	zebra service
zebra	2601	TCP	zebra vty
ripd	2602	TCP	RIPd vty
ripngd	2603	TCP	RIPngd vty
ospfd	2604	TCP	OSPFd vty
bgpd	2605	TCP	BGPd vty
ospf6d	2606	TCP	OSPF6d vty
ospfapi	2607	TCP	ospfapi
isisd	2608	TCP	ISISd vty

Tabela 11 – Tabela de Portas VTY do Quagga

no VTY, é recomendável reinicia-lo, uma vez que as interfaces de rede foram mudadas no capítulo anterior, através do comando:

```
> /etc/init.d/quagga restart
```

Após a reinicialização deve-se utilizar o comando `telnet` seguido de `localhost` e a porta VTY respectiva, conforme apresentado na Tabela 11, ficando da seguinte forma:

```
> telnet localhost 2601
```

Uma vez conectado ao VTY é necessário autenticar-se. A senha, por padrão, é `zebra`. Depois de inserir a senha e pressionar `Enter` o console apresentará a linha de comando do *Quagga* (Figura 17).

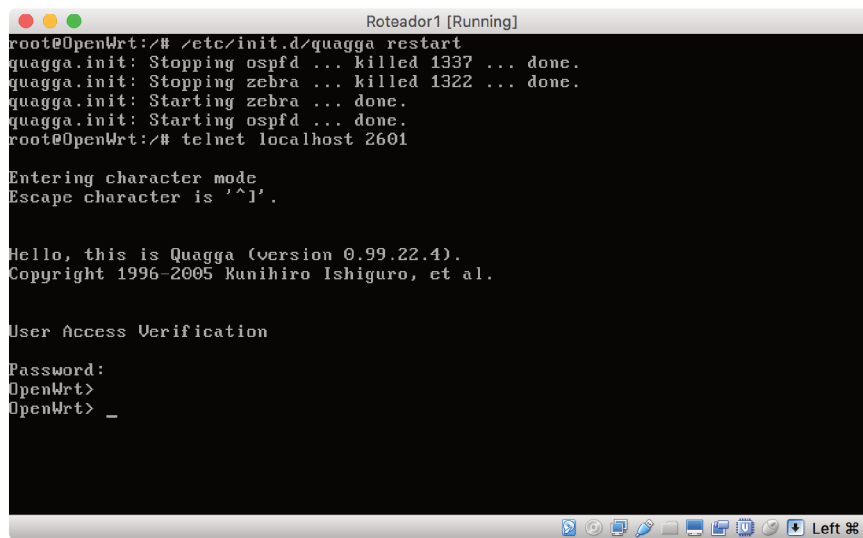


Figura 17 – Quagga - Zebra: Tela Inicial

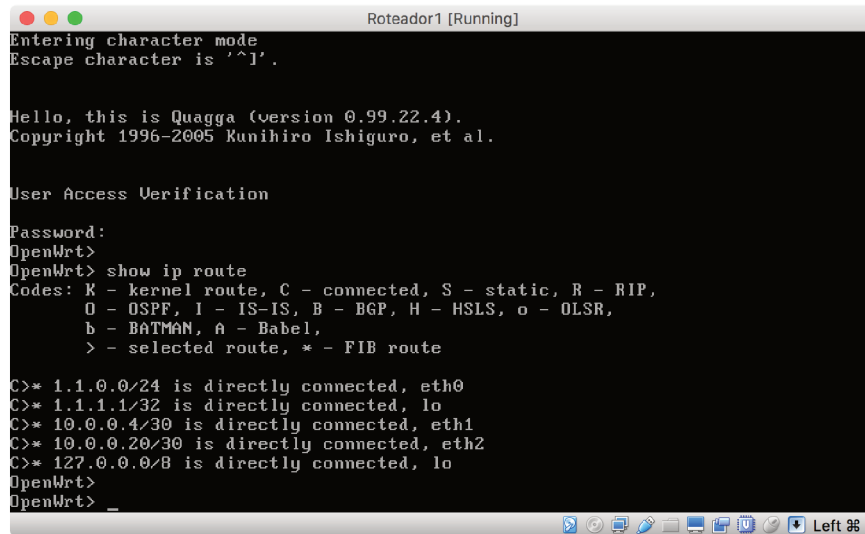
O *Zebra* oferece a apresentação de informações complementares sobre as rotas através do comando:

```
> show ip route
```

Como neste momento ainda não foi configurado o OSPF, na tabela só existe informações das rotas que estão conectadas diretamente no roteador (Figura 18). Para configurar os endereços de IP é necessário habilitar o terminal de configuração, através comandos:

```
> enable  
> configure terminal
```

Para cada uma das interfaces é necessário utilizar o comando `interface` seguido do nome da interface e o comando `ip address` seguido do número IP e da máscara. Após estes comandos, usar o comando `quit` para sair da configuração da interface e possibilitar a digitação das informações das outras interfaces. A sequência de comandos deve seguir a seguinte forma:



```
Roteador1 [Running]
Entering character mode
Escape character is '^1'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
OpenWrt>
OpenWrt> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, H - HSLs, o - OLSR,
       b - BATMAN, A - Babel,
       > - selected route, * - FIB route
C>* 1.1.0.0/24 is directly connected, eth0
C>* 1.1.1.1/32 is directly connected, lo
C>* 10.0.0.4/30 is directly connected, eth1
C>* 10.0.0.20/30 is directly connected, eth2
C>* 127.0.0.0/8 is directly connected, lo
OpenWrt>
OpenWrt> _
```

Figura 18 – Quagga - Zebra: Rotas do Roteador1 antes da Configuração

```
> interface ethx
> ip address a.b.c.d/m
> quit
```

Os valores de a.b.c.d/m devem ser substituídos pelos endereços de IP e máscara indicados nas tabelas de configuração de cada roteador e x pelo número da interface de rede. Após realizar todos esses passos o terminal deverá estar como na Figura 19. Para salvar as configurações e sair do terminal virtual, digitar os comandos seguintes, um por vez:

```
> write
> end
> quit
```

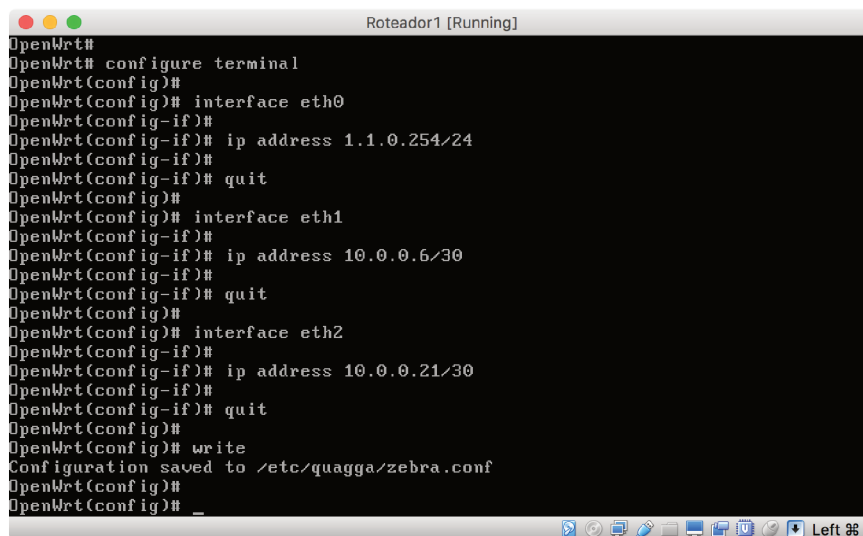
5.2 Configuração das rotas dinâmicas: *Quagga: OSPFd*

Após a configuração do *Zebra* deve-se fazer a configuração do *OSPFd*. A recomendação é que antes de entrar no VTY se execute novamente o reiniciamento do Quagga, pelo comando `/etc/init.d/quagga restart`.

Os passos para conexão ao VTY do OSPF são os mesmos do Zebra, trocando a porta VTY para a porta correspondente conforme a Tabela 11, ficando da seguinte forma:

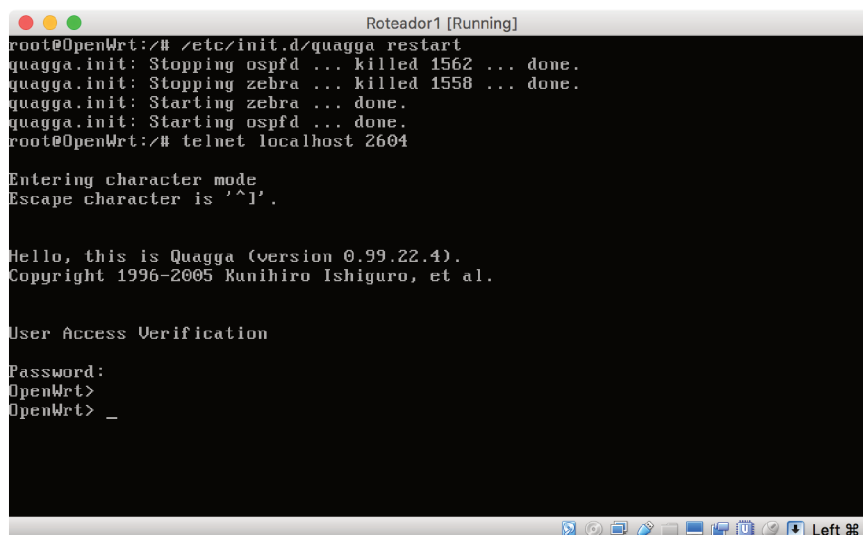
```
> telnet localhost 2604
```

A senha para entrar no VTY, por padrão, também é **zebra**. Após inserir a senha e pressionar **Enter** o console do *Quagga* será apresentado (Figura 20).

A terminal window titled 'Roteador1 [Running]' showing the configuration of the Zebra daemon. The user enters 'configure terminal' to enter configuration mode. Then, they configure three interfaces: 'eth0' with IP '1.1.0.254/24', 'eth1' with IP '10.0.0.6/30', and 'eth2' with IP '10.0.0.21/30'. After each interface configuration, they press 'quit'. Finally, they press 'write' to save the configuration to '/etc/quagga/zebra.conf'.

```
OpenWrt#  
OpenWrt# configure terminal  
OpenWrt(config)#  
OpenWrt(config)# interface eth0  
OpenWrt(config-if)#  
OpenWrt(config-if)# ip address 1.1.0.254/24  
OpenWrt(config-if)#  
OpenWrt(config-if)# quit  
OpenWrt(config)#  
OpenWrt(config)# interface eth1  
OpenWrt(config-if)#  
OpenWrt(config-if)# ip address 10.0.0.6/30  
OpenWrt(config-if)#  
OpenWrt(config-if)# quit  
OpenWrt(config)#  
OpenWrt(config)# interface eth2  
OpenWrt(config-if)#  
OpenWrt(config-if)# ip address 10.0.0.21/30  
OpenWrt(config-if)#  
OpenWrt(config-if)# quit  
OpenWrt(config)#  
OpenWrt(config)# write  
Configuration saved to /etc/quagga/zebra.conf  
OpenWrt(config)#  
OpenWrt(config)# _
```

Figura 19 – Quagga - Zebra: Configuração

A terminal window titled 'Roteador1 [Running]' showing the restart of the Quagga OSPFd daemon. The user enters '/etc/init.d/quagga restart', which outputs status messages for stopping and starting ospfd and zebra. Then, they enter 'telnet localhost 2604' to connect to the OSPF daemon. The connection is successful, and the user enters character mode. The screen displays the Quagga version (0.99.22.4), copyright information, and a password prompt.

```
root@OpenWrt:/# /etc/init.d/quagga restart  
quagga.init: Stopping ospfd ... killed 1562 ... done.  
quagga.init: Stopping zebra ... killed 1558 ... done.  
quagga.init: Starting zebra ... done.  
quagga.init: Starting ospfd ... done.  
root@OpenWrt:/# telnet localhost 2604  
  
Entering character mode  
Escape character is '^I'.  
  
Hello, this is Quagga (version 0.99.22.4).  
Copyright 1996-2005 Kunihiro Ishiguro, et al.  
  
User Access Verification  
  
Password:  
OpenWrt>  
OpenWrt> _
```

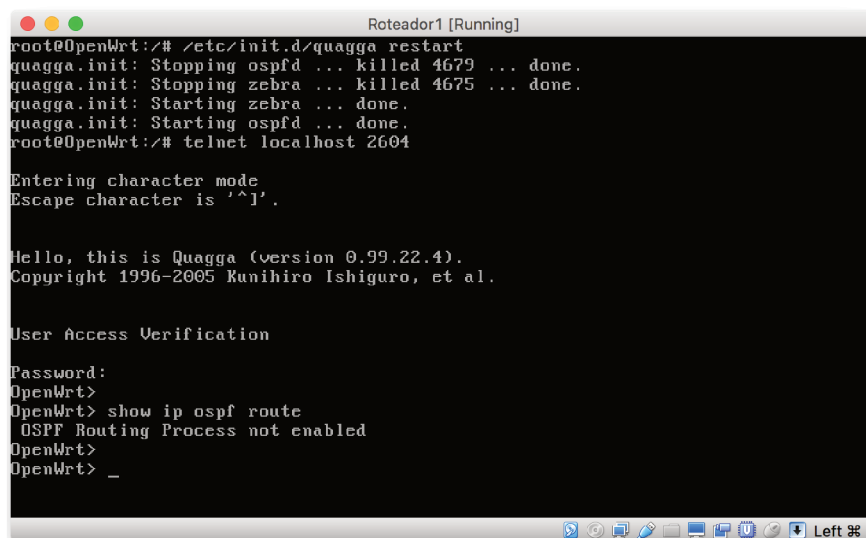
Figura 20 – Quagga - OSPFd: Tela Inicial

O terminal do *OSPFd* também contém informações sobre as rotas, através do comando:

```
> show ip ospf route
```

Como o OSPF ainda não foi configurado, a mensagem da Figura 21 é apresentada. A habilitação do terminal de configuração segue os mesmos passos da configuração do *Zebra*, porém com a adição da linha `router ospf` que ativará o protocolo de roteamento OSPF. O comando ficará da seguinte forma:

```
> enable  
> configure terminal  
> router ospf
```



```
root@OpenWrt:~# /etc/init.d/quagga restart
quagga.init: Stopping ospfd ... killed 4679 ... done.
quagga.init: Stopping zebra ... killed 4675 ... done.
quagga.init: Starting zebra ... done.
quagga.init: Starting ospfd ... done.
root@OpenWrt:~# telnet localhost 2604

Entering character mode
Escape character is '^J'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
OpenWrt>
OpenWrt> show ip ospf route
  OSPF Routing Process not enabled
OpenWrt>
OpenWrt> _
```

Figura 21 – Quagga - OSPFd: Rotas do Roteador1 antes da Configuração

Com o protocolo ativado, deve-se utilizar o comando `router-id a.b.c.d` substituindo os valores de `a.b.c.d` pelo endereço de loopback do roteador em configuração, conforme a Tabela 10. Após configurar o `router-id`, deve-se inserir as redes atingíveis pelo roteador, utilizando o comando:

```
> network e.f.g.h/m area x
```

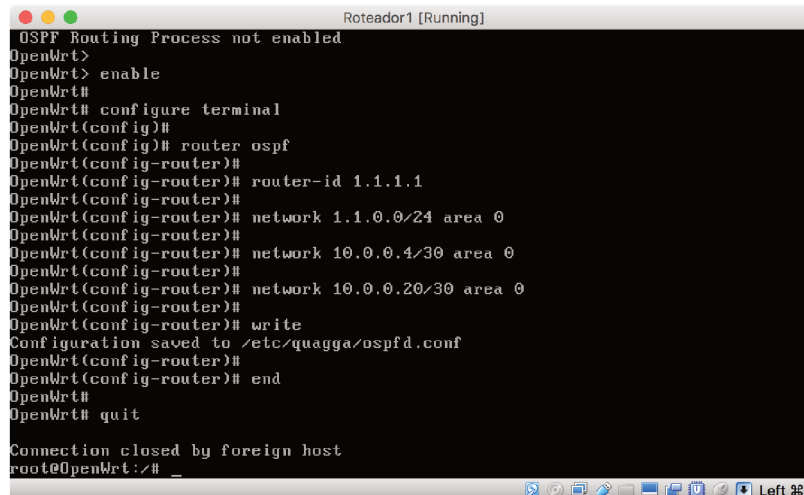
Os valores de `e.f.g.h` devem ser substituídos pelo endereço de IP da rede, o valor de `m` pela máscara de rede e o valor de `x` deve ser substituído pelo valor 0, que equivale ao backbone.

Uma vez inserido todas as redes, deve-se salvar o arquivo de configuração e sair do VTY (Figura 22), utilizando os comandos:

```
> write
> end
> quit
```

O comando `show ip ospf route`, executado novamente, mostra o resultado conforme a Figura 23. Como somente um roteador foi configurado (o Roteador1), a tabela apresenta somente os que estão diretamente conectados à ele. Na medida que o protocolo for ativado e configurado nos outros roteadores, esta tabela se atualizará dinamicamente. A Figura 24 mostra o comando sendo executado novamente no Roteador1 após a configuração do OSPF no Roteador2. Como é possível perceber, a partir deste momento o Roteador1 já consegue transmitir pacotes também para as redes ligadas ao Roteador2.

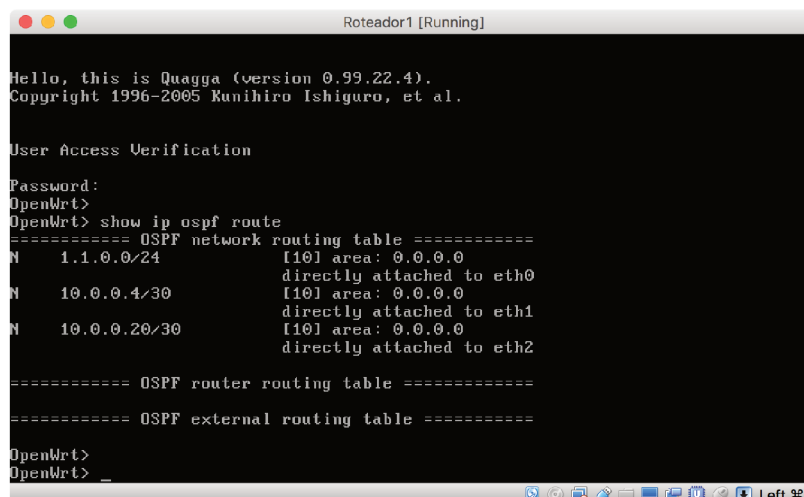
Se após a configuração de todos os roteadores alguns ainda não conseguirem trocar informações de rotas, a recomendação é que se desligue todos através do comando `halt` e inicie todos novamente.



```
Roteador1 [Running]
OSPF Routing Process not enabled
OpenWrt>
OpenWrt> enable
OpenWrt#
OpenWrt# configure terminal
OpenWrt(config)#
OpenWrt(config)# router ospf
OpenWrt(config-router)#
OpenWrt(config-router)# router-id 1.1.1.1
OpenWrt(config-router)#
OpenWrt(config-router)# network 1.1.0.0/24 area 0
OpenWrt(config-router)#
OpenWrt(config-router)# network 10.0.0.4/30 area 0
OpenWrt(config-router)#
OpenWrt(config-router)# network 10.0.0.20/30 area 0
OpenWrt(config-router)#
OpenWrt(config-router)# write
Configuration saved to /etc/quagga/ospfd.conf
OpenWrt(config-router)#
OpenWrt(config-router)# end
OpenWrt#
OpenWrt# quit

Connection closed by foreign host
root@OpenWrt:~#
```

Figura 22 – Quagga - OSPFd: Configuração do Protocolo no Roteador1



```
Roteador1 [Running]

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

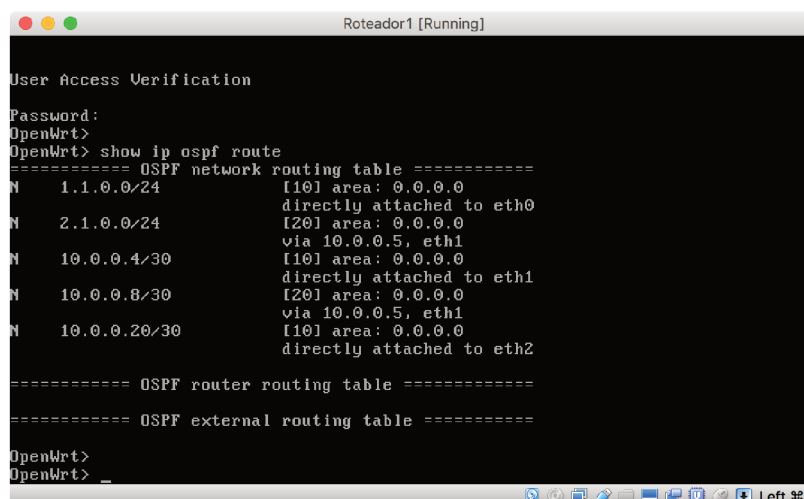
User Access Verification

Password:
OpenWrt>
OpenWrt> show ip ospf route
===== OSPF network routing table =====
N    1.1.0.0/24          [10] area: 0.0.0.0
      directly attached to eth0
N    10.0.0.4/30        [10] area: 0.0.0.0
      directly attached to eth1
N    10.0.0.20/30       [10] area: 0.0.0.0
      directly attached to eth2

===== OSPF router routing table =====
===== OSPF external routing table =====

OpenWrt>
OpenWrt> _
```

Figura 23 – Quagga - OSPFd: Rotas do Roteador1 após Configuração



```
Roteador1 [Running]

User Access Verification

Password:
OpenWrt>
OpenWrt> show ip ospf route
===== OSPF network routing table =====
N    1.1.0.0/24          [10] area: 0.0.0.0
      directly attached to eth0
N    2.1.0.0/24          [20] area: 0.0.0.0
      via 10.0.0.5, eth1
N    10.0.0.4/30         [10] area: 0.0.0.0
      directly attached to eth1
N    10.0.0.8/30         [20] area: 0.0.0.0
      via 10.0.0.5, eth1
N    10.0.0.20/30       [10] area: 0.0.0.0
      directly attached to eth2

===== OSPF router routing table =====
===== OSPF external routing table =====

OpenWrt>
OpenWrt> _
```

Figura 24 – Quagga - OSPFd: Rotas do Roteador1 com Atualização Dinâmica

5.3 Configuração das rotas estáticas

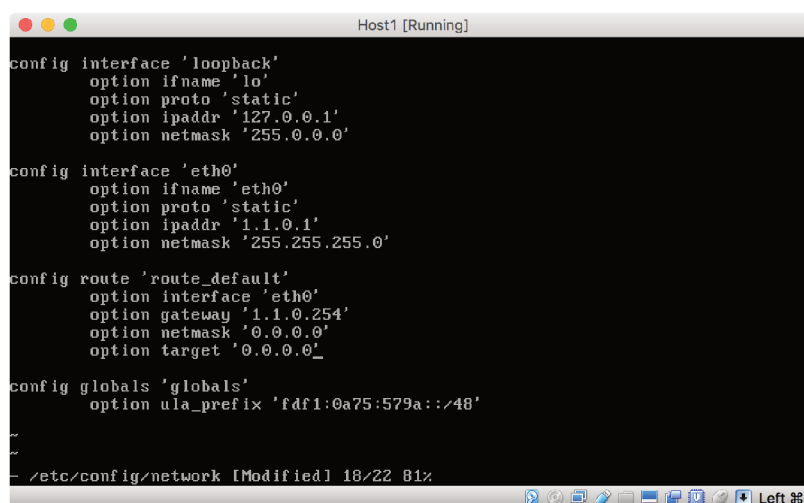
Para configurar uma rota estática é necessário abrir o arquivo de configuração de rede do *OpenWrt* (o mesmo editado na Seção 4.2) e inserir a rota manualmente, de acordo com a Tabela 12. A configuração de rota a ser adicionada no arquivo deve ser de acordo com o modelo:

```
config route 'nome da rota'
    option interface 'ethx'
    option gateway 'a.b.c.d'
    option netmask 'e.f.g.h'
    option target 'i.j.k.l'
```

Roteador	Interface	Gateway	Máscara	Target
Host1	Eth0	1.1.0.254	0.0.0.0	0.0.0.0
Host2	Eth0	2.1.0.254	0.0.0.0	0.0.0.0
Host3	Eth0	3.1.0.254	0.0.0.0	0.0.0.0
Host4	Eth0	4.1.0.254	0.0.0.0	0.0.0.0
Host5	Eth0	5.1.0.254	0.0.0.0	0.0.0.0

Tabela 12 – Tabela de Rota Padrão dos Hosts

O nome da rota deve ser substituído, por padronização, pelo nome `route_default`. Os valores de `a.b.c.d` devem ser substituídos pelos endereços de `gateway`, os valores de `e.f.g.h` devem ser substituídos pela máscara de rede e os valores de `i.j.k.l` devem ser substituídos pelos endereços de `target` indicados na Tabela 12 para os respectivos `hosts`. A Figura 25 representa a configuração de rota estática no `Host1`.



```
Host1 [Running]
config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config interface 'eth0'
    option ifname 'eth0'
    option proto 'static'
    option ipaddr '1.1.0.1'
    option netmask '255.255.255.0'

config route 'route_default'
    option interface 'eth0'
    option gateway '1.1.0.254'
    option netmask '0.0.0.0'
    option target '0.0.0.0'

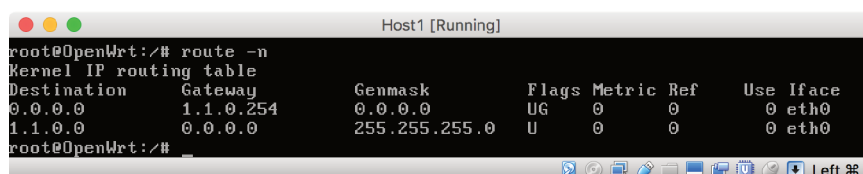
config globals 'globals'
    option ula_prefix 'fd11:0a75:579a::/48'

~
~
~
- /etc/config/network [Modified] 18/22 81%
```

Figura 25 – OpenWrt - Configuração da Rota Estática no Host1

Após terminar de editar, sair do modo de edição, executar o comando `:wq` para salvar o arquivo e em seguida o comando `/etc/init.d/network restart` para reiniciar

as interfaces. Para verificar se a rota padrão foi incluída deve-se executar o comando `route -n`. O resultado será como na Figura 26.



The image shows a terminal window titled 'Host1 [Running]' with the following output for the command 'route -n':

```
root@OpenWrt:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 1.1.0.254 0.0.0.0 UG 0 0 0 eth0
1.1.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
```

Figura 26 – OpenWrt - Tabela de Rota do Host1 com Rota Estática

Para o exemplo deste AS todas as interfaces das rotas padrões dos *hosts* serão a *Eth0*, pois ela é a única interface de rede ativa através da qual eles vão se conectar com o restante do sistema. O endereço de **gateway** serão os endereços de IP dos roteadores que pertencem à mesma rede que a *Eth0* dos *hosts*. Os endereços da máscara de rede e do **target** serão todos 0, indicando que para todo e qualquer IP que não seja da rede que o *host* faz parte deve-se encaminhar os pacotes pela interface e **gateway** indicados na rota estática.

6 Captura de Tráfego e Resultados

Neste capítulo serão apresentados os passos necessários para a captura do tráfego de pacotes das interfaces de rede dos componentes do AS e os resultados da análise do comportamento do protocolo OSPF no sistema autônomo virtualizado, divididos nas seguintes etapas:

- vinculação do arquivo para receber dados dos pacotes;
- inicialização das VMs via terminal;
- abertura do arquivo via *Wireshark*;
- verificação das tabelas de rotas de todos os componentes do AS;
- verificação da eleição do DR e do BDR entre dois vizinhos;
- emulação de transmissão de pacotes entre componentes de redes distintas;
- emulação de falhas em roteadores e verificação do comportamento da rede;
- análise do tráfego de pacotes.

6.1 Vinculação do arquivo para receber dados dos pacotes

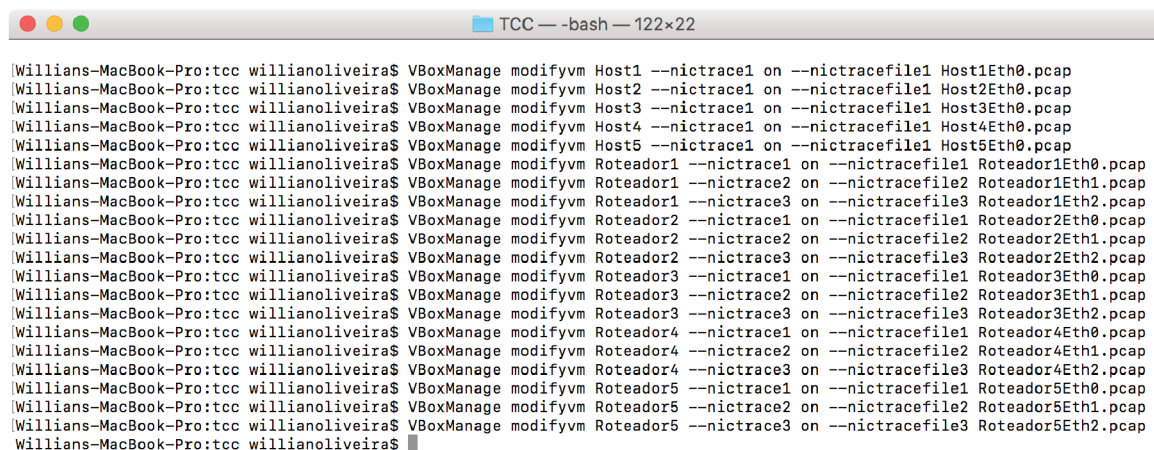
O *Wireshark* consegue capturar dados dos pacotes em tempo real quando esses pacotes estão trafegando no adaptador físico da máquina. Como não é o caso deste projeto, já que o tráfego está entrando e saindo de dentro da máquina virtual, é necessário uma outra forma de capturar os pacotes. O *VirtualBox* oferece a possibilidade de se logar todos os pacotes de determinado adaptador de rede de alguma VM através do comando:

```
> VBoxManage modifyvm [nome da VM] --nictracex on --nictracefilex  
[nome do arquivo].pcap
```

O nome da VM deve ser substituído pelo nome dado à VM. O valor de x em *nictracex* e *nictracefilex* deve ser substituído pelo número correspondente à interface que deseja-se analisar, de acordo com a Tabela 13. O nome do arquivo pode ser um nome qualquer. Para padronização será utilizado o nome do componente seguido pelo nome da interface. Por exemplo, para se capturar os pacotes da interface **Eth2** do **Roteador2** será utilizado o comando **VBoxManage modifyvm Roteador2 -nictrace3 on -nictracefile3 Roteador2Eth2.pcap**. O resultado final após vincular o arquivo para todas as interfaces deve estar como na Figura 27.

NicTrace (VirtualBox)	NicTraceFile (VirtualBox)	Interface de Rede (OpenWrt)
nictrace1	nictracefile1	Eth0
nictrace2	nictracefile2	Eth1
nictrace3	nictracefile3	Eth2
nictrace4	nictracefile4	Eth3

Tabela 13 – Tabela de Mapeamento entre Nictrace e Interfaces de Rede



```

TCC — -bash — 122x22

Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Host1 --nictrace1 on --nictracefile1 Host1Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Host2 --nictrace1 on --nictracefile1 Host2Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Host3 --nictrace1 on --nictracefile1 Host3Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Host4 --nictrace1 on --nictracefile1 Host4Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Host5 --nictrace1 on --nictracefile1 Host5Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador1 --nictrace1 on --nictracefile1 Roteador1Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador1 --nictrace2 on --nictracefile2 Roteador1Eth1.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador1 --nictrace3 on --nictracefile3 Roteador1Eth2.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador2 --nictrace1 on --nictracefile1 Roteador2Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador2 --nictrace2 on --nictracefile2 Roteador2Eth1.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador2 --nictrace3 on --nictracefile3 Roteador2Eth2.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador3 --nictrace1 on --nictracefile1 Roteador3Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador3 --nictrace2 on --nictracefile2 Roteador3Eth1.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador3 --nictrace3 on --nictracefile3 Roteador3Eth2.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador4 --nictrace1 on --nictracefile1 Roteador4Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador4 --nictrace2 on --nictracefile2 Roteador4Eth1.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador4 --nictrace3 on --nictracefile3 Roteador4Eth2.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador5 --nictrace1 on --nictracefile1 Roteador5Eth0.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador5 --nictrace2 on --nictracefile2 Roteador5Eth1.pcap
Willians-MacBook-Pro:tcc willianoliveira$ VBoxManage modifyvm Roteador5 --nictrace3 on --nictracefile3 Roteador5Eth2.pcap
Willians-MacBook-Pro:tcc willianoliveira$

```

Figura 27 – Terminal - Vinculação dos Arquivos .pcap às Interfaces do VirtualBox

Mesmo após a execução do comando os arquivos não aparecerão na pasta escolhida para serem armazenados. Os arquivos serão criados somente quando a VM for iniciada via terminal.

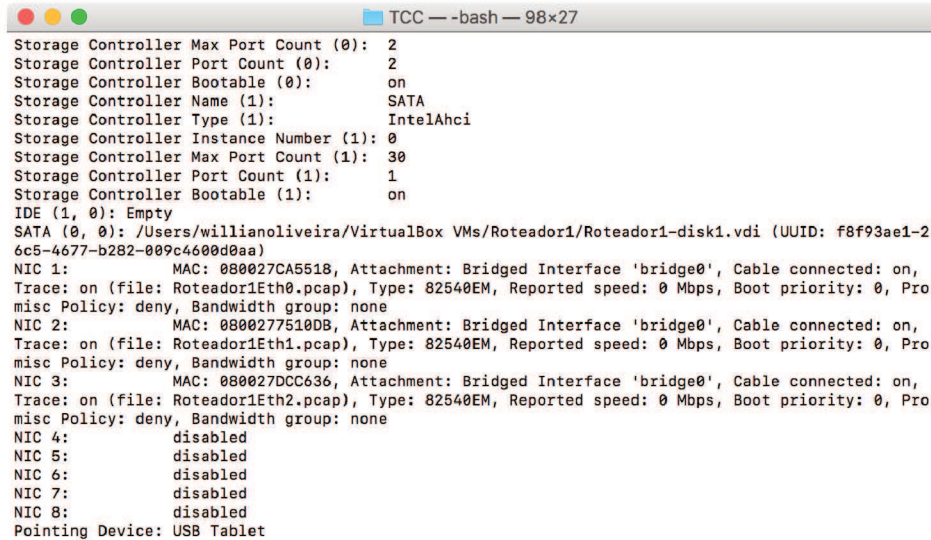
Para desativar a captura de pacotes deve-se utilizar o comando `VBoxManage modifyvm [nome da VM] --nictracex off`, substituindo o x em `nictracex` pelo número correspondente à interface. Para verificar se a captura está ativada pode-se utilizar o comando `VBoxManage showvminfo [nome da VM]` para mostrar as informações da VM. A opção `trace` do `nic` correspondente deve estar como `on` e em frente ao valor `file` deve estar o nome escolhido para captura do arquivo. Na Figura 28 é mostrado o trecho após o comando `VBoxManage showvminfo Roteador1`.

6.2 Inicialização das VM via terminal

Para que os dados sejam gravados nos arquivos é necessário fechar o software do *VirtualBox* e iniciar as VM através do comando no terminal, da seguinte forma:

```
> VirtualBox -startvm [nome da VM]
```

Após entrar com este comando a VM será inicializada e já estará gravando o tráfego de pacotes do arquivo .pcap correspondente. O terminal não devera ser fechado enquanto a VM estiver ligada.



```

TCC — -bash — 98x27
Storage Controller Max Port Count (0): 2
Storage Controller Port Count (0): 2
Storage Controller Bootable (0): on
Storage Controller Name (1): SATA
Storage Controller Type (1): IntelAhci
Storage Controller Instance Number (1): 0
Storage Controller Max Port Count (1): 30
Storage Controller Port Count (1): 1
Storage Controller Bootable (1): on
IDE (1, 0): Empty
SATA (0, 0): /Users/willianoliveira/VirtualBox VMs/Roteador1/Roteador1-disk1.vdi (UUID: f8f93ae1-26c5-4677-b282-009c4600d0aa)
NIC 1: MAC: 080027CA5518, Attachment: Bridged Interface 'bridge0', Cable connected: on,
Trace: on (file: Roteador1Eth0.pcap), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Pro
misc Policy: deny, Bandwidth group: none
NIC 2: MAC: 0800277510DB, Attachment: Bridged Interface 'bridge0', Cable connected: on,
Trace: on (file: Roteador1Eth1.pcap), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Pro
misc Policy: deny, Bandwidth group: none
NIC 3: MAC: 080027DCC636, Attachment: Bridged Interface 'bridge0', Cable connected: on,
Trace: on (file: Roteador1Eth2.pcap), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Pro
misc Policy: deny, Bandwidth group: none
NIC 4: disabled
NIC 5: disabled
NIC 6: disabled
NIC 7: disabled
NIC 8: disabled
Pointing Device: USB Tablet

```

Figura 28 – Terminal - Informação dos NICs da VM

6.3 Abertura do arquivo via *Wireshark*

Para se analisar os arquivos criados deve-se abri-los através do software *Wireshark*. A tela inicial do *Wireshark* é mostrada na Figura 29. Para se abrir um arquivo pcap, deve-se clicar no botão **Open** a **capture file** na barra de ferramenta e achar o arquivo criado na seção anterior. O arquivo será aberto com as informações até o momento de abertura, não sendo atualizadas em tempo real (Figura 30). Para mante-lo atualizado é necessário apertar o botão **Reload this file** na barra de ferramentas.

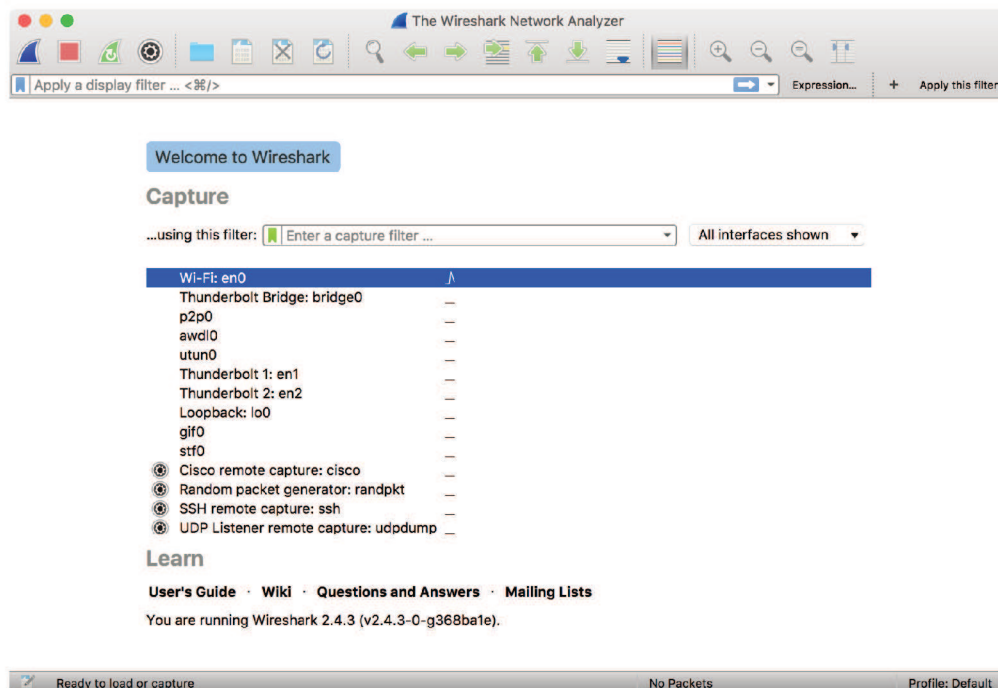


Figura 29 – Wireshark - Tela Inicial

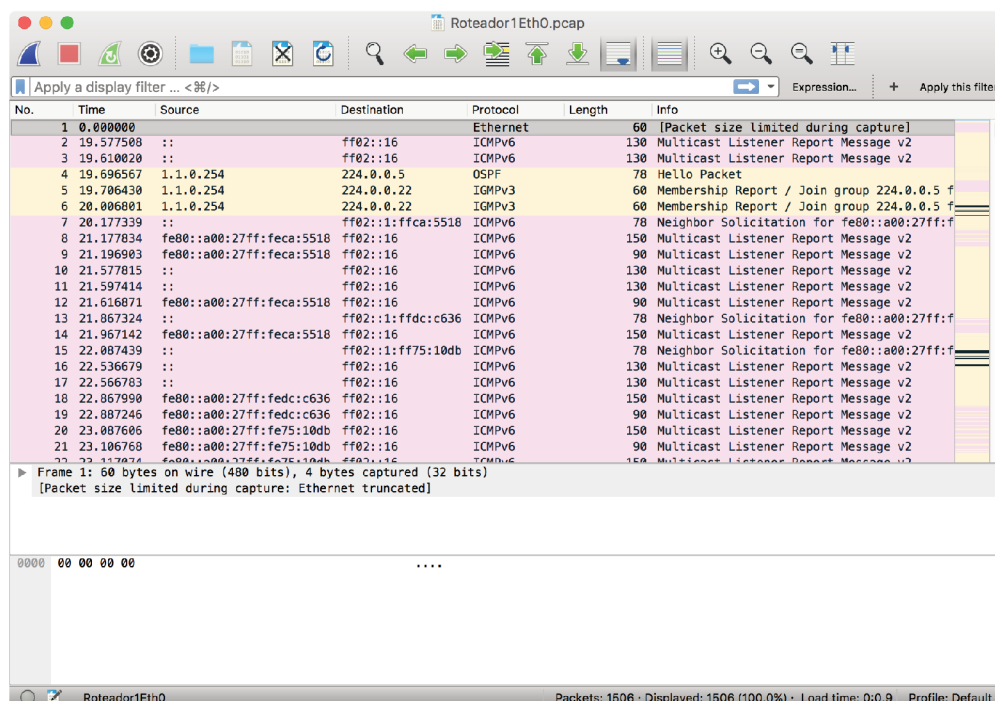


Figura 30 – Wireshark - Tela de Captura de Pacotes

Após completar todas as etapas descritas nos Capítulos 3, 4, 5 e 6 para todos roteadores, *hosts*, interfaces e adaptadores de redes correspondentes, a configuração de todo o AS estará concluída e pronta para ser trabalhada.

6.4 Verificação das tabelas de rotas

O AS do projeto é composto por dois tipos de componentes que operam de formas distintas. O primeiro tipo é composto pelos *hosts*, que estão na borda da rede e possuem somente uma rota de saída definida estaticamente na Seção 5.3.

O segundo tipo são os roteadores que formam o anel de comunicação interno do AS e que possuem rotas dinâmicas que são definidas através do OSPF configurado nas Seções 5.1 e 5.2. Desta forma, esses tipos devem ser analisados separadamente.

6.4.1 Rotas nos *hosts*

As informações das Tabelas 14, 15, 16, 17 e 18 foram extraídas através do comando `route -n` em cada um dos *hosts* que compõem o AS. Todos os *hosts* possuem duas entradas em comum na tabela de rotas. A entrada que possui somente a *flag* U indica que ela está ligada (U - *up*) e que é a rota para a rede definida na interface do *host*. Ela é adicionada automaticamente ao configurar a interface e assinala que os pacotes para um IP desta rede sempre sairão por essa interface. A distância é 0 porque não há cálculo para um IP ligado diretamente a ele e a interface configurada em todos os *hosts* foi a *Eth0*.

Destino	Gateway	Máscara	Flag	Distância	Interface
0.0.0.0	1.1.0.254	0.0.0.0	UG	0	Eth0
1.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0

Tabela 14 – Tabela de Rotas do Host1

Destino	Gateway	Máscara	Flag	Distância	Interface
0.0.0.0	2.1.0.254	0.0.0.0	UG	0	Eth0
2.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0

Tabela 15 – Tabela de Rotas do Host2

Destino	Gateway	Máscara	Flag	Distância	Interface
0.0.0.0	3.1.0.254	0.0.0.0	UG	0	Eth0
3.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0

Tabela 16 – Tabela de Rotas do Host3

Destino	Gateway	Máscara	Flag	Distância	Interface
0.0.0.0	4.1.0.254	0.0.0.0	UG	0	Eth0
4.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0

Tabela 17 – Tabela de Rotas do Host4

Destino	Gateway	Máscara	Flag	Distância	Interface
0.0.0.0	5.1.0.254	0.0.0.0	UG	0	Eth0
5.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0

Tabela 18 – Tabela de Rotas do Host5

A outra entrada, identificada pela *flag* UG, além de indicar que ela está ligada também mostra que ela sempre sai pelo *gateway* (G). Neste caso, essa é a rota padrão que foi definida manualmente. Os destinos e máscaras, que estão com os valores 0.0.0.0, assinalam que os pacotes sempre irão sair através do *gateway* definido para qualquer IP de destino que não esteja dentro da rede definida na interface. Cada *host* foi configurado para que o *gateway* seja a interface do roteador respectivo ligado em sua rede. A distância está em 0 porque não há cálculo para a rota definida e a interface em todos é a Eth0.

Um detalhe do projeto é que todos os *hosts* só possuem uma interface, então para todo e qualquer IP os pacotes sempre sairão por essa interface. Caso essa comunicação deixe de existir o *host* ficará incomunicável.

6.4.2 Rotas nos roteadores

As informações das Tabelas 19, 20, 21, 22 e 23 foram extraídas através do comando `route -n` em cada um dos roteadores que compõem o AS, após a configuração do OSPF em todos eles. Todos os roteadores possuem dez entradas em comum na tabela de rotas. As primeiras cinco entradas são as rotas para as redes ligadas aos *hosts* e as últimas cinco entradas são as rotas para as redes ligadas aos outros roteadores. As entradas que possuem somente a *flag* U indicam que elas estão ligadas e, assim como explicado anteriormente, seguem os mesmos conceitos para as rotas de redes definidas nas interfaces dos *hosts*. Como em cada roteador foram configuradas 3 interfaces (2 entre outros roteadores e 1 entre um *host*), cada tabela terá no mínimo 3 entradas deste tipo.

Destino	Gateway	Máscara	Flag	Distância	Interface
1.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0
2.1.0.0	10.0.0.5	255.255.255.0	UG	20	Eth1
3.1.0.0	10.0.0.5	255.255.255.0	UG	30	Eth1
4.1.0.0	10.0.0.22	255.255.255.0	UG	30	Eth2
5.1.0.0	10.0.0.22	255.255.255.0	UG	20	Eth2
10.0.0.4	0.0.0.0	255.255.255.252	U	0	Eth1
10.0.0.8	10.0.0.5	255.255.255.252	UG	20	Eth1
10.0.0.12	10.0.0.5	255.255.255.252	UG	30	Eth1
10.0.0.16	10.0.0.22	255.255.255.252	UG	20	Eth2
10.1.0.20	0.0.0.0	255.255.255.252	U	0	Eth2

Tabela 19 – Tabela de Rotas do Roteador1

Destino	Gateway	Máscara	Flag	Distância	Interface
1.1.0.0	10.0.0.6	255.255.255.0	UG	20	Eth1
2.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0
3.1.0.0	10.0.0.9	255.255.255.0	UG	20	Eth2
4.1.0.0	10.0.0.9	255.255.255.0	UG	30	Eth2
5.1.0.0	10.0.0.6	255.255.255.0	UG	30	Eth1
10.0.0.4	0.0.0.0	255.255.255.252	U	0	Eth1
10.0.0.8	0.0.0.0	255.255.255.252	U	0	Eth2
10.0.0.12	10.0.0.9	255.255.255.252	UG	20	Eth2
10.0.0.16	10.0.0.6	255.255.255.252	UG	30	Eth1
10.1.0.20	10.0.0.6	255.255.255.252	UG	20	Eth1

Tabela 20 – Tabela de Rotas do Roteador2

A outra entrada, identificada pela *flag* UG, além de indicar que ela está ligada também mostra que ela sempre sai pelo *gateway*. Diferentemente da entrada UG na tabela dos *host* estas entradas foram atualizadas pelo OSPF e indicam as rotas dinâmicas para as outras redes atingíveis através da ligação com os outros roteadores. Os destinos e máscaras

Destino	Gateway	Máscara	Flag	Distância	Interface
1.1.0.0	10.0.0.10	255.255.255.0	UG	30	Eth2
2.1.0.0	10.0.0.10	255.255.255.0	UG	20	Eth2
3.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0
4.1.0.0	10.0.0.13	255.255.255.0	UG	20	Eth1
5.1.0.0	10.0.0.13	255.255.255.0	UG	30	Eth1
10.0.0.4	10.0.0.10	255.255.255.252	UG	20	Eth2
10.0.0.8	0.0.0.0	255.255.255.252	U	0	Eth2
10.0.0.12	0.0.0.0	255.255.255.252	U	0	Eth1
10.0.0.16	10.0.0.13	255.255.255.252	UG	20	Eth1
10.1.0.20	10.0.0.13	255.255.255.252	UG	30	Eth1

Tabela 21 – Tabela de Rotas do Roteador3

Destino	Gateway	Máscara	Flag	Distância	Interface
1.1.0.0	10.0.0.17	255.255.255.0	UG	30	Eth2
2.1.0.0	10.0.0.14	255.255.255.0	UG	30	Eth1
3.1.0.0	10.0.0.14	255.255.255.0	UG	20	Eth1
4.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0
5.1.0.0	10.0.0.17	255.255.255.0	UG	20	Eth2
10.0.0.4	10.0.0.14	255.255.255.252	UG	30	Eth1
10.0.0.8	10.0.0.14	255.255.255.252	UG	20	Eth1
10.0.0.12	0.0.0.0	255.255.255.252	U	0	Eth1
10.0.0.16	0.0.0.0	255.255.255.252	U	0	Eth2
10.1.0.20	10.0.0.17	255.255.255.252	UG	20	Eth2

Tabela 22 – Tabela de Rotas do Roteador4

Destino	Gateway	Máscara	Flag	Distância	Interface
1.1.0.0	10.0.0.21	255.255.255.0	UG	20	Eth1
2.1.0.0	10.0.0.21	255.255.255.0	UG	30	Eth1
3.1.0.0	10.0.0.18	255.255.255.0	UG	30	Eth2
4.1.0.0	10.0.0.18	255.255.255.0	UG	20	Eth2
5.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0
10.0.0.4	10.0.0.21	255.255.255.252	UG	20	Eth1
10.0.0.8	10.0.0.21	255.255.255.252	UG	30	Eth1
10.0.0.12	10.0.0.18	255.255.255.252	UG	20	Eth2
10.0.0.16	0.0.0.0	255.255.255.252	U	0	Eth2
10.1.0.20	0.0.0.0	255.255.255.252	U	0	Eth1

Tabela 23 – Tabela de Rotas do Roteador5

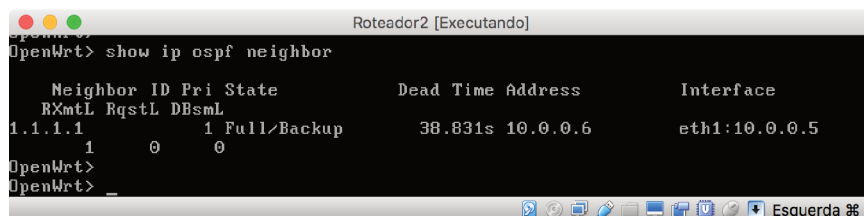
possuem os valores definidos na configuração da rede. O *gateway* e a interface assinalam por qual rota os pacotes irão sair quando o endereço de IP estiver no domínio daquele destino. A coluna distância indica a distância para alcançar a rede determinada. Para saber a quantidade de saltos basta dividi-la por 10.

Esta tabela é atualizada dinamicamente pelo protocolo de roteamento sempre que há mudanças na topologia da rede, como será visto nas próximas etapas.

6.5 Verificação da eleição do DR e do BDR

Como no AS proposto pelo projeto um roteador possui no máximo 2 vizinhos, será verificado o processo de eleição após o ligamento do Roteador2, seguido pelo ligamento do Roteador5 e por último o Roteador1. O comando `show ip ospf neighbor` mostra as informações dos roteadores vizinhos e o estado dele destes vizinhos em relação ao atual.

Na Figura 31, que mostra os vizinhos do Roteador2, é possível ver que o estado do vizinho Roteador1 é de BDR, pois na hora da eleição o Roteador2 tinha o maior *router-id*. Da mesma forma acontece na Figura 32, que mostra os vizinhos do Roteador5. O Roteador1 também é o BDR do Roteador5, pois o segundo também possui no maior *router-id*. Já na Figura 33, onde é mostrado os vizinhos do Roteador1, é possível perceber que ambos roteadores, Roteador2 e Roteador5 foram eleitos os DR. Isto é explicado pois a eleição se dá entre vizinhos e o Roteador2 não é vizinho do Roteador5.

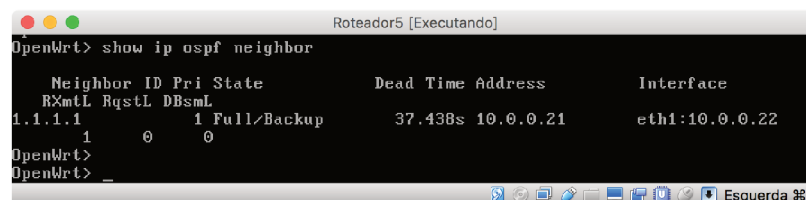


```

Roteador2 [Executando]
OpenWrt> show ip ospf neighbor
      Neighbor ID Pri State          Dead Time Address      Interface
      RXmtL RqstL DBsml
1.1.1.1      1 0      1 Full/Backup    38.831s 10.0.0.6      eth1:10.0.0.5
OpenWrt>
OpenWrt>

```

Figura 31 – OpenWrt - Eleição do DR e do BDR no Roteador2



```

Roteador5 [Executando]
OpenWrt> show ip ospf neighbor
      Neighbor ID Pri State          Dead Time Address      Interface
      RXmtL RqstL DBsml
1.1.1.1      1 0      1 Full/Backup    37.438s 10.0.0.21     eth1:10.0.0.22
OpenWrt>
OpenWrt>

```

Figura 32 – OpenWrt - Eleição do DR e do BDR no Roteador5

6.6 Emulação de transmissão de pacotes

Na seção 4.3 foi testado se o Host3 transmitia pacotes ao Roteador2 (Figura 15) e se o Roteador5 transmitia pacotes ao Roteador3 (Figura 16). Como eles não estavam

```

Roteador1 [Executando]
OpenWrt> show ip ospf neighbor
      Neighbor ID Pri State          Dead Time Address        Interface
  RXmtL  RqstL DBsmL
2.2.2.2    1    0    1 Full/DR          39.879s 10.0.0.5         eth1:10.0.0.6
5.5.5.5    1    0    1 Full/DR          39.520s 10.0.0.22        eth2:10.0.0.21
OpenWrt>
OpenWrt> _

```

Figura 33 – OpenWrt - Eleição do DR e do BDR no Roteador5

na mesma rede e não possuíam informações de rotas para se comunicarem, a mensagem exibida ao utilizar o comando `ping` foi: *Network is unreachable* (Rede é inalcançável).

Neste momento, com as tabelas de rotas atualizadas, os componentes de redes distintas conseguem comunicar entre si. Através do comando `ping` e utilizando o mesmo exemplo, a mensagem de rede inalcançável não é mais exibida, mas sim a resposta dos pacotes transmitidos como mostra a Figura 34 para a transmissão de pacotes do Host3 para o Roteador2 e a Figura 35 para a transmissão de pacotes do Roteador3 para o Roteador5.

```

Host3 [Running]
root@OpenWrt:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 3.1.0.254 0.0.0.0 UG 0 0 0 eth0
3.1.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@OpenWrt:~# ping 2.1.0.254
PING 2.1.0.254 (2.1.0.254): 56 data bytes
64 bytes from 2.1.0.254: seq=0 ttl=63 time=0.844 ms
64 bytes from 2.1.0.254: seq=1 ttl=63 time=0.661 ms
64 bytes from 2.1.0.254: seq=2 ttl=63 time=0.726 ms
64 bytes from 2.1.0.254: seq=3 ttl=63 time=0.749 ms
^Z[31* Stopped ping 2.1.0.254
root@OpenWrt:~# _

```

Figura 34 – OpenWrt - Ping Host3 para Roteador2

```

Roteador5 [Running]
root@OpenWrt:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
1.1.0.0 10.0.0.21 255.255.255.0 UG 20 0 0 eth1
2.1.0.0 10.0.0.21 255.255.255.0 UG 30 0 0 eth1
3.1.0.0 10.0.0.18 255.255.255.0 UG 30 0 0 eth2
4.1.0.0 10.0.0.18 255.255.255.0 UG 20 0 0 eth2
5.1.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.0.0.4 10.0.0.21 255.255.255.252 UG 20 0 0 eth1
10.0.0.8 10.0.0.21 255.255.255.252 UG 30 0 0 eth1
10.0.0.12 10.0.0.18 255.255.255.252 UG 20 0 0 eth2
10.0.0.16 0.0.0.0 255.255.255.252 U 0 0 0 eth2
10.0.0.20 0.0.0.0 255.255.255.252 U 0 0 0 eth1
root@OpenWrt:~# ping 10.0.0.14
PING 10.0.0.14 (10.0.0.14): 56 data bytes
64 bytes from 10.0.0.14: seq=0 ttl=63 time=0.990 ms
64 bytes from 10.0.0.14: seq=1 ttl=63 time=0.780 ms
64 bytes from 10.0.0.14: seq=2 ttl=63 time=0.868 ms
64 bytes from 10.0.0.14: seq=3 ttl=63 time=0.770 ms
^Z[11* Stopped ping 10.0.0.14
root@OpenWrt:~# _

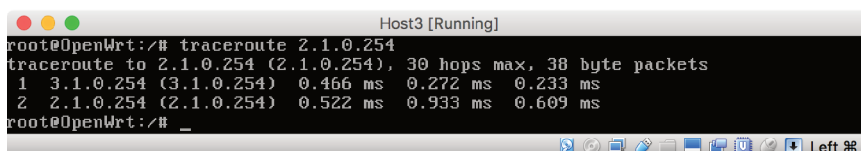
```

Figura 35 – OpenWrt - Ping: Roteador5 para Roteador3

Outro comando útil para verificar a transmissão de pacotes entre componentes é o `traceroute`. Este comando mostra a rota feita para que o pacote saia da origem e chegue ao destino. Através da rota mostrada na Figura 36 é possível visualizar que os pacotes

saíram do **Host3** e foram transmitidos pelo *gateway* com o **Roteador3** através da interface **Eth0**. Este último, como é conectado diretamente no **Roteador2** através da **Eth2** entregou os pacotes através da interface que se comunicam.

Já na rota mostrada na Figura 37 é possível visualizar que os pacotes saíram do **Roteador5** e foram transmitidos pela interface **Eth2** ligada ao **Roteador4**. Este último, como é conectado diretamente no **Roteador3** através da **Eth1** entregou os pacotes através da interface que se comunicam.

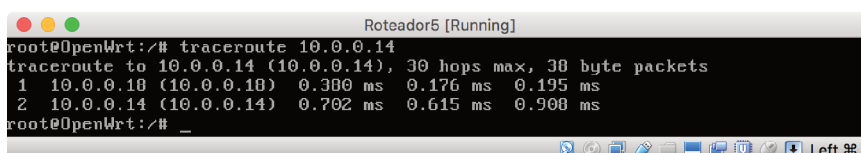


```

Host3 [Running]
root@OpenWrt:~# traceroute 2.1.0.254
traceroute to 2.1.0.254 (2.1.0.254), 30 hops max, 30 byte packets
 1  3.1.0.254 (3.1.0.254)  0.466 ms  0.272 ms  0.233 ms
 2  2.1.0.254 (2.1.0.254)  0.522 ms  0.933 ms  0.609 ms
root@OpenWrt:~#

```

Figura 36 – OpenWrt - Traceroute: Host3 para Roteador2



```

Roteador5 [Running]
root@OpenWrt:~# traceroute 10.0.0.14
traceroute to 10.0.0.14 (10.0.0.14), 30 hops max, 30 byte packets
 1  10.0.0.18 (10.0.0.18)  0.300 ms  0.176 ms  0.195 ms
 2  10.0.0.14 (10.0.0.14)  0.702 ms  0.615 ms  0.908 ms
root@OpenWrt:~#

```

Figura 37 – OpenWrt - Traceroute: Roteador5 para Roteador3

Nos dois casos acima a rota escolhida foi realmente a menor, pois em ambos casos existia outra rota (mais longa). No primeiro caso é a rota que sai do **Host3**, vai para o **Roteador3** e faz toda a volta pelo **Roteador4**, **Roteador5** e **Roteador1** até chegar no **Roteador2**. No segundo caso é a rota que sai do **Roteador5**, vai para o **Roteador1**, **Roteador2** e chega no **Roteador3**. Baseado nestas análises pode-se dizer que os resultados foram os esperados e que o protocolo foi configurado corretamente.

6.7 Emulação de falhas

Para testar o comportamento do AS em caso de falhas será utilizado o exemplo da transmissão de pacotes do **Host1** para o **Host4**. Existem duas rotas possíveis para este cenário. A primeira (e mais curta) é a rota que passa pelo **Roteador1**, **Roteador5** e **Roteador4** até chegar no **Host4**. Nesta rota são necessários 4 saltos. A segunda (que é a mais longa) é a rota que passa pelo **Roteador1**, **Roteador2**, **Roteador3**, **Roteador4** e então chega no **Host4**. Nesta rota são necessários 5 saltos. O algoritmo OSPF escolhe a primeira rota para transmitir os pacotes, como pode ser visto na Figura 38.

A partir das informações das Tabelas 19, 22 e 23 é possível verificar a razão desta rota ter sido a escolhida. Quando o pacote chega no **Roteador1** é olhada a rede de destino (neste caso a 4.1.0.0), sendo escolhida a interface **Eth2** que sai pelo *gateway* 10.0.0.22. Esta

```

Host1 [Running]
root@OpenWrt:~# traceroute 4.1.0.1
traceroute to 4.1.0.1 (4.1.0.1), 30 hops max, 38 byte packets
 1  1.1.0.254 (1.1.0.254)  0.411 ms  0.208 ms  0.185 ms
 2  10.0.0.22 (10.0.0.22)  0.565 ms  0.497 ms  0.474 ms
 3  10.0.0.18 (10.0.0.18)  0.911 ms  0.626 ms  0.757 ms
 4  4.1.0.1 (4.1.0.1)    1.606 ms  0.926 ms  0.949 ms
root@OpenWrt:~#

```

Figura 38 – OpenWrt - Traceroute: Host1 para Host4

interface é a que liga o Roteador1 ao Roteador5. Quando o pacote chega no Roteador5 acontece o mesmo processo. Ele é encaminhado pela interface Eth2 que sai pelo *gateway* 10.0.0.18, encaminhando o pacote para o Roteador4. Neste momento o Roteador4 encaminha o pacote para o Host4, pois tem uma conexão direta com a rede 4.1.0.0. O pacote chega ao seu destino e confirma a rota mostrada na Figura 38.

Porém, em caso de falha, o OSPF deve ser capaz de recalculando a melhor rota e continuar encaminhando os pacotes normalmente se existir um caminho alternativo. Para testar este cenário o Roteador5, responsável por ser a ponte entre o Roteador1 e Roteador4, foi desligado. Com isto, a melhor rota do Host1 para o Host4 passa a ser então a segunda e única rota. A Figura 39 mostra o comando `traceroute` executado após o desligamento do Roteador5. Como é possível perceber, o OSPF recalculou a rota e transmitiu os pacotes através da segunda rota.

```

Host1 [Running]
root@OpenWrt:~# traceroute 4.1.0.1
traceroute to 4.1.0.1 (4.1.0.1), 30 hops max, 38 byte packets
 1  1.1.0.254 (1.1.0.254)  0.516 ms  0.232 ms  0.234 ms
 2  10.0.0.5 (10.0.0.5)    0.717 ms  3.247 ms  0.741 ms
 3  10.0.0.9 (10.0.0.9)    0.838 ms  0.921 ms  0.740 ms
 4  10.0.0.13 (10.0.0.13)  1.272 ms  1.799 ms  1.270 ms
 5  4.1.0.1 (4.1.0.1)     1.641 ms  1.960 ms  3.130 ms
root@OpenWrt:~#

```

Figura 39 – OpenWrt - Traceroute: Host1 para Host4 após Recálculo de Rotas

As Tabelas 24 e 25 mostram o resultado do comando `route -n` nos roteadores Roteador1 e Roteador4 após o desligamento do Roteador5. As informações foram atualizadas com os valores e redes atingíveis recalculados pelo OSPF. Por estas novas tabelas é possível verificar que quando o pacote chega no Roteador1 a rede de destino 4.1.0.0 agora sai pela interface Eth1 do *gateway* 10.0.0.5. Esta interface liga o Roteador1 ao Roteador2. O Roteador2 continua encaminhando os pacotes da rede 4.1.0.0 pela interface Eth2 que sai pelo *gateway* 10.0.0.9. O pacote chega então ao Roteador3 que transmite para o Roteador4 pela interface Eth1 saindo pelo *gateway* 10.0.0.13. Ao chegar no Roteador4, que possui ligação direta com a rede, o pacote então é encaminhado para o Host4, encontrando seu destino e confirmando a rota mostrada na Figura 39.

É importante destacar que o Host5 neste momento está isolado da rede. Nenhum outro componente do AS consegue transmitir pacotes à ele e vice-versa. Como os pacotes

entram e saem somente pela rota padrão ligada pela interface **Eth0** ao **Roteador5**, este *host* não consegue mais se comunicar com o restante do AS.

Se o **Roteador5** for ligado novamente acontece todo o processo de reconhecimento da topologia da rede e troca de informações das tabelas de rotas com os outros componentes. O OSPF então atualiza as informações das tabelas para o que eram antes dele ser desligado e a primeira rota passa ser novamente a rota mais curta, sendo utilizada então para transmitir pacotes do **Host1** para o **Host4**. Este é o resultado esperado para o comportamento do protocolo em caso de falhas.

Destino	Gateway	Máscara	Flag	Distância	Interface
1.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0
2.1.0.0	10.0.0.5	255.255.255.0	UG	20	Eth1
3.1.0.0	10.0.0.5	255.255.255.0	UG	30	Eth1
4.1.0.0	10.0.0.5	255.255.255.0	UG	40	Eth1
10.0.0.4	0.0.0.0	255.255.255.252	U	0	Eth1
10.0.0.8	10.0.0.5	255.255.255.252	UG	20	Eth1
10.0.0.12	10.0.0.5	255.255.255.252	UG	30	Eth1
10.0.0.16	10.0.0.5	255.255.255.252	UG	40	Eth1
10.1.0.20	0.0.0.0	255.255.255.252	U	0	Eth2

Tabela 24 – Tabela de Rotas do Roteador1 após Recálculo de Rotas

Destino	Gateway	Máscara	Flag	Distância	Interface
1.1.0.0	10.0.0.14	255.255.255.0	UG	40	Eth1
2.1.0.0	10.0.0.14	255.255.255.0	UG	30	Eth1
3.1.0.0	10.0.0.14	255.255.255.0	UG	20	Eth1
4.1.0.0	0.0.0.0	255.255.255.0	U	0	Eth0
10.0.0.4	10.0.0.14	255.255.255.252	UG	30	Eth1
10.0.0.8	10.0.0.14	255.255.255.252	UG	20	Eth1
10.0.0.12	0.0.0.0	255.255.255.252	U	0	Eth1
10.0.0.16	0.0.0.0	255.255.255.252	U	0	Eth2
10.1.0.20	10.0.0.14	255.255.255.252	UG	40	Eth1

Tabela 25 – Tabela de Rotas do Roteador4 após Recálculo de Rotas

6.8 Análise do tráfego de pacotes

Para melhor entendimento do processo de roteamento é importante fazer a análise do tráfego de pacotes para verificar as mensagens do protocolo OSPF sendo enviadas e recebidas entre os componentes. Será utilizado como exemplo o arquivo gerado para a interface **Eth1** do **Roteador1**.

Na Figura 40 é mostrado a estrutura da mensagem de *Hello Packet* antes do Roteador2 ser ligado. Neste momento as mensagens que chegam são somente das interfaces do próprio roteador. Entre os campos do datagrama do IPv4 destaca-se o IP de origem e o IP de destino. Através do IP de origem é possível perceber que este pacote só é enviado e recebido das outras interfaces do Roteador1, pois nenhum outro roteador foi ligado ainda. O IP de destino mostra o endereço *multicast* do OSPF, ou seja, as mensagens para esse endereço são enviadas para todos os roteadores em um segmento do OSPF.

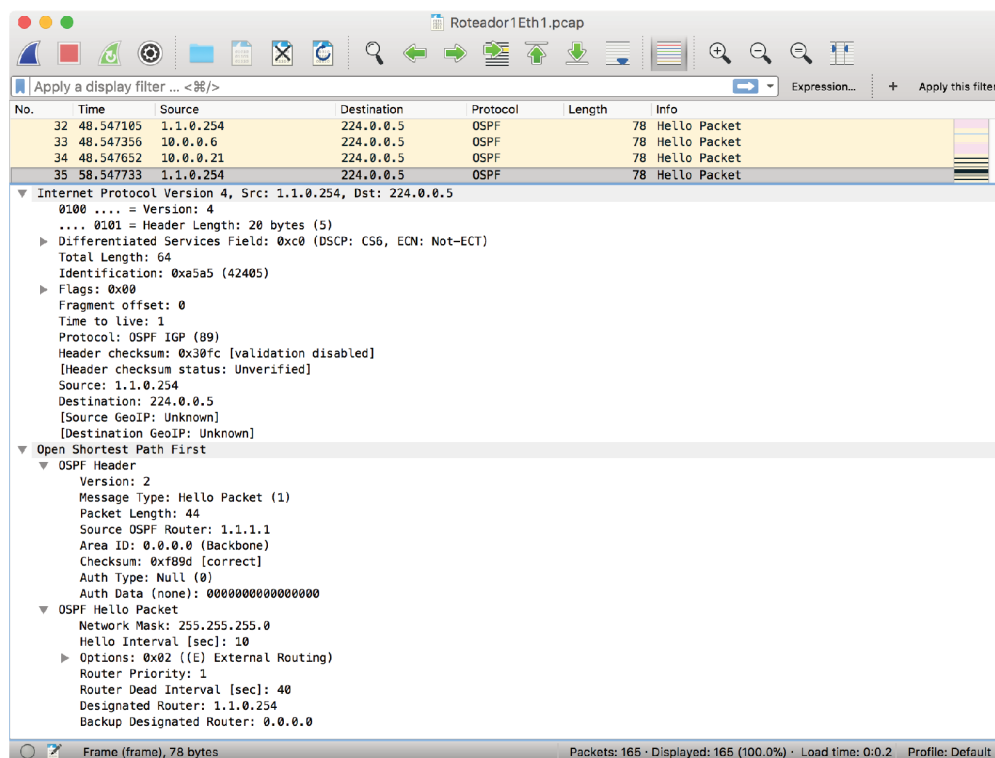


Figura 40 – Wireshark - Mensagem Hello Packet antes do ligamento do Roteador2

O Roteador1 começa a receber mensagens de *Hello Packet* das interfaces do Roteador2 após este ser ligado, conforme mostrado na Figura 41. O campo IP de origem no datagrama do IPv4 da mensagem destacada mostra que o pacote está vindo da interface Eth2 do Roteador2, mas que também chegaram das outras interfaces. Na mensagem de *Hello Packet* é possível notar que a eleição ainda não aconteceu, pois não há DR.

Os roteadores continuam a trocar várias mensagens, dentre elas as de *Database Description*, onde é possível ver troca de informações dos seus estados dos enlaces, como o tipo do enlace, o número de sequência o tipo da LSA e qual o roteador está fazendo o inundamento (Figura 42). Após este processo, o Roteador2 informa o Roteador1 sobre as rotas que são atingíveis por ele e vice-versa. As mensagens de *LS Request* (Figura 43), *LS Update* (Figura 44) e *LS Acknowledge* (Figura 45) também são trocadas, onde é possível ver as informações que o Roteador1 envia ao Roteador2 sobre os detalhes dos seus enlaces, inclusive as métricas. Estas mensagens também chegam nas outras interfaces, desta forma

o protocolo se mantém sempre atualizado.

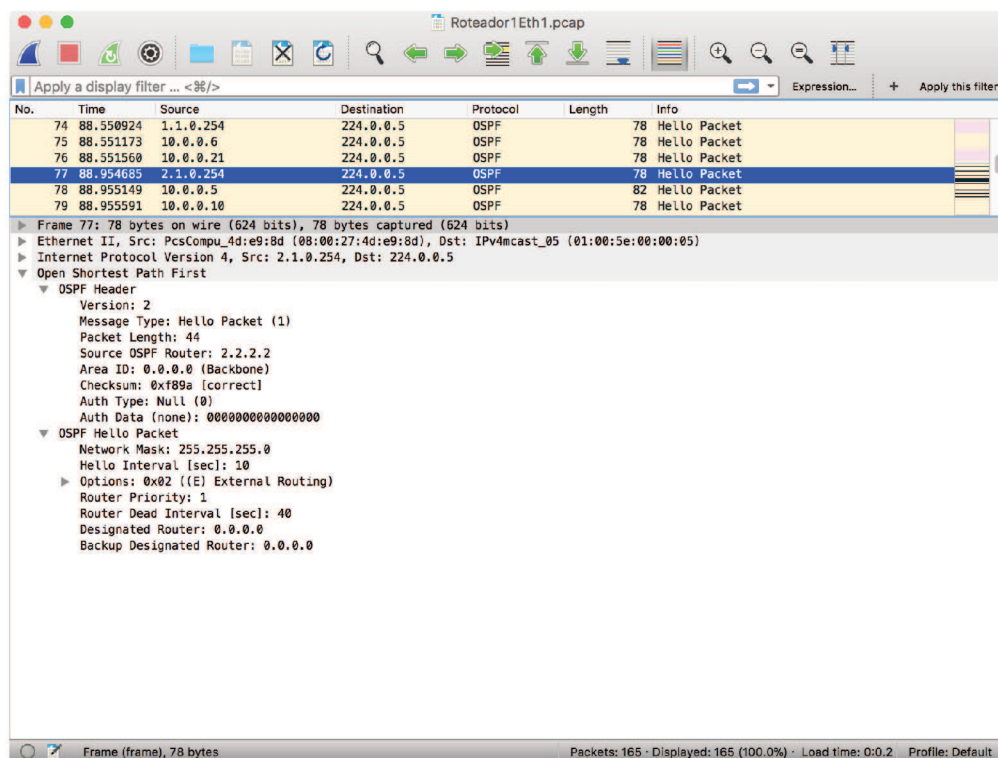


Figura 41 – Wireshark - Mensagem Hello Packet após o ligamento do Roteador2

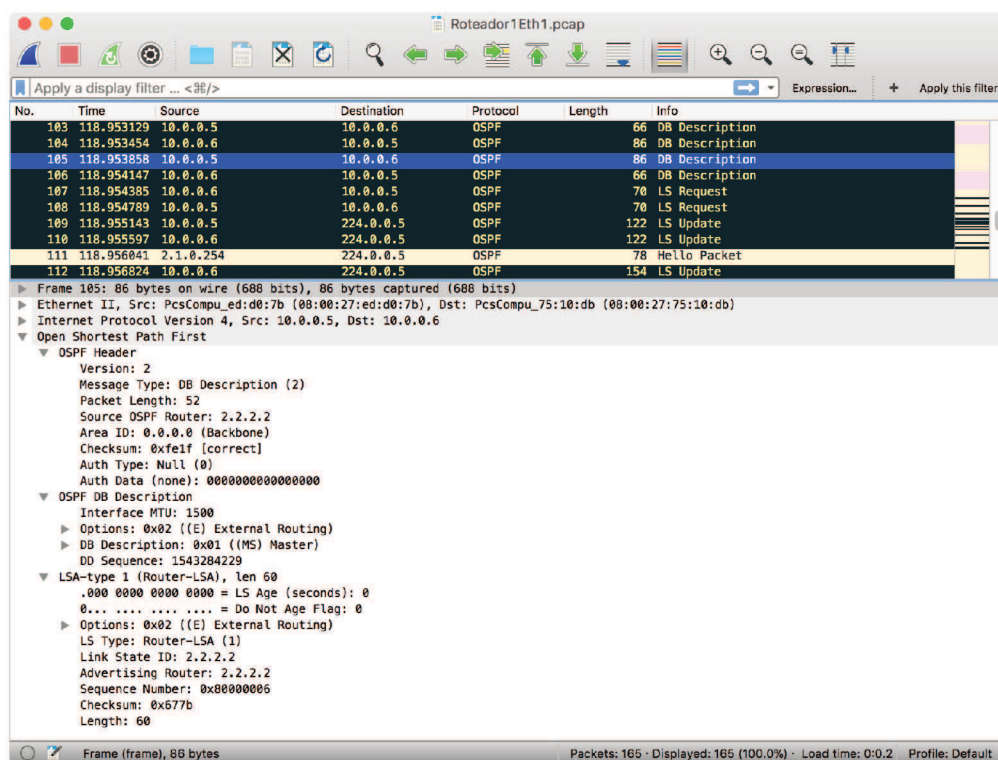


Figura 42 – Wireshark - Mensagem Database Description

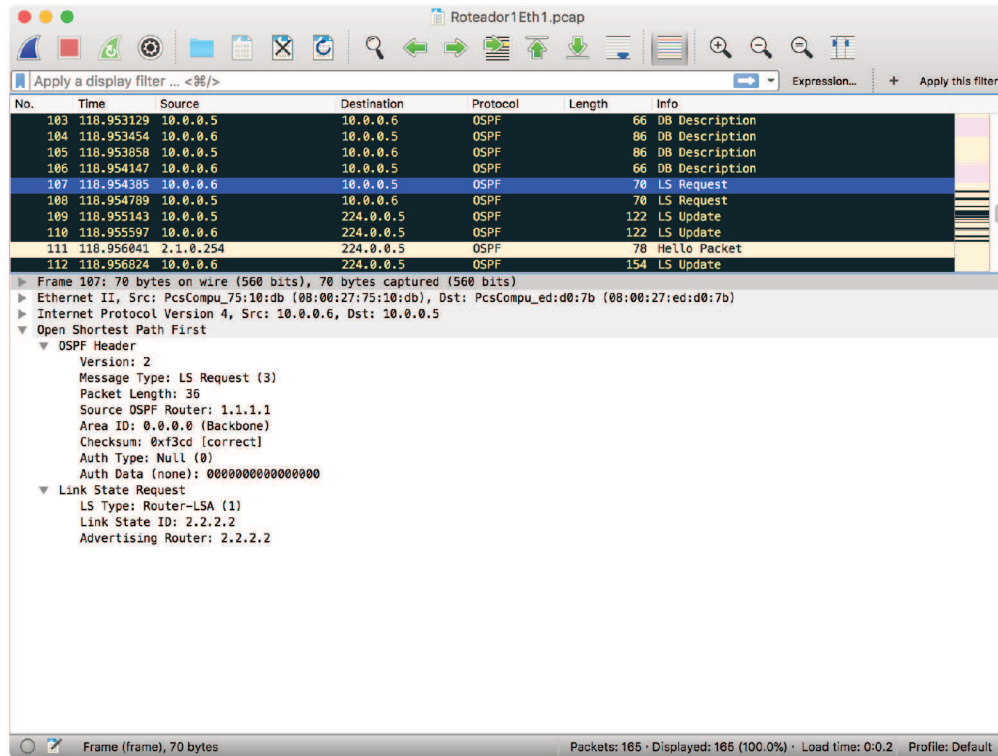


Figura 43 – Wireshark - Mensagem LS Request

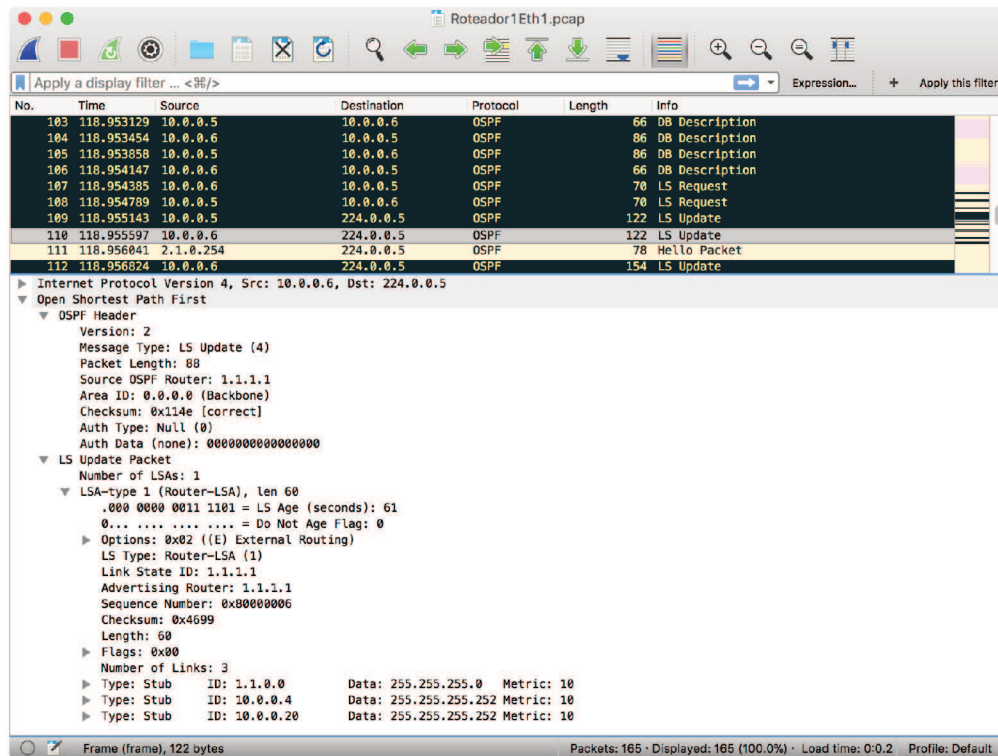


Figura 44 – Wireshark - Mensagem LS Update

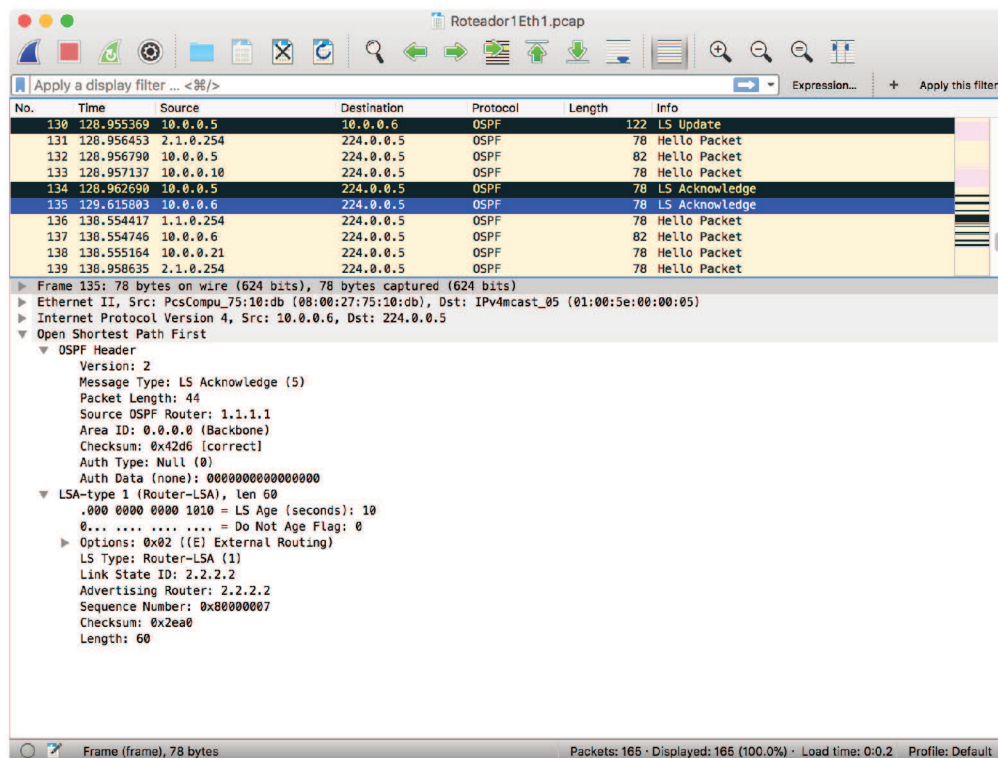


Figura 45 – Wireshark - Mensagem LS Acknowledge

Conclusão

O protocolo de roteamento dinâmico OSPF é muito importante e muito utilizado até hoje, pois soluciona problemas e limitações de protocolos anteriores, como o RIP. Por se tratar de um algoritmo que é utilizado em ambientes além do doméstico, pode existir uma dificuldade em ter um contato prático com os fundamentos aprendidos nas aulas de Redes de Computadores. Desta forma, é importante ter a possibilidade de poder fazer uma simulação sozinho e conseguir aprender sem necessitar de muitos recursos.

A suíte *Quagga* utilizada para gerenciar o OSPF no *Open Wrt* neste projeto também suporta vários outros protocolos. Portanto, também é possível criar novos AS virtualizados e configura-los para analisar estes protocolos.

Seguindo todos os passos de configurações e experimentos propostos neste projeto, até mesmo um leigo é capaz de fazer a montagem de um sistema autônomo no próprio computador utilizando componentes virtualizados para análise do Protocolo OSPF. Sendo assim, os objetivos propostos foram alcançados com sucesso.

Referências

DOYLE, J. *OSPF and IS-IS: Choosing an IGP for Large-Scale Networks*. [S.l.]: Addison-Wesley Professional, 2005. Citado na página 15.

POSTEL, J. (Ed.). *Internet Protocol - DARPA Internet Programm, Protocol Specification*. IETF, 1981. RFC 791. (Request for Comments). Disponível em: <<http://tools.ietf.org/html/rfc791>>. Citado na página 12.

POSTEL, J. (Ed.). *Transmission Control Protocol - DARPA Internet Programm, Protocol Specification*. IETF, 1981. RFC 793. (Request for Comments). Disponível em: <<http://tools.ietf.org/html/rfc793>>. Citado na página 13.

ISHIGURO, K. et al. *Quagga - A Routing Software Package for TCP/IP Networks*. 2017. Acesso em 3 nov. 2018. Disponível em: <<http://www.nongnu.org/quagga/docs/quagga.pdf>>. Citado na página 18.

KUROSE, J.; ROSS, K. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. 6. ed. São Paulo: Pearson, 2013. Citado 2 vezes nas páginas 13 e 14.

LAMPING, U.; SHARP, R.; WARNICKE, E. *Wireshark User's Guide for Wireshark 2.9.0*. 2018. Acesso em 3 nov. 2018. Disponível em: <<https://www.wireshark.org/download/docs/user-guide.pdf>>. Citado na página 19.

MOY, J. *OSPF Version 2*. IETF, 1998. RFC 2328. (Request for Comments). Disponível em: <<https://tools.ietf.org/html/rfc2328>>. Citado 2 vezes nas páginas 15 e 16.

MOY, J. T. *OSPF: Anatomy of an Internet Routing Protocol*. Boston, MA, USA: Addison-Wesley, 1998. ISBN 0201634724. Citado 2 vezes nas páginas 15 e 16.

ORACLE. *Oracle VM VirtualBox - User Manual*. 2018. Acesso em 3 nov. 2018. Disponível em: <<http://download.virtualbox.org/virtualbox/UserManual.pdf>>. Citado na página 16.

TADIMETY, P. R. *OSPF: A Network Routing Protocol*. New York: Apress, 2015. Citado 2 vezes nas páginas 15 e 16.

TANENBAUM, A. S.; WETHERALL, D. J. *Computer Networks*. [S.l.]: Prentice Hall, 2011. Citado 2 vezes nas páginas 12 e 14.