Emerson Gomes

Tolerância a falhas em VPNs BGP/MPLS/VRF usando DMVPNs

Proposta de TCC2 apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Eraldo Silveira e Silva , Dr.

Curso Superior de Tecnologia em Sistemas de Telecomunicações
Instituto Federal de Santa Catarina

São José – SC

julho / 2013

Monografia sob o título *"Tolerância a falhas em VPNs BGP/MPLS/VRF usando DMVPNs"*, defendida por Emerson Gomes e aprovada em 27 de agosto de 2014, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Eraldo Silveira e Silva, Dr.
Orientador

Prof. Marcelo Maia Sobral, Dr.
IFSC

Ricardo Dias da Cunha, Sr.
Gerência O&M DadosBanda Larga - OI

Sempre que te perguntarem se podes fazer um trabalho,
respondas que sim e te ponhas em seguida a aprender como se faz.
F. Roosevelt

Agradecimentos

Dedico meus sinceros agradecimentos à todos que me auxiliaram ao longo deste trabalho. Em especial à minha mãe Maura Lopes, por ter me incentivado e cobrado em todos estes anos de estudos, e ao meu orientador Dr. Eraldo Silveira, por sua paciência, dedicação e interesse neste trabalho.

Dedico também aos colegas e amigos de faculdade pelos belos anos juntos, e aos meus companheiros de trabalho que muito me auxiliaram com muita paciência e atenção para mais esta realização pessoal.

Resumo

Uma rede privada é algo comum em um contexto corporativo, em que se deseja interligar os *sites* de uma rede com segurança e privacidade dos dados. Essa abordagem é normalmente feita através de um serviço contratado junto à uma prestadora de serviços de internet. Porém, é possivel o uso da infrastrutura da rede Internet para prover este tipo de serviço, que pode futuramente, funcionar como uma alternativa de redundância à uma rede privada contratada.

Este trabalho visa avaliar a complexidade, escalabilidade e o uso de uma rede privada criada sem intervenção de uma prestadora de serviços de internet, fazendo uso de ferramentas e protocolos *opensource*. Além disso, é avaliada também a compatibilidade e complexidade destes softwares, bem como a possibilidade de uso desta rede privada funcionar em um contexto de redundância de uma VPN.

Abstract

A private network is common in a corporate context where you want to link the sites of a network with security and data privacy. This approach is usually done via a hired service from an iinternet services provider. However, it is possible to use the Internet infrastructure to provide this type of service, which can further function as an alternative to redundancy of a hired private network.

This study aims to evaluate the complexity, scalability and the use of a private network created without the intervention of a internet services provider using opensource and free softwares. Moreover, it is also evaluated the compatibility and complexity of these software, as well as the possibility of using private network in a redundancy context of a VPN.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas p. 15

1 Introdução p. 17

1.1 Motivação . . . . . p. 17

1.2 Objetivos do Trabalho . . . . . p. 18

1.3 Organização do texto . . . . . p. 19

2 Fundamentação Teórica p. 20

2.1 Conceitos em *Virtual Private Network* (VPN) . . . . . p. 20

2.2 VPN baseada em BGP/MPLS/VRF . . . . . p. 23

2.2.1 Protocolos e Mecanismos usados na VPN baseada em BGP/MPLS/VRF . . . . . p. 24

2.2.1.1 *Multiprotocol Label Switching* (MPLS) . . . . . p. 24

2.2.1.2 O protocolo *Border Gateway Protocol* (BGP) . . . . . p. 26

2.2.1.3 Características básicas . . . . . p. 26

2.2.1.4 Funcionamento . . . . . p. 26

2.2.1.5 *Virtual Routing and Forwarding* (VRF) . . . . . p. 28

2.2.2 O plano de controle da VPN BGP/MPLS/VRF . . . . . p. 28

2.2.3 O plano de dados BGP/MPLS/VRF . . . . . p. 30

2.3 A tecnologia *Dynamic Multipoint VPN* (DMVPN) . . . . . p. 31

2.3.1 Visão geral . . . . . p. 31

2.3.2 Protocolos e Mecanismos usados na DMVPN . . . . . p. 33

2.3.2.1 *Multipoint GRE* (mGRE) . . . . . p. 33

2.3.2.2 O *Next Hop Resolution Protocol* (NHRP) e a construção dinâmica . . . . . p. 35

2.3.2.3 *IP Security Protocol* (IPSec) . . . . . p. 36

2.3.3 *Routing Information Protocol* (RIP) . . . . . p. 36

2.3.4 Formação de uma rede DMVPN . . . . . p. 36

2.3.5 Soluções de DMVPN . . . . . p. 38

2.3.5.1 Soluções Comerciais . . . . . p. 38

2.3.5.2 Soluções Abertas . . . . . p. 38

3 Experimentos com Redes Privadas Dinâmicas (DMVPN) utilizando software livre p. 39

3.1 Introdução . . . . . p. 39

3.2 Ferramentas utilizadas . . . . . p. 39

3.3 Experimentos realizados . . . . . p. 41

3.3.1 Cenário 1: Teste básico de construção de túnel utilizando o OpenNHRP . . . . . p. 41

3.3.1.1 Objetivos do experimento . . . . . p. 41

3.3.2 Cenário 2: DMVPN com Linux . . . . . p. 43

3.3.2.1 Objetivos . . . . . p. 43

3.3.2.2 Configurações realizadas . . . . . p. 45

3.3.2.3 Testes e trocas de mensagens . . . . . p. 48

Etapa 1 . . . . . p. 48

Etapa 2 . . . . . p. 50

3.4 Conclusão . . . . . p. 52

4 Utilização de uma DMVPN atuando como redundância de uma VPN baseada em VRF p. 53

4.1 Introdução . . . . . p. 53

4.2 Redundância em VPNs . . . . . p. 53

4.3 Avaliação do RIP como protocolo de roteamento . . . . . p. 56

4.3.1 Objetivos do experimento . . . . . p. 56

4.3.2 Configurações e testes preliminares usando o multicast do mGRE p. 58

4.3.3 Configurações e testes usando o comando *neighbor* . . . . . p. 61

4.3.3.1 Configurações realizadas . . . . . p. 61

4.3.4 Testes realizados . . . . . p. 63

4.4 Discussão sobre os resultados . . . . . p. 64

4.5 Conclusão . . . . . p. 65

5 Conclusões p. 66

Apêndice A – Arquivos de configuração após último experimento p. 67

A.1 HUB . . . . . p. 67

A.1.1 OpenNHRP . . . . . p. 67

A.1.2 Quagga . . . . . p. 68

A.1.2.1 Zebra . . . . . p. 68

A.1.2.2 RIPD . . . . . p. 69

A.1.3 Ativar interface túnel no Linux . . . . . p. 69

A.2 SPOKE2 . . . . . p. 70

A.2.1 OpenNHRP . . . . . p. 70

A.2.2 Quagga . . . . . p. 71

A.2.2.1 Zebra . . . . . p. 71

A.2.2.2 RIPD . . . . . p. 72

A.2.3 Ativar interface túnel no Linux . . . . . p. 72

A.3 Configurações comuns à todos . . . . . p. 73

A.3.1 IPSecTools . . . . . p. 73

A.3.1.1 PSK . . . . . p. 73

A.3.1.2 Racoon . . . . . p. 73

Referências p. 74

Lista de Figuras

1 Exemplo básico do funcinamento de uma VPN . . . . . p. 21

2 VPN implementada utilizando Frame Relay . . . . . p. 22

3 Frame Relay é invisível para o cliente . . . . . p. 22

4 VPN no modelo peer-to-peer . . . . . p. 23

5 Visão do cliente sobre um serviço de VPN baseada em VRFs. . . . . p. 24

6 Funcionamento básico do MPLS . . . . . p. 25

7 Exemplificação de IBGP e EBGP . . . . . p. 27

8 Exemplo de funcionamento do algoritmo vetor-caminho . . . . . p. 27

9 o *design* de uma rede baseada em VRF utiliza trunks 802.1q, túneis GRE ou *tags* MPLS para extender e amarrar estas VRFs. . . . . p. 28

10 VRF baseada em VPN,BGP e MPLS . . . . . p. 29

11 Configuração básica de uma VRF em um roteador CISCO . . . . . p. 30

12 Funcionamento do MPLS. . . . . p. 30

13 VPN criada a partir de túneis GRE. A medida em que a rede cresce, maior a complexidade. . . . . p. 32

14 VPN criada a partir de túneis mGRE. A complexidade é menor se comparada com túneis GRE comuns. . . . . p. 33

15 Implementação de túneis GRE em uma topologia overlay. . . . . p. 34

16 Implementação de túneis mGRE . . . . . p. 35

17 Exemplo de uma implementação usando DMVPN . . . . . p. 37

18 DMVPN - comunicação entre 2 SPOKEs . . . . . p. 41

19 Ping para o IP do túnel de BRASIL. . . . . p. 43

20 DMVPN com HUB e 2 SPOKEs. Túneis mGRE são permanentes. . . . p. 44

21 Túneis GRE entre os SPOKEs são criados durante uma comunicação entre eles. . . . . p. 45

22 Configurando o VirtualBox no GNS3 . . . . . p. 46

23 Configurando o VirtualBox no GNS3 . . . . . p. 46

24 Arquivo de configuração do opennhrp no HUB e SPOKEs . . . . . p. 47

25 Arquivo de configuração do IPSecTools no HUB e SPOKEs . . . . . p. 47

26 Script configurando a interface túnel . . . . . p. 48

27 Mensagens de inicio de sesão opennhrp (log opennhrp de um dos SPOKEs) p. 49

28 Diagrama das mensagens de inicio de sesão opennhrp . . . . . p. 49

29 Log opennhrp do SPOKE ARGENTINA no momento de um ping entre os dois SPOKEs . . . . . p. 50

30 Log opennhrp do HUB no momento de um ping entre os dois SPOKEs p. 51

31 Log opennhrp do HUB no momento de um ping entre os dois SPOKEs p. 52

32 Forma simplificada de uma redundância com DMVPN. . . . . p. 54

33 Cliente TABAJARA com uma redundância DMVPN simples utilizando rotas estáticas. . . . . p. 55

34 Cenário para teste de troca de acesso utilizando RIP. . . . . p. 56

35 DMVPN em conjunto com Quagga. As rotas são divulgadas para os outros *sites* via RIP. . . . . p. 57

36 Identificação dos vizinhos RIP . . . . . p. 58

37 Especificação de vizinhos no roteador Quagga. . . . . p. 59

38 Resultado esperado do RIP x Resultado real obtido. . . . . p. 60

39 Configuração final RIP. . . . . p. 61

40 Partes relevantes do arquivo zebra.conf do SPOKE RÚSSIA. . . . . p. 62

41 Arquivo ripd.conf do SPOKE RÚSSIA. . . . . p. 62

42 Resultado dos comandos ”show ip rip”e ”show ip rip status”no quagga router do SPOKE ARGENTINA . . . . . p. 63

43 Resultado de um ping entre ARGENTINA e a LAN de RÚSSIA . . . . p. 64

44 opennhrp.conf no HUB . . . . . p. 67

45 zebra.conf no HUB . . . . . p. 68

46 ripd.conf no HUB . . . . . p. 69

47 gre1_up no HUB . . . . . p. 69

48 opennhrp.conf no SPOKE2 . . . . . p. 70

49 zebra.conf no SPOKE2 . . . . . p. 71

50 ripd.conf no SPOKE2 . . . . . p. 72

51 gre1_up no SPOKE2 . . . . . p. 72

52 psk.txt em todos os *sites*. . . . . p. 73

53 racoon.conf em todos os *sites*. . . . . p. 73

Lista de Tabelas

Lista de Abreviaturas

VPN *Virtual Private Network*

ISP *Internet Service Provider*

PE *Provider's Edge*

CE *Customer's Edge*

BGP *Border Gateway Protocol*

MPLS *Multiprotocol Label Switching*

VRF *Virtual Routing and Forwarding*

AS *Autonomous System*

FIB *Forwarding Information Base*

RD *Route Distinguisher*

RT *Route Target*

RIP *Routing Information Protocol*

OSPF *Open Shortest Path First*

LDP *Label Distribution Protocol*

LSP *Label Switched Path*

FEC *Forwarding Equivalence Class*

LER *Label Edge Router*

LSR *Label Switching Router*

FEC *Forwarding Equivalence Class*

DMVPN *Dynamic Multipoint VPN*

NHRP *Next Hop Resolution Protocol*

IPSec *IP Security Protocol*

ARP *Address Resolution Protocol*

GRE *Generic Routing Encapsulation*

mGRE *Multipoint GRE*

IPv4 *Internet Protocol version 4*

IPv6 *Internet Protocol version 6*

1 *Introdução*

Neste capítulo são apresentados a motivação, os objetivos e a organização do texto desta proposta de TCC .

1.1 Motivação

Ao passar dos anos, dezenas de novas tecnologias surgem à nossa volta. Rapidamente passamos a usufruir delas, que acabam fazendo parte do nosso cotidiano. Um grande exemplo disto é a Internet.

Atualmente a internet está acessível em praticamente qualquer lugar que se vá e acabamos por não imaginar a complexidade que tal tecnologia leva desde a sua criação até podermos fazer uso da mesma.

Assim como usários finais, empresas corporativas vêm adquirindo diversas tecnologias para facilitação de negociações, trabalhos e melhores formas de se resolver problemas. Neste sentido, a interligação entre dois pontos fisicamente distantes é hoje que as empresas de médio a grande porte fazem uso à algum tempo. Neste contexto, a segurança dos dados logicamente é algo que não pode ser deixada de lado.

As prestadoras de serviços (ISP) provêm a comunicação de dados entre dois ou mais pontos usando o conceito de VPN. Uma VPN se comporta como uma rede completamente privada para um conjunto de *sites* geograficamente distribuídos. Essa comunicação pode ser no modo *overlay*, onde a ISP faz puramente a ligação dos pontos, sem interferência no nível de rede (camada OSI), ou pode ser do modo *peer-to-peer*, onde a ISP provê não somente a interligação dos pontos mas também participa no roteamento, tornando o trabalho feito pelo cliente muito mais fácil.

Assim como qualquer tecnologia, as VPNs também estão sujeitas a falhas. Apesar de serem difíceis de ocorrer, uma empresa corporativa com diversas filiais não pode arriscar a perda de comunicação de dados em um ou mais dos pontos contratados por muito tempo.

Por este motivo, o aconselhável é ter sempre um ponto secundário servindo como *backup* do serviço principal.

Backups para este tipo de serviço podem ser implementados de diversas formas: pode-se ter um circuito secundário com o mesmo tipo de serviço contratado ou pode ser utilizada uma outra abordagem com menor custo de implementação e de caráter dinâmico. Isto porque uma falha em um circuito deste tipo não dura muito tempo e não seria necessário todas as vantagens que o circuito principal oferecia, mas apenas uma alternativa para que o serviço do cliente não seja descontinuado.

Pensando nisso, podemos implementar de forma menos robusta o mesmo serviço oferecido pela ISP, desde que tenhamos um circuito com um IP público em cada uma das pontas que se queira a redundância de dados. Tal serviço seria implementado pelo próprio cliente, sem intervenções da ISP. A tecnologia DMVPN proposta pela CISCO pode ser usada neste sentido, permitindo implementar dinâmicamente túneis seguros entre os sites do clientes. Hoje já existe uma proposta *opensource* para implementação de DMVPN sobre o Linux. É dentro deste contexto que se insere este trabalho. A proposta visa explorar a implementação *opensource* para implementação de redundância em VPNs implementadas com o MPLS/BGP/VRF.

1.2 Objetivos do Trabalho

O objetivo do trabalho é avaliar o uso de DMVPN como uma alternativa de redundância para VPNs BGP/MPLS/VRF. Como objetivo específico tem-se:

- avaliar a complexidade e a escalabilidade da implementação de uma DMVPN usando uma implementação *opensource*;
- avaliar a capacidade da DMVPN atuar como redundância a uma VPN VRF/MPLS/BGP, identificando de que forma a VPN secundária pode atuar frente a determinados tipos de falhas: falha no enlace do lado CPE, do lado PE, do lado da matriz, do lado filiais entre outras;
- avaliar qual o melhor protocolo de roteamento a ser usado do lado cliente que permite um melhor desempenho na recuperação de falhas.

1.3 Organização do texto

O texto está organizado da seguinte forma: No capítulo 2 é apresentada a fundamentação teórica, com conceitos básicos e conhecimentos necessários para o entendimento e implementação. O capítulo 3 contém a proposta deste trabalho, contendo a motivação, os objetivos, o cronograma das atividades previstas e recursos necessários para a implementação prática.

2 *Fundamentação Teórica*

Neste capítulo é apresentada a fundamentação téorica necessária ao entendimento da proposta de TCC. Inicialmente será explicado o que é *Virtual Private Network* (VPN), suas características e os serviços envolvidos em sua criação. Após isso é apresentado um estudo sobre uma opção à *Virtual Private Network* (VPN), chamada DMVPN. Serão mostradas as suas características de implementação e também os serviços necessários para o seu funcionamento.

2.1 Conceitos em *Virtual Private Network* (VPN)

As VPNs são, de forma básica, interligações privadas feitas entre dois ou mais pontos fisicamente distantes. Têm como característica a velocidade real contratada e segurança dos dados trafegados, que nao estão sujeitos a ataques externos. Esse serviço de VPN é comumente utilizado por empresas a fim de interligar seus pontos remotos. Estsa VPNs são isoladas de outras que utilizam a mesma infraestrutura de conexão.

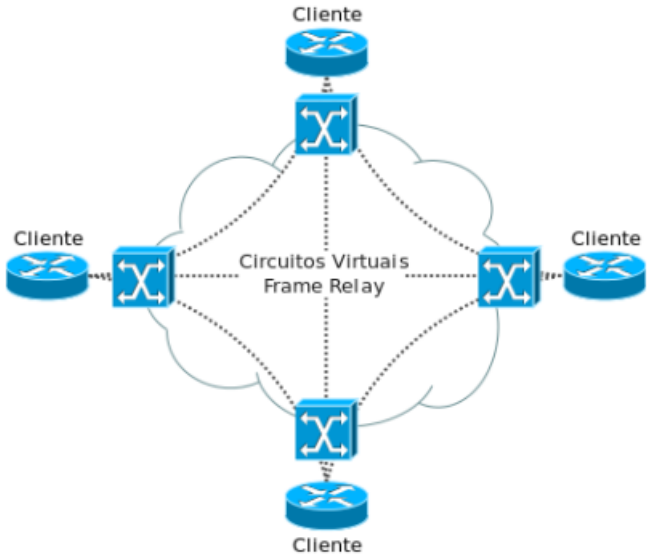
Utilizando a definição da RFC4364 (ROSEN; REKHTER, 2006): "Considere-se um conjunto de *sites* que são ligados a uma mesma rede chamada de *backbone*. Aplique-se alguma política para criar um certo número de subconjuntos desses *sites* e imponha-se a regra: dois *sites* podem ter conectividade IP no *backbone* apenas se no mínimo um destes subconjuntos contenha os dois *sites*. Estes subconjuntos são VPNs. Dois *sites* possuem conectividade IP somente se existem uma VPN que contenha ambos." Segundo a definição acima, podemos concluir que uma VPN funciona fazendo com que os pontos (*sites*) conectados fizessem parte de uma mesma rede. A figura 1 ilustra uma VPN típica.

(TYSON, 2007)

Figura 1: Exemplo básico do funcinamento de uma VPN

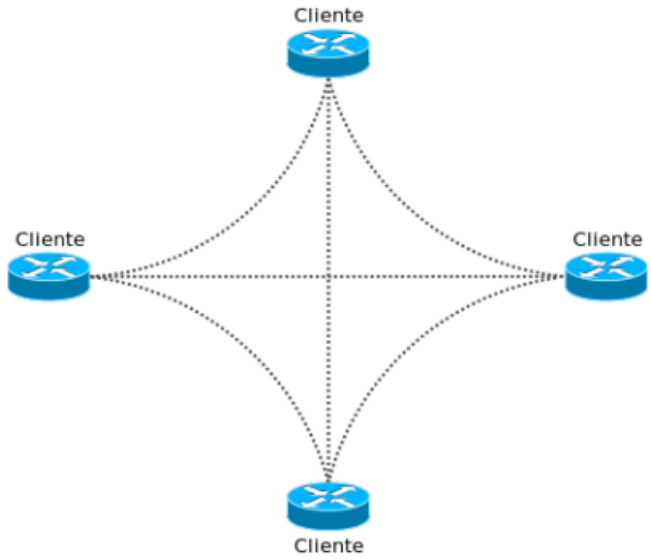
As *Virtual Private Network* (VPN) podem ser configuradas de dois modos diferentes. Abaixo são apresentados brevemente esses dois tipos de implementações, que são VPN *overlay* e VPN peer-to-peer.

- **Overlay:** (FERGUSON; HUSTON, 2007) Em uma implementação *overlay*, o *Internet Service Provider* (ISP) ou prestadora de serviços de internet provê apenas a interligação entre as redes do cliente, não participando em momento algum na parte de roteamento. O cliente é o responsável pelo roteamento entre as suas redes, podendo ter, por exemplo, um de seus roteadores como um "concentrador" que faz de fato o roteamento dentro de toda a sua rede. Além disso, todos os pontos da rede da ISP devem estar conectados entre si em um formato estrela. Um caso a ser citado são as estruturas Frame-Relay, atualmente já em desuso, onde a ISP provê apenas a comunicação física e enlace dos dados para os *sites* conectados. Como parte de ilustração de uma VPN baseada em *overlay* está abaixo a figura 2. A figura 3 ilustra como seria a visão do cliente de uma rede deste tipo.



(WESTPHAL, 2011)

Figura 2: VPN implementada utilizando Frame Relay



(WESTPHAL, 2011)

Figura 3: Frame Relay é invisível para o cliente

Um problema deste tipo de implementação é o número de conexões necessárias para a interligação de todos os componentes da VPN do cliente. Conforme colocado por Westphal (2011): "A fórmula para calcular o número de conexões fim a fim em uma rede totalmente entrelaçada é ((n)(n - 1))/2 , onde n é o número de sítios que se deseja conectar. Caso um cliente tenha 10 sítios em uma VPN, isso significa que seriam necessárias 45 conexões fim a fim no *backbone* para prover uma conectividade total entre esses sítios".

- **Peer-to-Peer**: (FERGUSON; HUSTON, 2007) Nessa implementação o ISP participa ativamente do roteamento das redes do cliente, oferecendo suporte na camada de rede (WESTPHAL, 2011). Neste caso não é mais necessário ao cliente ter um roteador responsável por todo o roteamento, pois isto é feito pela ISP. Porém, o cliente ainda assim pode ter roteadores que controlam a publicação de suas rotas na rede. Os roteadores de borda aprendem novas rotas dos *Customer's Edge*s (CEs), que são os roteadores do cliente, e as divulgam para os outros participantes da VPN. Assim as rotas não ficam concentradas em um só roteador e a tendência é de se obter um melhor desempenho geral da VPN, visto que possíveis erros de roteamento feitos no concentrador do cliente nao mais acontecerão. A figura 4 ilustra uma VPN *peer-to-peer*, em que o cliente tem 4 *sites* conectados e o roteamento desta VPN é feito pela ISP.

(WESTPHAL, 2011)

Figura 4: VPN no modelo peer-to-peer

2.2 VPN baseada em BGP/MPLS/VRF

O *backbone* ou rede principal de uma ISP geralmente funciona com o MPLS, com a finalidade de se obter um melhor desempenho e uma melhor estruturação da rede. A distribuição de tabelas de roteamento de protocolos dinâmicos, como o BGP, é feita de forma mais simples e eficiente desta forma. VPNs baseadas em BGP/MPLS/VRF são descritas em Rosen e Rekhter (2006), que terão o mesmo funcionamento de uma VPN *peer-to-peer*. Tendo como base a figura 5, é possivel observar como é o ponto de vista do cliente TABAJARA sobre suas redes interligadas via VRF. Para ele, é como se todos os seus pontos estivessem em uma sala conectados em um mesmo roteador.

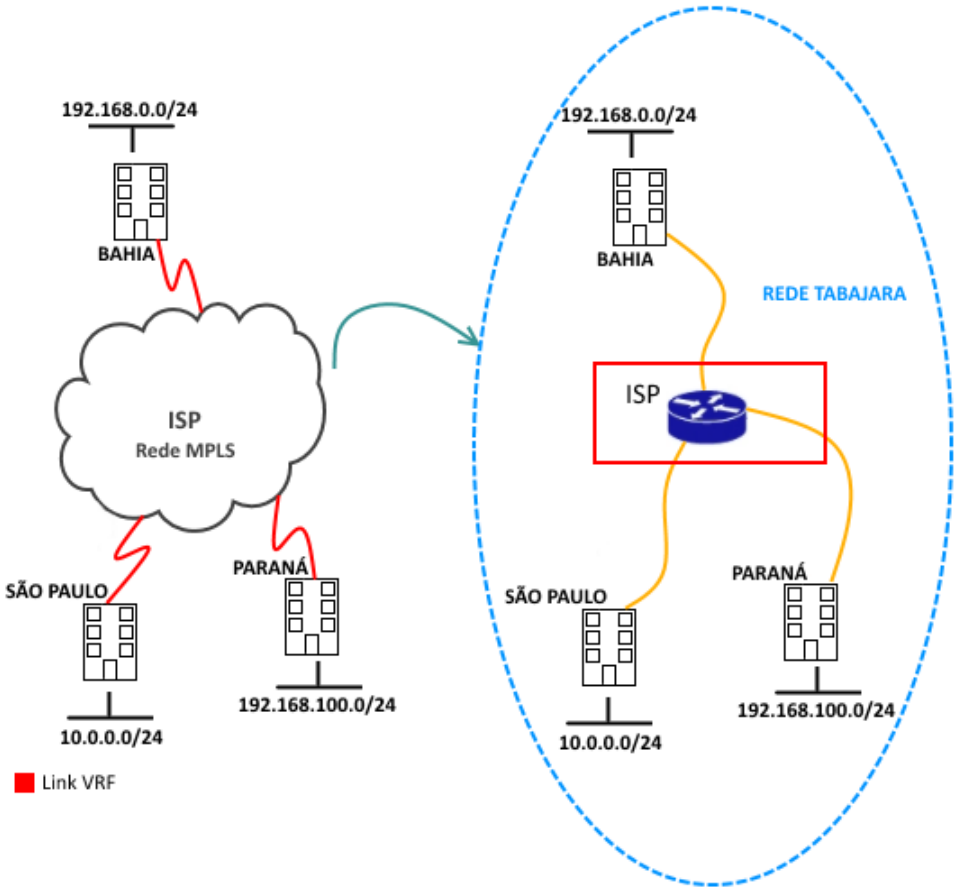


Figura 5: Visão do cliente sobre um serviço de VPN baseada em VRFs.

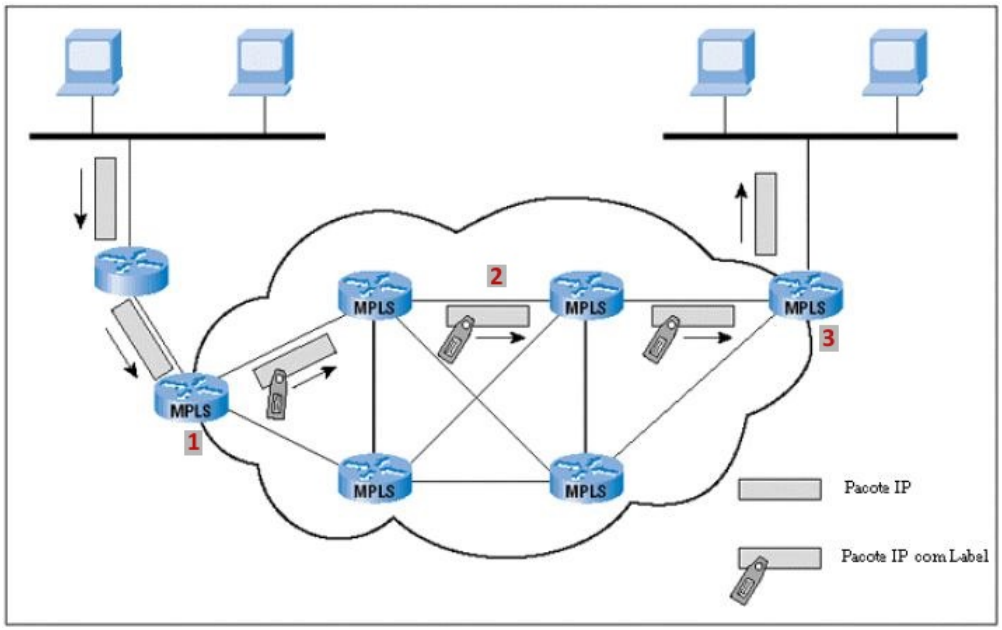
A seguir serão apresentadas as tecnologias MPLS, o protocolo BGP e o conceito de VRF.

2.2.1 Protocolos e Mecanismos usados na VPN baseada em BGP/MPLS/VRF

2.2.1.1 *Multiprotocol Label Switching* (MPLS)

O MPLS (ROSEN; VISWANATHAN; CALLON, 2001) é um mecanismo de encaminhamento com características de circuitos virtuais. O que ele faz é um encaminhamento de pacotes ao adicionar neles um cabeçalho, chamado de rótulo ou *label*. Este cabeçalho é incluso nos roteadores MPLS de borda, chamados de LER, e pode ser alterado durante o seu trajeto dentro da rede pelos roteadores MPLS, chamados LSR. Estes labels são somente legíveis à roteadores da rede MPLS, portanto quando o pacote chegar ao seu último salto dentro desta rede, o último roteador de borda irá retirar os labels para seu encaminhamento IP. Cada roteador MPLS tem uma tabela de encaminhamento, que é formada baseada em fluxos de pacotes que recebam as mesmas características de rotea-

mento, chamados de FEC. O caminho por onde determinado fluxo de pacotes seguirá é chamado de *Label Switched Path* (LSP), que é um caminho que fluxos de pacotes com mesma característica seguirão. Este caminho é modificado de acordo com a disponibilidade dos roteadores MPLS. A figura 6mostra que no momento em que o pacote IP entra na rede MPLS, ele recebe um *label* [1]. De acordo com o caminho escolhido, o pacote fará seu percurso dentro da rede MPLS até o destino [2]. Ao chegar útimo roteador da rede MPLS, o *label* é removido e o pacote IP é encaminhado ao destino.



(UFRJ, 2012)

Figura 6: Funcionamento básico do MPLS

As siglas referentes ao MPLS seguem:

- *Label Switching Router* (LSR): Estes são equipamentos que somente farão encaminhamento baseando-se nos rótulos MPLS. Ao receber um pacote, estes roteadores procuram em sua tabela de encaminhamento correspondente qual deverá ser o *next-hop* (próximo salto) e o rótulo antigo pode ser substituído por um novo (por exemplo, o pacote chega com um rótulo 10, de acordo com sua tabela de encaminhamento, esse pacote agora deve ter um rótulo 15).

- *Label Edge Router* (LER): Estes equipamentos fazem o mesmo que os LSR e também operam na borda de uma rede MPLS e eles irão basicamente incluir um rótulo em um pacote que entrará na rede MPLS ou retirar o rótulo de um pacote que sairá da rede MPLS. Ao incluir um label em um pacote que entrará na rede MPLS, ele usa

as informações de roteamento para determinar o label a ser inserido, e, ao retirar um label para saída da rede MPLS, ele irá tratar o pacote com roteamento IP.

- *Label Distribution Protocol* (LDP): Este é o responsável pela distribuição dos rótulos entre LSRs e LERs.
- *Forwarding Equivalence Class* (FEC) ou classe de encaminhamento equivalente: Um termo aplicado para fluxos de pacotes que recebem as mesmas características de encaminhamento.
- *Label Switched Path* (LSP): Um circuito virtual por onde um determinado fluxo de pacotes irá trafegar. Este túnel é unidirecional, então não necessariamente o caminho de ida é o mesmo da volta. (TAFT, 2004)No momento em que um pacote entra numa rede MPLS, este é associado a uma classe de equivalência (FEC) e assim é criado um LSP relacionado a esta FEC

2.2.1.2 O protocolo *Border Gateway Protocol* (BGP)

O BGP é um protocolo de roteamento dinâmico que utiliza um algoritmo de vetor-distância para escolha da menor distância até o destino.

2.2.1.3 Características básicas

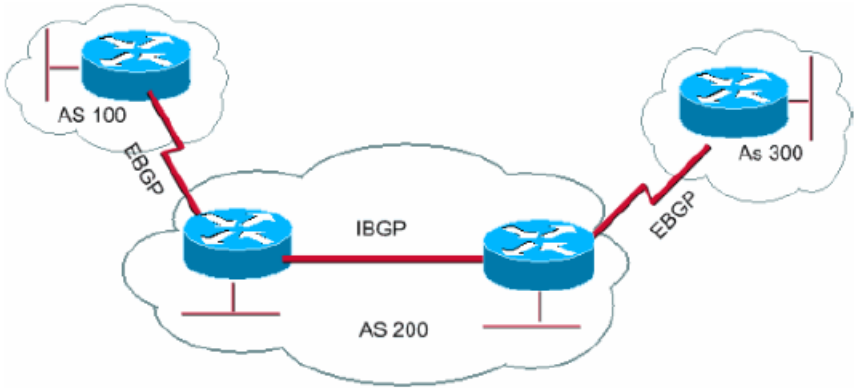
O BGP é um protocolo de roteamento dinâmico utilizado para conexões entre sistemas autônomos, conhecidos como *Autonomous System*s (ASs) e encontra-se atualmente na versão 4 (REKHTER; LI; HARES, 2006). O termo AS serve para identificar grupos de redes que compartilham uma mesma administração e uma mesma política de roteamento. É o caso de um ISP onde um identificador de AS é associado a sua rede interna. O BGP utiliza um algoritmo de escolha de rotas vetor-distância, onde cada roteador mantém uma tabela de rotas com a menor distância conhecida até o destino.

2.2.1.4 Funcionamento

Dois roteadores BGP (*peers* ou vizinhos) trocarão mensagens de abertura e confirmação de parametros de conexão. Estes roteadores trocam informações de alcançabilidade de rede. Inicialmente, toda a tabela de roteamento BGP é trocada entre eles, após isso são enviadas apenas atualizações destas tabelas.

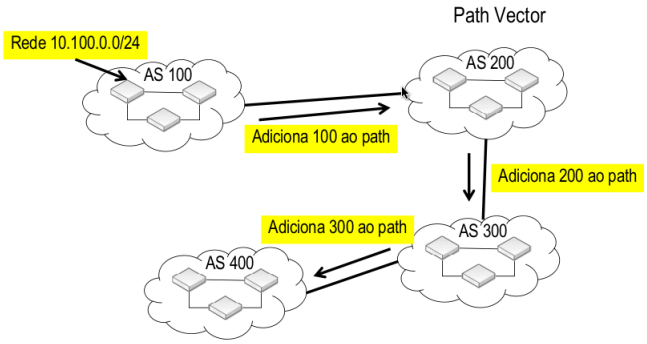
Através de um atributo chamado ASPATH na publicação de uma rede, concatena-se o identificador do AS que está realizando a publicação com toda a sequência de ASs por onde a publicação foi recebida. Na figura 8 é ilustrado o modo de funcionamento de um algoritmo vetor-distância. É possivel verificar que a cada salto é adicionado um valor ao peso do caminho . Se um AS recebe uma publicação por vários roteadores de borda ou enlaces diferentes, a seleção do melhor caminho é baseada no menor ASPATH, ou seja, no menor número de saltos em termos de número de ASs. Não são considerados a capacidade e ocupação dos enlaces do caminho ou quaisquer outros custos.

O BGP funciona com dois conceitos: o IBGP e o EBGP. É chamado IBGP quando os roteadores pertencem a um mesmo AS (geralmente fazendo parte do núcleo da rede) e servem como trânsito para outros AS. O EBGP seria uma conexão entre *peers* com número de AS distintos. A figura 7 ilustra o posicionamento do EBGP e IBGP.



(CISCO, 2008)

Figura 7: Exemplificação de IBGP e EBGP

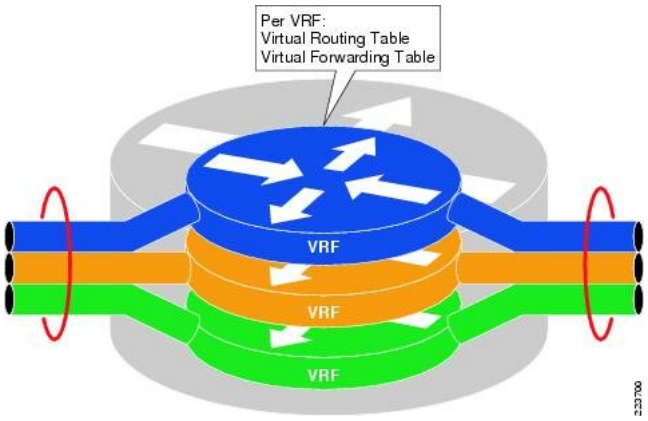


(SOUZA, 2011)

Figura 8: Exemplo de funcionamento do algoritmo vetor-caminho

2.2.1.5 *Virtual Routing and Forwarding* (VRF)

A tecnologia VRF foi introduzida pela Cisco e permite ter múltiplas tabelas de roteamento dentro de um mesmo roteador, dando a impressão de que existem diversos roteadores trabalhando dentro de um só roteador físico. Essas tabelas de roteamento são distintas umas das outras e são conhecidas como *Forwarding Information Base* (FIB). Cada uma destas *FIBs* representa um roteador virtual. Por exemplo: seja 2 clientes no mesmo *Provider's Edge* (PE), que é o roteador de borda da ISP que provê acesso ao roteador do cliente, um deles pertencente à VRF "alfa", enquanto que o outro pertence à VRF "beta". O endereço IP configurado no CE pertecente à "alfa"é 192.168.0.2, que coincidentemente é igual ao endereço IP configurado no CE pertencente à "beta". Apesar da duplicidade de IPs, não haveria nenhum tipo de problema neste cenário, pois cada VRF é vista como um roteador, e um não enxerga a rede do outro. A figura 9 ilustra a idéia de diversos roteadores virtuais dentro de um real.



(PATTERSON, 2009)

Figura 9: o *design* de uma rede baseada em VRF utiliza trunks 802.1q, túneis GRE ou *tags* MPLS para extender e amarrar estas VRFs.

2.2.2 O plano de controle da VPN BGP/MPLS/VRF

As tecnologias BGP, MPLS e VRF podem ser combinadas para a criação de VPNs BGP/MPL/VRF(ROSEN; REKHTER, 2006) no modelo *peer-to-peer* e permitem que clientes que estejam ligados em um mesmo *backbone*, não se comuniquem e não saibam da existência um do outro, dando a impressão de que cada cliente tem uma rede de links dedicados e fisicamente separados.

A descoberta de novas rotas dentro de uma VRF é realizada pelo BGP com apoio dos campos *Route Distinguisher* (RD) e *Route Target* (RT). Estes dois campos são necessários

não só para a distinção de uma VRF das outras, mas também é utilizado para o BGP fazer a distribuição de rotas da VRF em questão. Um RD é um número que distingue uma VRF de outra , ou seja, cada VRF tem a sua. O RT permite uma flexiblidade adicional permitindo a exportação e importação de rotas entre VPNs.

Deve ser observado que o cliente pode optar em utilizar um outro protocolo de roteamento no lugar de BGP nas bordas, como *Routing Information Protocol* (RIP) ou *Open Shortest Path First* (OSPF), porém nas bordas da rede de transporte é utilizado o BGP.

O modo como funciona a distribuição de rotas é explorado junto à figura 10:

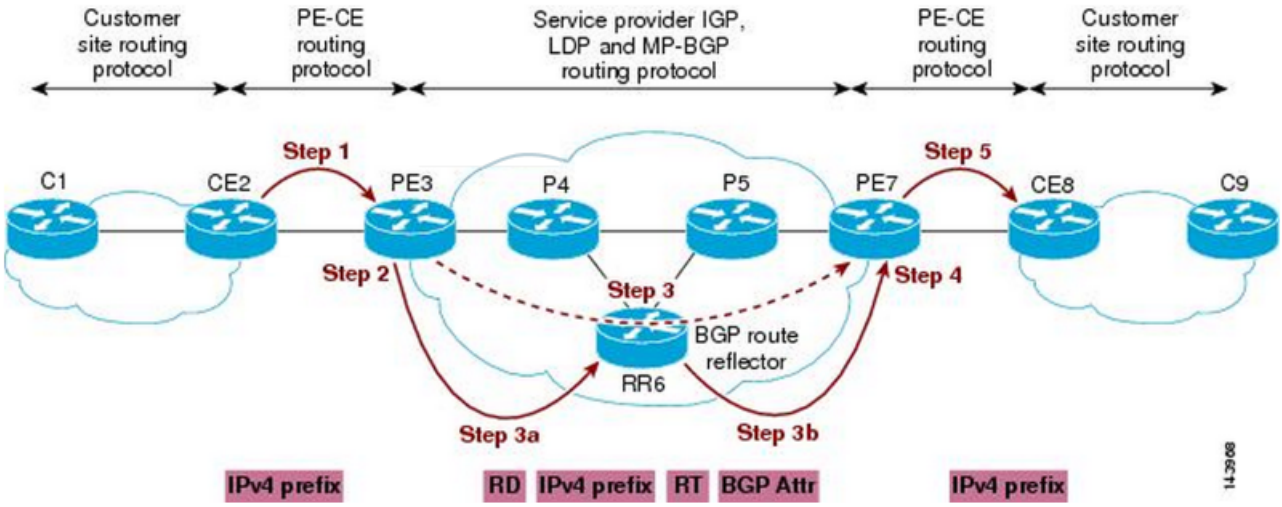


Figura 10: VRF baseada em VPN,BGP e MPLS

1. As rotas dos CEs são divulgadas para a VRF no PE através de rota estática, RIP, OSPF ou BGP. As rotas do cliente são passadas como prefixos IPv4. [*Step* 1]

2. As rotas da VRF no PE são exportadas como prefixos VPNv4 que são constituídos pelo prefixo da rede publicada com o RD anexado. [*Step* 2]

3. As rotas exportadas são enviadas para o núcleo da rede via MPLS, chegando a todos os roteadores BGP de borda da rede. O iBGP é utilizado neste processo e todos os roteadores devem estar ligados em rede mesh ou usando alguma técnica de federação (exemplo: *route reflector*). O PE originário das rotas deve criar o rótulo MPLS para a VPNv4 e agregar este rótulo na publicação; [*Step* 3a e 3b]

4. As rotas são importadas pela VRF nos PEs destino [*Step* 4] e divulgadas para os CEs [*Step* 5]. Cada VRF tem definidas as configurações de *import* e *export*. O campo *import* define de onde serão acrescentadas novas rotas à tabela de roteamento da VRF no PE em questão. O campo *export* define para onde sua rotas serão

anunciadas. No caso do BGP eles são definidos via RT, como exemplifica a figura 11, que é parte da configuração em um roteador real. Note-se que somente os prefixos VPNv4 iguais ao tag configurado em um VRF serão importados. O rótulo MPLS é associado a ao prefixo da rede de forma que os pacotes possam ser devidamente encaminhados;

5. No momento em que uma nova rota é adicionada em uma das pontas do cliente, o CE as divulga via roteamento para o PE, que através de seu RD e RT correspondente as divulga para toda a rede do cliente, ou seja, todo o processo se repete.

```
ROUTER1#sho conf | be xxxx
ip vrf xxxx
 rd 5151:10000
 route-target export 5151:10000
 route-target import 5151:10000
!
```

Figura 11: Configuração básica de uma VRF em um roteador CISCO

2.2.3 O plano de dados BGP/MPLS/VRF

O modo de funcionamento do MPLS é explorado abaixo juntamente à figura 12 (**??**):

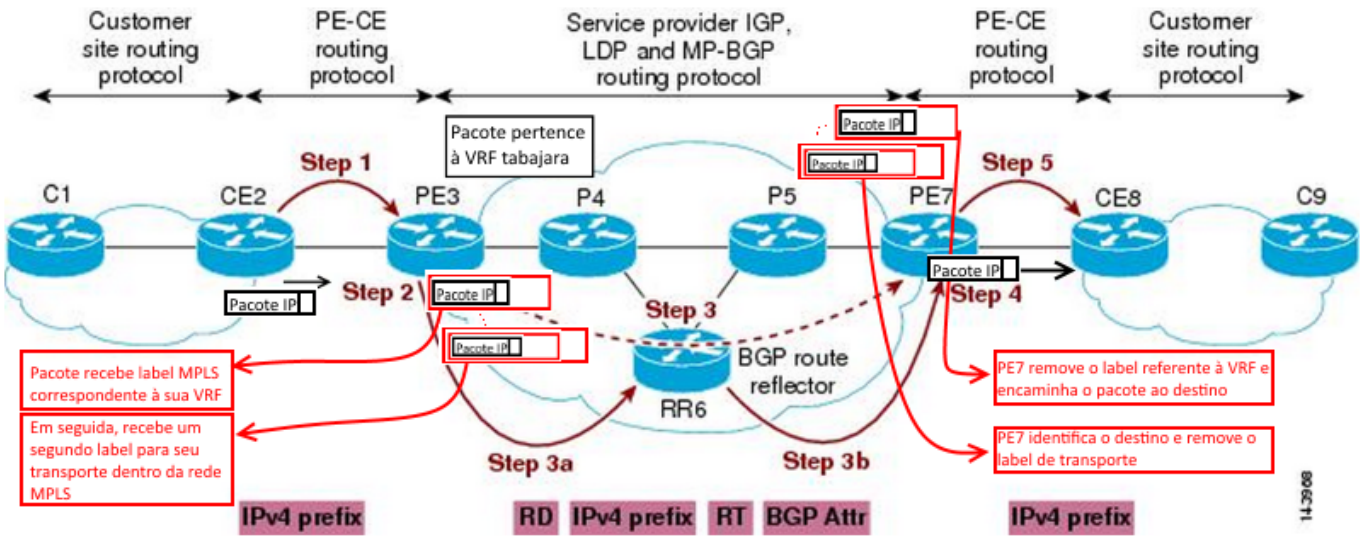


Figura 12: Funcionamento do MPLS.

1. Ao ser gerado um fluxo de dados com um outro ponto daquela VRF como destino, este fluxo é encaminhado ao PE.

2. O PE recebe os pacotes e analisa a tabela de roteamento da VRF atrelada àquela interface. Neste caso, ele fez uma descoberta do PE destino via BGP. É adicionado ao pacote um primeiro *label* MPLS que serve para identificação da VPN em questão. Após isso, um novo *label* MPLS é adicionado. Este segundo encapsula o primeiro *label* e serve para seu encaminhamento dentro da rede MPLS.

3. Os pacotes seguem seu caminho pela rede da ISP. Cada roteador da ISP toma uma decisão de encaminhamento diferente. Neste cenário, o segundo *label* pode ser trocado diversas vezes até chegar ao seu PE destino, porém o primeiro *label* não sofre modificações durante seu trajeto, pois corresponde à VRF.

4. O PE destino recebe o pacote e reconhece a VPN correspondente através do primeiro *label*. O PE remove ambos os *labels* e encaminha o pacote para a interface do CE destino correspondente.

   Alternadamente, o segundo label pode também ser removido no último salto antes do PE destino, dependendo da configuração feita no MPLS.

2.3 A tecnologia DMVPN

DMVPN é um conceito para a criação de uma VPN em que são utilizados acessos à Internet para a ligação entre os pontos. Esta é uma abordagem relativamente nova e nao foi encontrado um produto que ofereça este tipo de serviço. É possível sua implementação com recursos *opensource* e através de roteadores Cisco com o devido suporte.

2.3.1 Visão geral

VPNs podem ser construídas de diversas formas quando não é pretendido contratar uma ISP para fazê-lo. Desde que se tenha conectividade IP, é possível formar uma VPN entre os *sites* desejados fazendo uso do GRE, que é um protocolo de tunelamento que permite o encapsulamento de uma variedade de outros protocolos. Supondo que se tenha quatro *sites* fisicamente distantes e com IPs válidos de Internet imutáveis, pode-se formar uma VPN utilizando túneis GRE, cujos pontos finais são IPs válidos, como mostra a figura 13. Estes são túneis formados estaticamente entre todos os *sites* participantes. Desta forma, esta VPN tem uma abordagem *overlay*. A VPN possuirá sua própria faixa de IPs privados e terá como transporte a Internet. Para ajudar a manter os dados trafegados

não visíveis a possíveis interceptações, pode ser utilizado o IPSec para encriptografar os pacotes desta VPN. A figura 13 ilustra uma VPN criada desta forma.

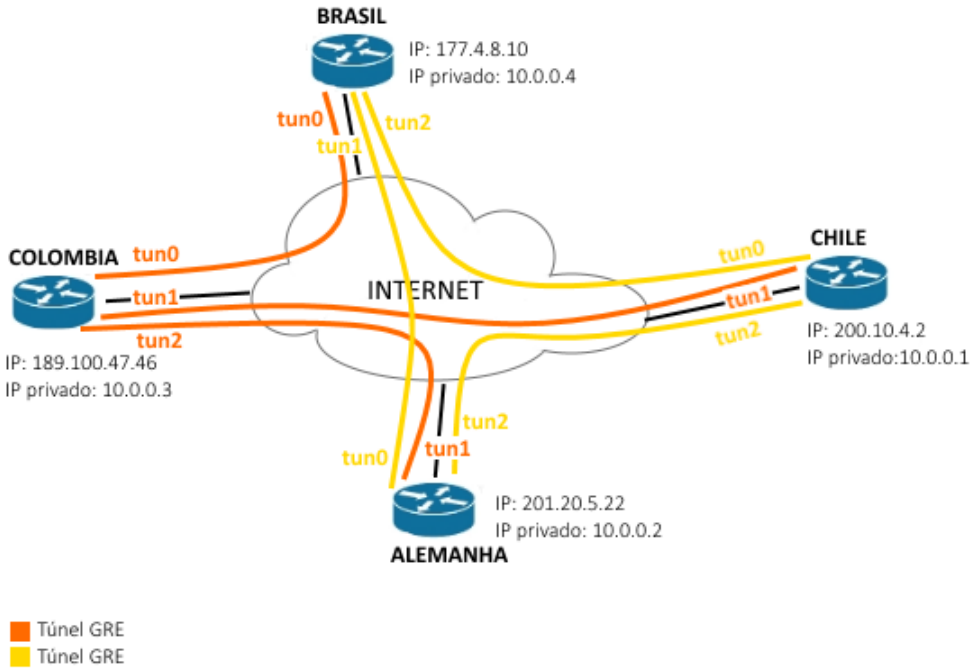


Figura 13: VPN criada a partir de túneis GRE. A medida em que a rede cresce, maior a complexidade.

A dificuldade nesta implementação aumenta com o crescimento da VPN, pois a cada novo participante é criado um túnel adicional em cada um dos outros *sites*. A quantidade de túneis necessários pode ser obtida pela fórmula ((n)(n - 1))/2, onde n é o número de *sites* que se deseja conectar. Uma forma de contornar isso é remover os túneis estáticos entre todos os *sites* e formar túneis multiponto mGRE com um servidor que resolverá endereços IP válidos dos destinos. O mGRE é uma proposta de formação de túneis multiponto e é uma extensão dos túneis GRE. Na figura 17 este servidor é o CHILE. Este servidor conhecerá todos os outros participantes da VPN, pois manterá um túnel mGRE com cada um, como é possível acompanhar na figura 17. Túneis mGRE são uma variação dos túneis GRE, e permitem a criação de um único túnel com diversos pontos finais. Quando um dos participantes quiser se comunicar com um outro, ele perguntará ao servidor o IP válido do destino. Com a resposta, *sites* origem e destino criam um túnel GRE entre eles, e após um tempo de inatividade, este túnel GRE é removido. Este é o conceito de DMVPN, que faz uso de túneis mGRE, NHRP e IPSec. A figura 17 ilustra uma DMVPN com quatro sites. É possivel verificar que é necessário apenas uma interface túnel em cada ponto, diferente da topologia anterior. Cada site continua com os endereços públicos e privados.

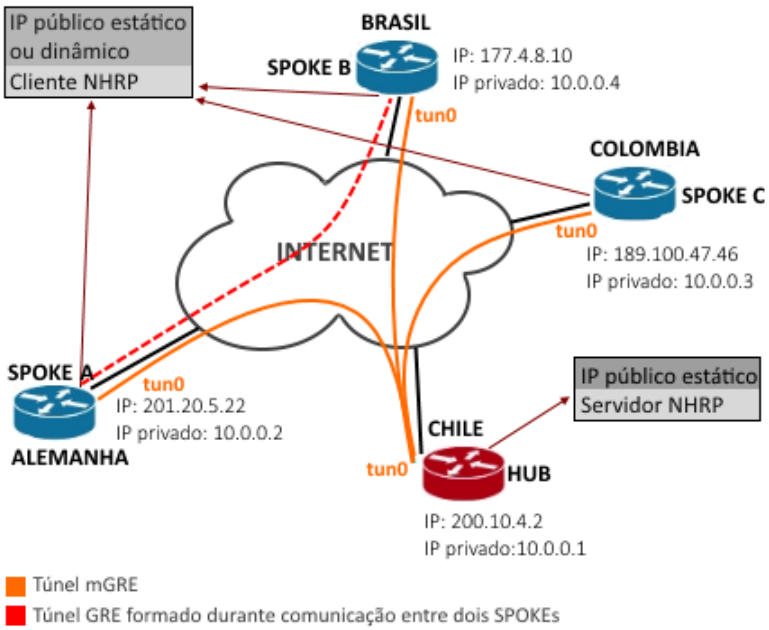


Figura 14: VPN criada a partir de túneis mGRE. A complexidade é menor se comparada com túneis GRE comuns.

2.3.2 Protocolos e Mecanismos usados na DMVPN

A DMVPN faz a combinação de alguns protocolos para a criação das redes privadas, que são o mGRE para a formação dos túneis multiponto, o NHRP para resolução de endereços IP e IPSec para criptografia dos dados. O roteamento de redes de cada *site* pode ser feita utilizando o *Routing Information Protocol* (RIP).

2.3.2.1 *Multipoint GRE* (mGRE)

O mGRE é uma proposta de formação multiponto de túneis e se consitui em uma extensão do túnel GRE (FARINACCI et al., 2000).

O túnel *Generic Routing Encapsulation* (GRE) permite o transporte datagramas IP. Um exemplo de utilização é a criação de uma rede privada utilizando a internet como transporte. Um túnel GRE pode ser considerado como um túnel ponto a ponto, em que é necessário um IP público na rede transportadora para cada *site*, com finalidade de todas as pontas se "enxergarem".

Na formação de uma rede privada totalmente conectada fazendo uso de túneis GRE, é utilizada a fórmula ((n)(n - 1))/2 para determinar a quantidade de túneis necessários, onde n é o número de *sites* que se deseja conectar. Em uma rede privada pequena, considerando

a internet como transporte, estes túneis podem ser considerados para a utilização, pois o número de túneis formados em cada *sites* é baixo. Isso muda se for considerada uma rede de médio a grande porte, em que fica mais complexa e difícil a implementação devido à configuração que deve ser modificada em todos os *sites* quando um novo participante da VPN é inserido. Outro ponto que dificulta a implementação é a necessidade de um IP público imutável em cada participante da rede privada. A figura 15 mostra a formação de uma rede privada utilizando túneis GRE em uma topologia *overlay*. Neste caso, todos os pontos da VPN tem IPs imutáveis em suas interfaces físicas e mantém túneis GRE com os outros *sites*. É possível verificar que todos estes túneis têm como saída as interfaces físicas. A cada novo participante na VPN adicionado, são modificadsa as configurações dos *sites* para a criação de um novo túnel.

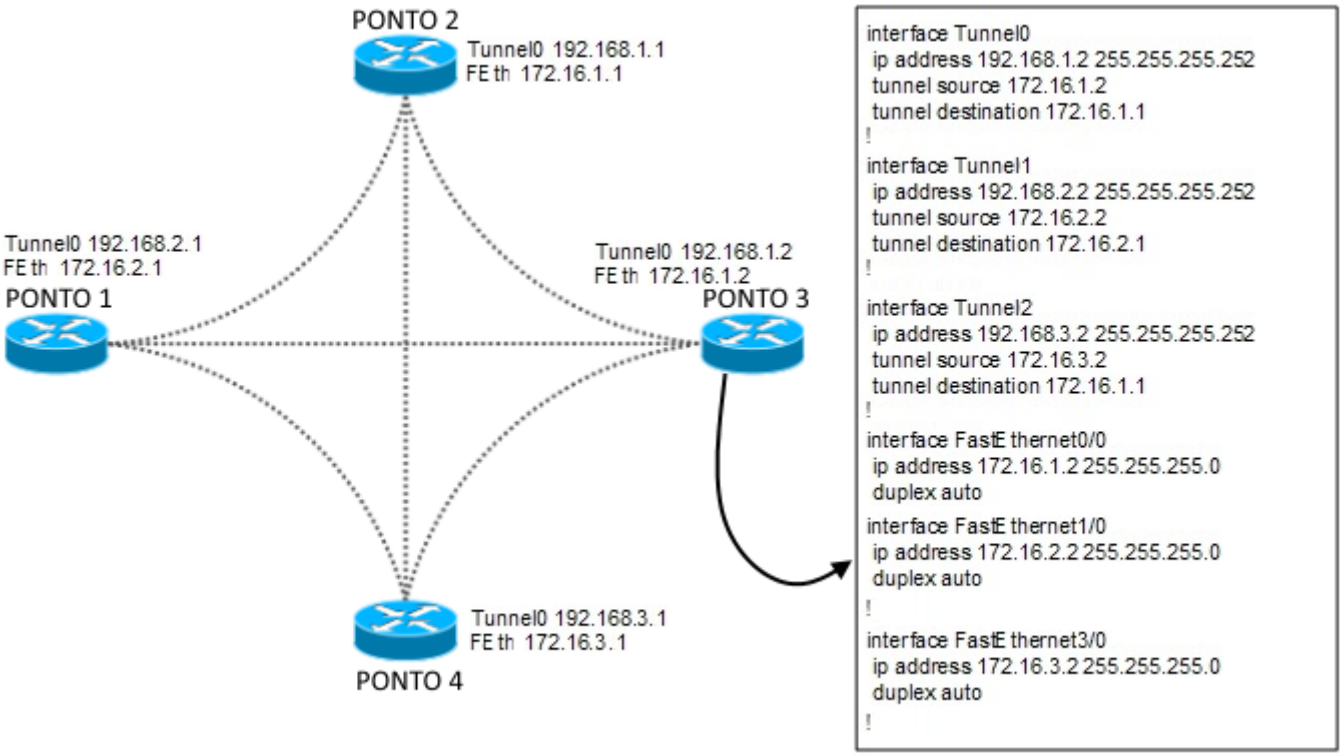
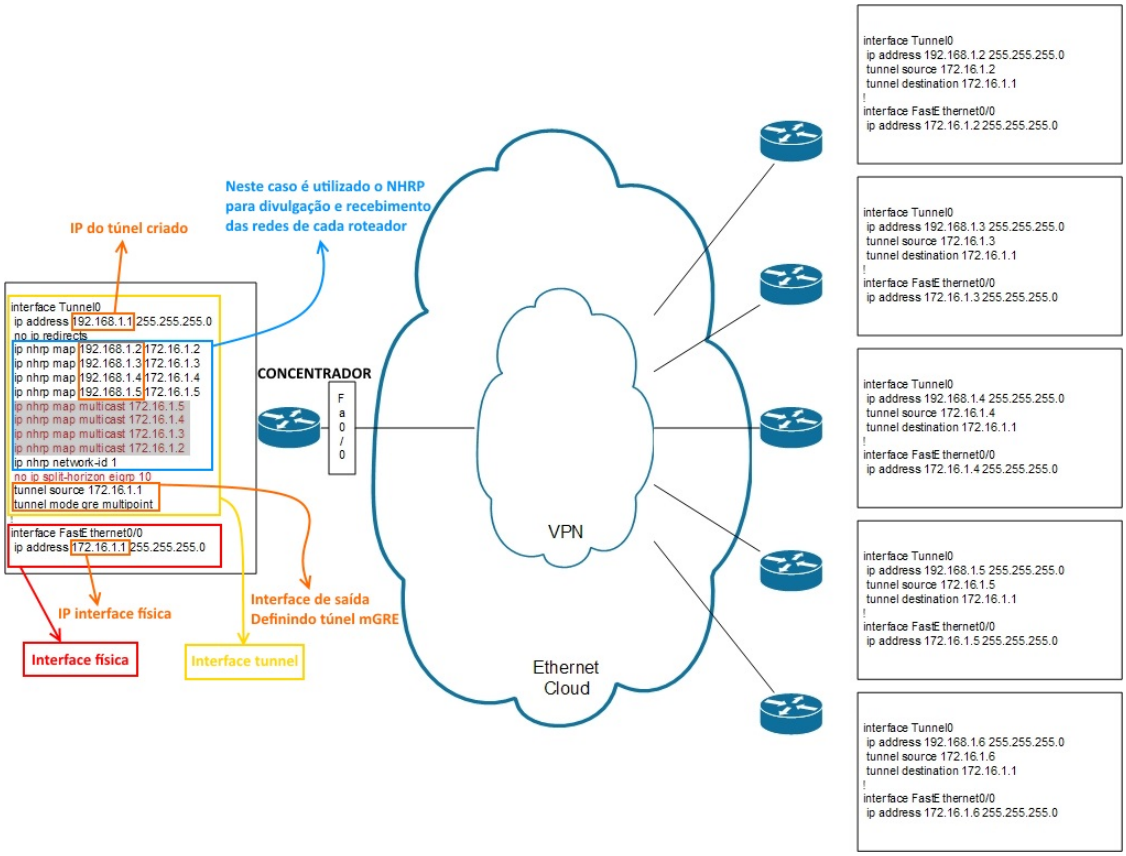


Figura 15: Implementação de túneis GRE em uma topologia overlay.

Uma solução para o GRE comum é o mGRE, que graças à habilidade de mapeamento de tuneis IP com um endereçamento IP lógico, o possibilita ter tuneis criados de forma dinâmica sem a necessidade de intervenção em todo túnel existente na rede. A figura 16 ilustra a mesma VPN formada acima, porém utilizando túneis mGRE. É possivel verificar que agora existe somente uma interface túnel no concentrador do tipo *multipoint*. É possível observar que neste caso é utilizado o NHRP para resolução de endereços lógicos. Utilizados em conjunto, o concentrador necessita de um túnel mGRE apenas para conexão

de todos os pontos e resolução dos endereços IP privados, a configuração em si é mais simplificada. Os endereços IP nas interfaces físicas continuam imutáveis. Em adição, como será visto é possível adicionar dinamicamente novos pontos.



(SAXENA, 2013)

Figura 16: Implementação de túneis mGRE

2.3.2.2 O *Next Hop Resolution Protocol* (NHRP) e a construção dinâmica

O protocolo NHRP (LUCIANI et al., 1998) é um protocolo que funciona em um sistema cliente-servidor, onde um servidor chamado de HUB resolve endereços IP de clientes, que são chamados de SPOKEs. São criadas sessões NHRP entre o servidor e clientes. Desta forma, o servidor mantém uma base de dados com os endereços IP de todos os SPOKEs. Um SPOKE registra o seu endereço com o HUB após o inicio de sessão e quando necessita, pergunta ao HUB pelo endereço de um outro SPOKE.

Assim, no contexto de VPNs, o NHRP pode ser utilizado para determinar o endereçamento IP lógico para subredes através de uma rede pública (FOX; PETRI, 1999), funcionando de forma similar ao ARP na ethernet.

2.3.2.3 *IP Security Protocol* (IPSec)

O IPSec é um serviço previsto na RFC 6071 que oferece segurança nos dados IP trafegados na rede. Ele permite a criptografia dos pacotes trafegados, assim caso os dados sofram interceptações eles não serão legíveis. O IPSec opera na camada de rede do modelo OSI, é opcional tanto em implementações *Internet Protocol version 6* (IPv6) como em *Internet Protocol version 4* (IPv4). Há dois modelos básicos de funcionamento: ele opera no modo transporte ou no modo túnel. No modo transporte ou básico, somente a mensagem (payload) é criptografado, utilizdo em comunicações host-a-host. No modo túnel, todo o pacote IP é criptografado, recebendo um novo encapsulamento para ser distribuído, utilizado em comunicações rede-a-rede, host-a-rede ou host-a-host sobre a internet. O IPSec é um mecanismo de segurança que é utilizada na formação de DMVPNs neste trabalho, porém nao serão abordados especifidades relacionadas à ele.

2.3.3 *Routing Information Protocol* (RIP)

O RIP é um protocolo de roteamento dinâmico, em que os roteadores utilizando RIP trocam mensagens periodicamente. Elas são relacionadas as rotas que são conhecidas com base na tabela de roteamento atual e a distância do roteador para cada uma das rotas. Essa distância é descoberta fazendo uso do algoritmo vetor-distância, baseada no número de roteadores existentes no caminho entre os dois roteadores. O roteador que recebe estas mensagens calcula a distância para as demais redes recebidas baseando-se na distância para o roteador que enviou a mensagem.

Após uma sessão RIP ser iniciada entre dois ou mais roteadores, as tabelas de roteamento deles é trocada periodicamente, por padrão é entre intervalos de 30 segundos. Isso acontece mesmo que nao haja nenhuma alteração nas tabelas de roteamento.

2.3.4 Formação de uma rede DMVPN

A DMVPN é baseada nas tecnologias mGRE, NHRP e IPSec, que atuam em conjunto neste cenário. Através de túneis mGRE criados entre o HUB e os SPOKEs, o NHRP é utilizado para resolução de IPs lógicos e o IPSec criptografa os pacotes trafegos. Pode-se fazer uma analogia do NHRP ao *Address Resolution Protocol* (ARP), que provê a habilidade de mapeamento de um endereço IP com um endereço IP lógico. Desta forma o mGRE pode configurar os túneis dinamicamente sem uma configuração explícita de próximo salto entre cada cada destino. A DMVPN é ilustrada conforme a figura 17, que

é possível notar o HUB como servidor NHRP e tendo endereço IP estático. Os SPOKEs são clientes NHRP e podem ter endereçamento estático ou dinâmico. Os túneis mGRE são criados com o HUB e cada *site* tem seu endereço privado.

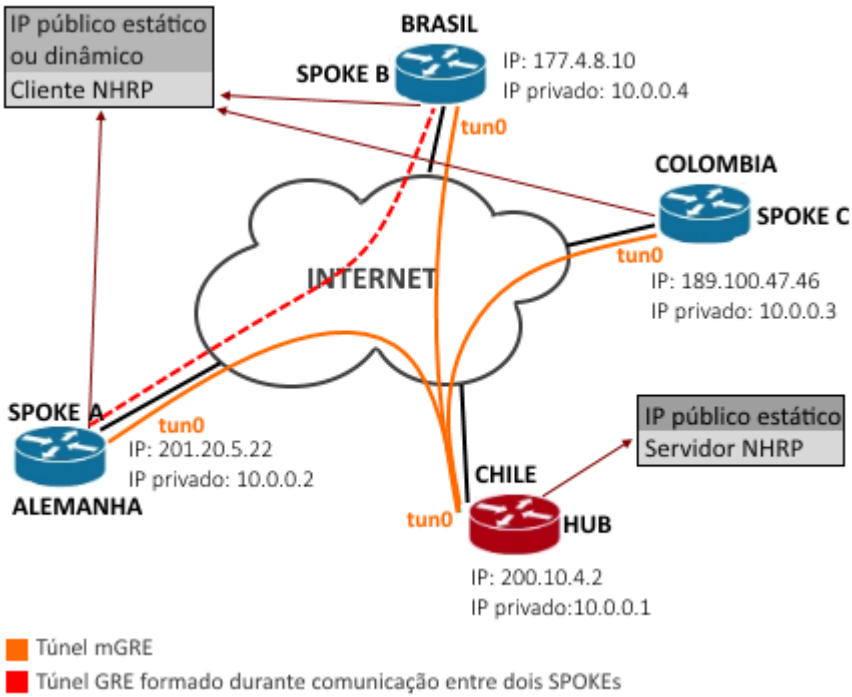


Figura 17: Exemplo de uma implementação usando DMVPN

De forma geral, a formação de uma rede DMVPN é realizada da seguinte forma:

- Cada SPOKE mantém, de forma estática, um tunel mGRE para um HUB. O HUB em princípio é único e possui um endereço IP estaticamente alocado. No HUB se executa um servidor NHRP que recebe registros dos clientes (SPOKEs). Não existem túneis alocados estaticamente entre SPOKEs.

- Quando um SPOKE fonte quer enviar um pacote para um destino privado, ou seja, uma subnet ligada a um outro SPOKE, ele pergunta ao servidor NHRP qual é o endereço do SPOKE alvo;

- De posse do endereço do SPOKE alvo, o SPOKE fonte inicia dinamicamente um túnel GRE com o SPOKE alvo. O túnel SPOKE-SPOKE é construído sobre uma interface GRE;

- Os pacotes transmitidos pelo SPOKE origem para o SPOKE alvo são conduzidos pelo túnel GRE dinamicamente formado. Os pacotes não passam pelo HUB.

- Após um tempo de inatividade pré-configurado no SPOKE alvo, o túnel SPOKE-SPOKE é removido.
- (FILIPETTI, 2008) Caso um protocolo de roteamento dinâmico seja utilizado para divulgação de rotas, o multicast tem que estar ativo nos túneis HUB-SPOKE.
- O HUB necessita de um IP público estático, que serve para os SPOKEs o encontrarem e formarem o túnel mGRE. Os SPOKEs não necessitam de IP público estático, uma vez que o caminho a se formar o túnel GRE com outros SPOKEs é informado pelo HUB.
- Independente de quantos SPOKEs se tenha, somente é configurada 1 interface túnel no HUB, onde se determina a quantidade de conexões que se manterão ativas. Túneis GRE comuns necessitam de configuração de uma interface túnel para cada SPOKE no HUB.

2.3.5 Soluções de DMVPN

2.3.5.1 Soluções Comerciais

A implementação em roteadores cisco é simplificada, basta a configuração de túneis mGRE nas pontas conforme figura 12. Como DMVPN, assim como NHRP são proprietários da cisco, roteadores de outras marcas podem não suportar o serviço, apesar de suportarem essa funcionalidade de modo alternativo. Roteadores Juniper SRX suportam a detecção de mais de um vizinho para uma conexão ou ainda realizando a configuração VPN de um SRX para múltiplos sites. Roteadores Viatta suportados pelo software 6.6R1 e superiores têm suporte à DMVPN via mGRE/NHRP/IPSec.

2.3.5.2 Soluções Abertas

É possível implementar DMVPNs no Linux combinando pacotes existentes para a construção de mGREs e IPsec com uma solução aberta do NHRP, o OpenNHRP - que implementa o NHRP. Esta solução é, inclusive, compatível com as DMVPNs da cisco.Essa tecnologia tem uma configuração simplificada e alguns testes já publicados confirmam sua compatibilidade com roteadores Cisco (PREUSS, 2009).

3 Experimentos com Redes Privadas Dinâmicas (DMVPN) utilizando software livre

3.1 Introdução

Neste capítulo serão apresentados os cenários que foram criados ao longo do trabalho, sendo mostrados detalhes de instalação e configuração das ferramentas utilizadas, das topologias e o modo de funcionamento dos protocolos envolvidos. Também são apresentados todos os testes executados com os cenários, como conectividade e análises de troca de mensagens.

O objetivo dos experimentos é avaliar o comportamento da DMVPN utilizando softwares livres sobre o Linux, avaliando se este comportamento está de acordo com a proposta da Cisco.

3.2 Ferramentas utilizadas

Toda a parte prática foi feita utilizando um conjunto de máquinas virtuais e um software que auxilia na interligação delas, que se chama GNS3. As máquinas virtuais utilizadas foram criadas a partir do VirtualBox e o sistema instalado nelas foi o Ubuntu 13.04. Internamente, cada um desses sistemas utiliza também um conjunto de ferramentas e protocolos, que são o OPENNHRP, IPSecTools e também o Quagga, conforme descritos a seguir.

- GNS3: é uma ferramenta que possibilita simulação de redes utilizando máquinas virtuais, roteadores, switches ou mesmo a máquina física, permitindo conectá-los como se estivessem em uma situação real. Este software tem alguns detalhes que devem ser observados, como:

1. É necessária a configuração delas dentro do próprio GNS3, em que é possível especificara quantidade de placas de rede que as máquinas virtuais terão;
2. Para a criação de roteadores ou switches Cisco/Juniper virtuais, é necessária a imagem do sistema rodando no equipamento real (IOS). Para este trabalho não foi utilizada esta possibilidade por não possuir licensas destar imagens.

- VirtualBox: foi utilizado para a criação das máquinas virtuais, isso se deu por dois motivos principais: familiaridade com o software e também porque é compatível com o GNS3.

- OpenNHRP: Esta é uma implementação do protocolo NHRP em sistemas Linux. Sua configuração é feita através do arquivo opennhrp.conf, criado após a sua instalação. Esta ferramenta tem algumas particularidades a serem observadas:

  - A máquina que rodará este serviço deve estar utilizando o kernel 2.6.x ou superior. Porém, Algumas versões posteriores também não são compatíveis, como a 3.11 que foi testada em laboratório e o software não funcionou corretamente. No trabalho foi utilizada a versão 3.6.6 e detalhes de configuração são encontrados nos apêndices. A sua execução é feita através do comando abaixo.

  opennhrp *todo o log de mensagens será mostrado na tela ou*

  opennhrp -d *para executá-lo em segundo plano*

- IPSec:É a implementação do IPSec em sistemas Linux.

- Quagga: é um software abertoque provê a implementação de um roteador com linhas de comando no estilo Cisco em um sistema Linux. Com ele é possível a utilização de protocolos de roteamento como o RIP, OSPF e BGP. Ele proporciona também uma linha de comando similar a um roteador Cisco. Sua configuração depende de quais módulos serão utilizados.

  Cada um dos módulos precisa de um arquivo de configuração. Neste caso, foram utilizados o zebra e o ripd. O zebra é necessário para a configuração das interfaces através do Quagga e o ripd corresponde ao protocolo RIP, que foi o escolhido para o roteamento dentro da DMVPN. A princípio, o terminal do Quagga é dividido para cada módulo utilizado, como por exemplo as configurações relacionadas ao RIP ficarão em ripd.conf e relacionadas a configuração das interfaces em zebra.conf. Porém, é possível juntar os terminais em um só, tendo a experiência bastante parecida com um Cisco real. Isso é feito através do arquivo vtysh.conf.

3.3 Experimentos realizados

Primeiramente é apresentado um cenário em que tem-se duas máquinas se comunicando através de túneis GRE. No segundo cenário, é implantada uma DMVPN simples com um HUB e dois SPOKEs.

3.3.1 Cenário 1: Teste básico de construção de túnel utilizando o OpenNHRP

3.3.1.1 Objetivos do experimento

Este foi um cenário criado para um teste inicial do OpenNHRP. Nele, tem-se 2 máquinas virtuais conectadas a uma núvem simulando a internet. O objetivo foi a criação de um túnel GRE com criptografia IPSec usando o OpenNHRP. As duas máquinas possuem IPs públicos conhecidos por ambos os lados. O túnel tem um IP privado em cada *site* e usa os IPs públicos para o transporte dos pacotes desta rede privada. A figura 18 ilustra o cenário.

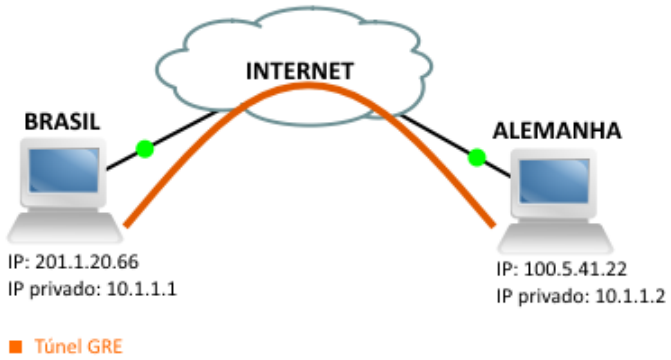


Figura 18: DMVPN - comunicação entre 2 SPOKEs

Foi utilizado um script em cada máquina para a criação e associação local dos túneis, que pode ser observado abaixo. Primeiramente é criado o túnel do tipo GRE com a chave IPSec escolhida. Em seguida é atribuido o IP privado do túnel e sua associação à interface física.

#!/bin/bash ip tunnel add gre1 mode gre key 1234 ttl 64

ip addr add 10.1.1.1/24 dev gre1

ip tunnel change gre1 local 201.1.20.66

ip link set gre1 up

O OpenNHRP foi utilizado apenas para mapear os endereços físico e lógico da outra máquina, sua configuração está abaixo, em que é mapeado o endereço IP lógico 10.1.1.2/24 para o endereço IP público 100.5.41.22.

interface gre1

map 10.1.1.2/24 100.5.41.22 register cisco

cisco-authentication 1234

redirect

holding-time 360

multicast dynamic

As configurações foram feitas através de um script inicializado com o sistema Linux, que cria os túneis, atribuindo-os um endereço IP privado e atrelando-os à interface de saída para a rede pública internet. É também especificada a chave IPSec utilizada.

Os testes realizados foram de conectividade. Foi possível o ping de um *site* à interface túnel do outro, como pode-se observar na figura 19.

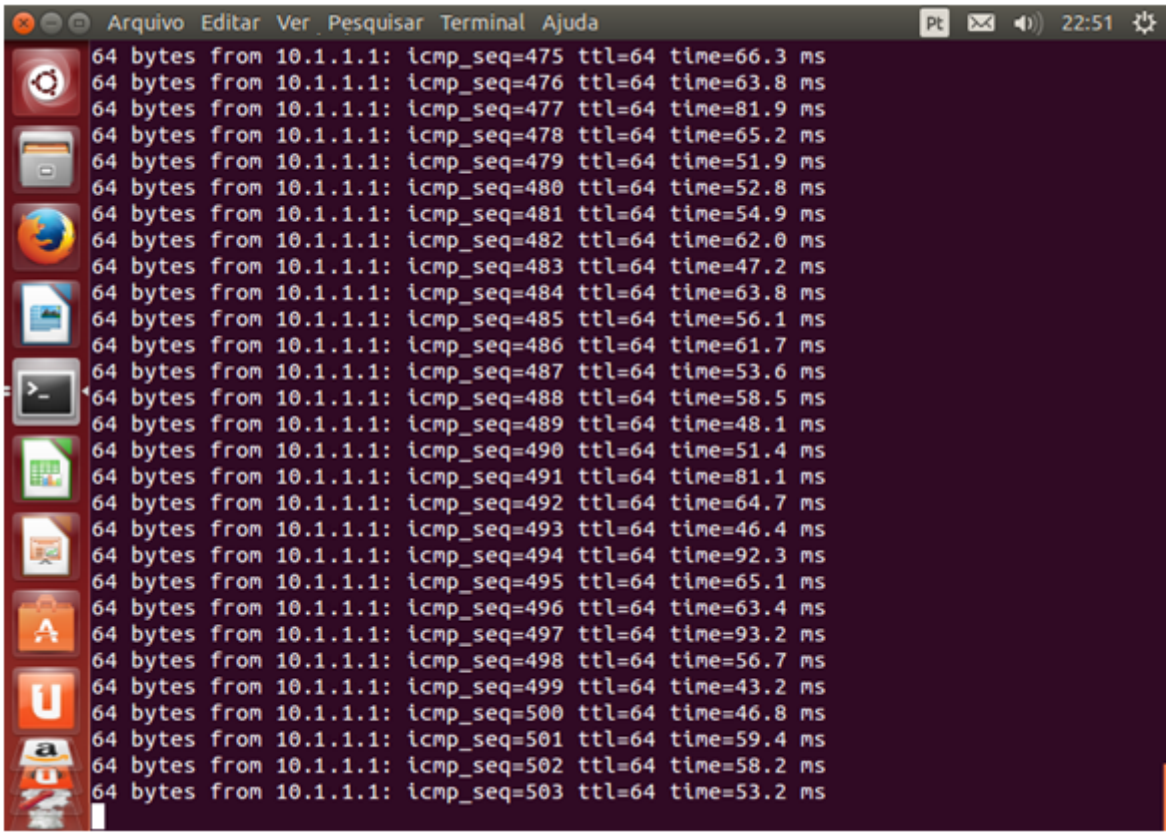


Figura 19: Ping para o IP do túnel de BRASIL.

3.3.2 Cenário 2: DMVPN com Linux

3.3.2.1 Objetivos

O objetivo deste experimento é contruir uma rede DMVPN simples com 2 SPOKEs e HUB, verificando se existe a formação dinâmica de túnel entre dois SPOKEs quando for solicitada uma comunicação entre eles. Note-se que entre SPOKEs e HUB são formados túneis estáticos.

O HUB tem o papel de servidor NHRP, enquanto os SPOKES são os clientes NHRP. No momento em que o SPOKE 1 gerar um pacote destinado ao endereço lógico do SPOKE 2, de forma similar ao ARP, o endereço físico do SPOKE 2 será resolvido pelo servidor NHRP. Túneis mGRE são criados dinamicamente entre o HUB e os SPOKEs, e o IPSec irá encriptar todos os pacotes que trafeguem entre os *sites*.

Deve-se observar que nos SPOKEs não é preciso ter um IP público estático, eles pode ser fornecido pela ISP (um acesso ADSL, um IP dedicado ou mesmo disponibilizado via 3G). Já o HUB precisa de um IP público de internet e estático, para os SPOKEs poderem encontrá-lo e formarem o túnel mGRE com ele. Este IP é contratado junto à uma ISP

e é imutável. Além dos IPs válidos, são necessários os IPs privados para a DMVPN, que devem fazer parte da mesma rede, por ex.: 10.0.0.0/24. Através do NHRP, o HUB associará cada IP privado ao IP válido do seu respectivo SPOKE. Cada *site* pode também ter outras LANs vinculadas à DMVPN, sendo necessária a criação das rotas estáticas em todos eles. A figura 20 ilustra o cenário apresentado. Cada máquina com um IP público e um privado e os túneis mGRE formados entre SPOKEs e o HUB de forma estática. A figura 21 ilustra o cenário durante a comunicação entre dois SPOKEs usando o túnel formado dinamicamente. Após o tempo de inatividade este túnel entre ARGENTINA e RÚSSIA é desfeito voltando ao que é ilustrado na figura 20.

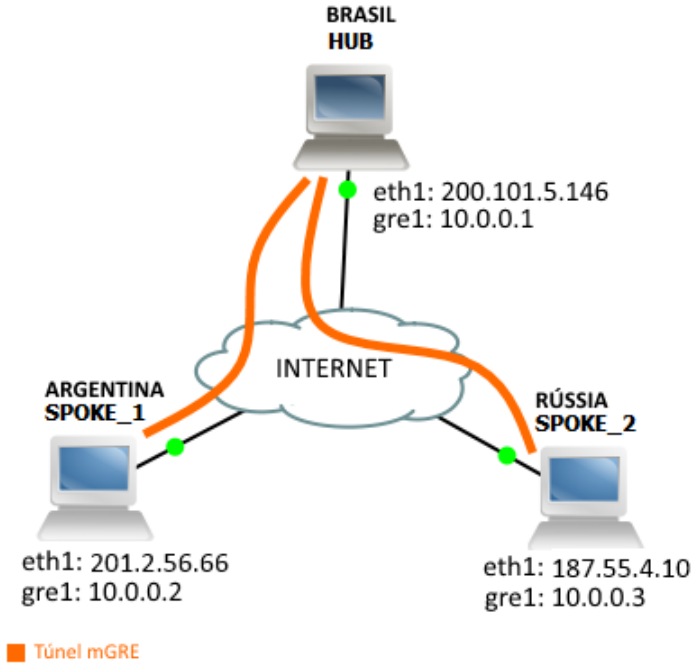


Figura 20: DMVPN com HUB e 2 SPOKEs. Túneis mGRE são permanentes.

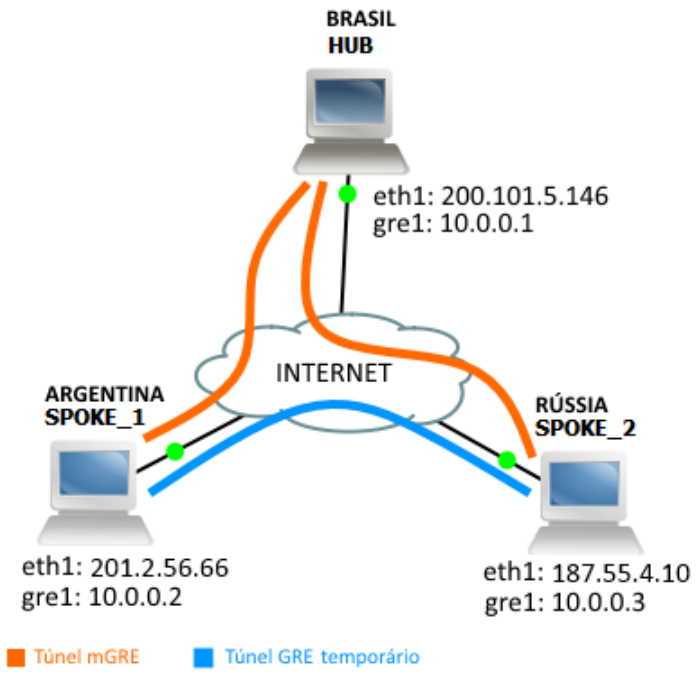


Figura 21: Túneis GRE entre os SPOKEs são criados durante uma comunicação entre eles.

O cenário funciona da seguinte forma: Túneis mGRE são criados entre o HUB e SPOKEs, mas não entre os SPOKEs diretamente. Após se registrarem com o HUB via NHRP, este passa a ter conhecimento dos IPs de todos os SPOKEs com ele registrados. Quando um SPOKE "A"deseja se comunicar com um SPOKE "B", este pergunta ao HUB o endereço IP válido do SPOKE destino. Com a resposta, SPOKE origem e destino formam um túnel mGRE dinamicamente entre eles e iniciam a comunicação. Após um tempo de inatividade definido, o túnel é desfeito.

3.3.2.2 Configurações realizadas

Para o funcionamento do cenário foram necessárias 3 máquinas virtuais rodando Ubuntu 13.04. As máquinas foram conectadas a uma núvem simulando a rede internet, como ilustrado na figura 20.

A figura 23 ilustra o cenário montado já no GNS3. Na figura 22 é mostrado um detalhe de configuração das máquinas virtuais no GNS3, em que é necessário escolher e atribuir os nomes que elas terão no software. No campo "NICs"é escolhido quantas interfaces Ethernet a máquina virtual terá no GNS3.

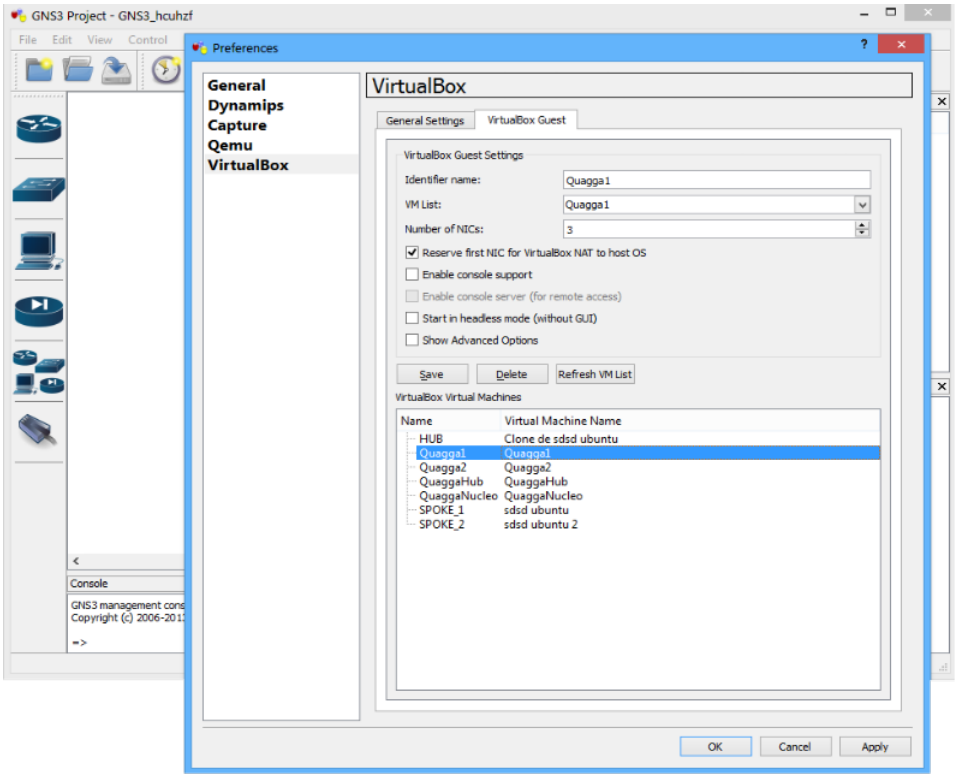


Figura 22: Configurando o VirtualBox no GNS3

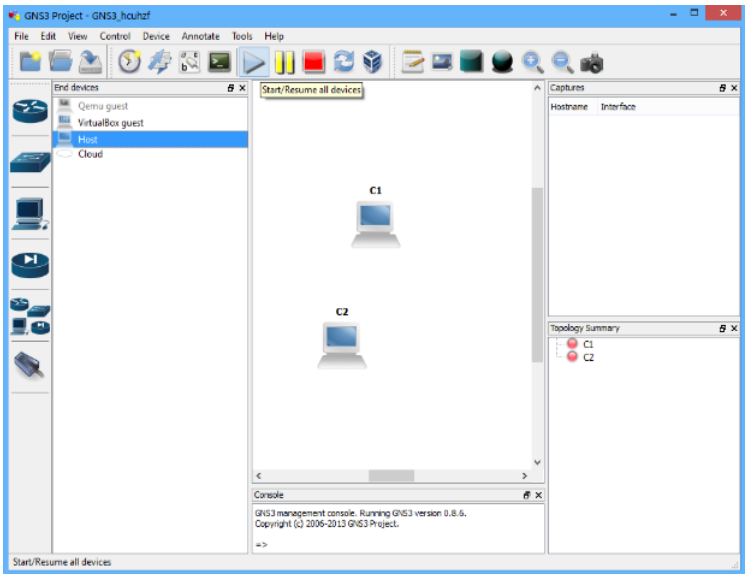


Figura 23: Configurando o VirtualBox no GNS3

Em todas as 3 máquinas foram instalados os softwares opennhrp e IPSecTools. A figura 24 compara os arquivos de configuração do opennhrp para o HUB e para um SPOKE. Nele deve ser especificada a interface que fará o tunel mGRE. Pode-se observar que no HUB não existe nenhum mapeamento em relação aos SPOKEs. Já no SPOKE é indicado o endereços físico e lógico do HUB. Em ambos, é configurada a chave IPSec

escolhida e o tempo de duração do cache das rotas recebidas. No caso do HUB, não é necessário o mapeamento para chegar nos SPOKEs, isso é feito dinamicamente.

```
emerson@emerson-VirtualBox:~$ cat /etc/opennhrp/opennhrp.conf
interface gre1
      cisco-authentication 1234
      holding-time 5
```

HUB

```
emerson@emerson-VirtualBox:~$ cat /etc/opennhrp/opennhrp.conf
interface gre1
      map 10.0.0.1/24 172.0.0.2 register cisco
      cisco-authentication 1234
      holding-time 5
```

SPOKE

Figura 24: Arquivo de configuração do opennhrp no HUB e SPOKEs

O IPSecTools tem 2 arquivos de configuração. O primeiro é o psk.txt, onde é informado de quais IPs serão aceitas determinadas chaves IPSec. A figura 25 mostra um modo genérico dessa configuração, em que será aceita a chave 1234 de todos os IPs. Esse arquivo é configurado em todos os integrantes da DMVPN.

```
root@emerson-VirtualBox:/home/emerson# cat /etc/racoon/psk.txt
*       1234
```

Figura 25: Arquivo de configuração do IPSecTools no HUB e SPOKEs

O outro arquivo de configuração do IPSec é o racoon.conf. Nele é especificado onde se encontra o arquivo de chaves, que neste caso é o psk.txt. Há também regras que podem ser colocadas nele, porém foi utilizada uma configuração genérica. O foco do trabalho não foi o IPSec, portanto não serão discutidos maiores detalhes sobre o mesmo. As configurações se encontram nos apêndices.

O kernel do sistema Linux precisou ser modificado para a versão 3.6.6, com fins de compatibilidade com o opennhrp. As interfaces de rede utilizadas das máquinas virtuais foram configuradas com IP estático, caso contrário o próprio sistema causaria problemas.

Além destas modificações, foi necessária a criação dos túneis mGRE. Na criação dos túneis, é especificada a chave IPSec, o IP privado do site na DMVPN e também o IP válido por onde sairá o tráfego. No cenário isso foi criado de forma automática através de um script rodando junto à inicializacão do sistema, que é mostrado na figura 26.

```
root@emerson-VirtualBox:/home/emerson# cat /etc/init.d/gre1_up
#!/bin/bash
ip tunnel add gre1 mode gre key 1234 ttl 64
ip addr add 10.0.0.2/24 dev gre1
ip tunnel change gre1 local
ip link set gre1 up 201.2.56.66
```

Figura 26: Script configurando a interface túnel

3.3.2.3 Testes e trocas de mensagens

A análise de trocas de mensagens deste cenário será realizada em duas etapas. A primeira etapa analisa as mensagens trocadas no momento em que as máquinas são ativadas e o openhhrp é executa e, em uma segunda etapa, a troca de mensagens durante um teste de conectividade entre os IPs lógicos.

Etapa 1 Em um primeiro momento os SPOKEs não se comunicam diretamente, então as primeiras mensagens trocadas foram pedidos de registro NHRP dos SPOKEs com o HUB. A figura 27 mostra o que foi mostrado na tela do SPOKE ARGENTINA no momento do pedido de registro com o HUB. As numerações feitas estão relacionadas com o diagrama de troca de mensagens ilustrado na figura 28. Primeiramente é criado o túnel mGRE entre o HUB e o SPOKE (msg 1), neste momento não há nenhuma troca de mensagens. Em seguida o SPOKE envia um pedido de registro NHRP ao HUB (msg 2). O HUB analisa o pedido e a chave IPSec e envia a confirmação de registro ao SPOKE (msgs 3 e 4). O SPOKE verifica a resposta e a chave IPSec (msg 5). Finalmente, a sessão NHRP é estabelecida entre HUB e SPOKE (msg 6).

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# opennhrp
opennhrp[2507]: OpenNHRP 0.14.1 starting
opennhrp[2507]: Interface lo: configured UP, mtu=0
opennhrp[2507]: Interface eth0: configured UP, mtu=1500
opennhrp[2507]: Interface eth1: configured UP, mtu=1500
opennhrp[2507]: Interface eth2: configured UP, mtu=1500
opennhrp[2507]: Interface gre0: config change, mtu=1476
opennhrp[2507]: Interface gretap0: config change, mtu=1476
opennhrp[2507]: Interface gre1: configured UP, mtu=1472
opennhrp[2507]: Interface gre1: GRE configuration changed. Purge
d 1 peers.
opennhrp[2507]: Filter code installed (19 opcodes)
opennhrp[2507]: Interface gre1: config change, mtu=1472
Create link from 10.0.0.3 (187.55.4.10) to 10.0.0.1 (200.101.5.146)
VPN connexion established
Phase 2 established 187.55.4.10[500] -> 200.101.5.146[500]
opennhrp[2507]: Sending Registration Request to 10.0.0.1 (my mtu
=0)     2
opennhrp[2507]: Received Registration Reply from 10.0.0.1: succe
ss      5
opennhrp[2507]: Sending Purge Request (of local routes) to 10.0.
0.1
```

Figura 27: Mensagens de inicio de sesão opennhrp (log opennhrp de um dos SPOKEs)

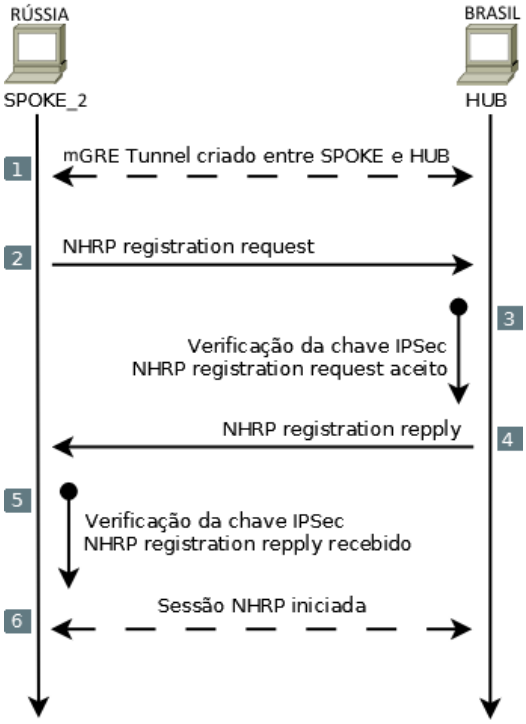


Figura 28: Diagrama das mensagens de inicio de sesão opennhrp

Etapa 2 Em um segundo momento foi observado o log de mensagens do opennhrp no SPOKE ARGENTINA no momento em que foi executado um ping com o destino SPOKE RUSSIA. Conforme a figura 29 pode-se observar a tentativa de resolução do IP destino, em que é enviado ao HUB um pedido de resolução (msg 3). Pode-se observar também o recebimento desta resolução, em que 10.0.0.2 é acessível pelo IP público 201.2.56.66. Como o ping tem um pedido reverso, o SPOKE origem também resolve o seu endereço para o HUB.

```
root@emerson-VirtualBox: /home/emerson
opennhrp[2507]: Received Registration Reply from 10.0.0.1: succe
ss
opennhrp[2507]: Sending Purge Request (of local routes) to 10.0.
0.1
opennhrp[2507]: Sending Registration Request to 10.0.0.1 (my mtu
=0)
opennhrp[2507]: Received Registration Reply from 10.0.0.1: succe
ss
opennhrp[2507]: Sending Resolution Request to 10.0.0.2    3
opennhrp[2507]: Received Resolution Reply 10.0.0.2/32 is at prot
o 10.0.0.2 nbma 201.2.56.66                                  6
Create link from 10.0.0.3 (187.55.4.10) to 10.0.0.2 (201.2.56.66)
VPN connexion established
Phase 2 established : 187.55.4.10[500] -> 201.2.56.66[500]7
opennhrp[2507]: Received Resolution Request from proto src 10.0.
0.2 to 10.0.0.3
opennhrp[2507]: Sending Resolution Reply 10.0.0.3/32 is-at 187.
55.4.10 (holdtime 360)
opennhrp[2507]: Sending Registration Request to 10.0.0.1 (my mtu
=0)
opennhrp[2507]: Received Registration Reply from 10.0.0.1: succe
ss
```

Figura 29: Log opennhrp do SPOKE ARGENTINA no momento de um ping entre os dois SPOKEs

Foi observado que no HUB, ele encaminha este pedido da origem ao destino, como pode ser observado na figura 30 (msg 4). Este comportamento não era exatamente o esperado, pois à princípio o HUB deveria responder diretamente à este pedido, já que ele conhece todos os SPOKEs. E isso acontece também com o pedido reverso do ping, em que o HUB encaminha a resolução da origem para o destino (msg 6).

```
root@emerson-VirtualBox: /home/emerson
opennhrp[2649]: Sending Registration Reply from proto src 10.0.0
.1 to 10.0.0.2 (1 bindings accepted, 0 rejected)
opennhrp[2649]: Forwarding packet from nbma src 187.55.4.10, proto
 src 10.0.0.3 to proto dst 10.0.0.2, hop count 16
opennhrp[2649]: Forwarding packet from nbma src 187.55.4.10, proto
 src 10.0.0.3 to proto dst 10.0.0.2, hop count 16      4
opennhrp[2649]: Forwarding packet from nbma src 201.2.56.66, proto
 src 10.0.0.2 to proto dst 10.0.0.3, hop count 16
opennhrp[2649]: Forwarding packet from nbma src 201.2.56.66, proto
 src 10.0.0.2 to proto dst 10.0.0.3, hop count 16      6
opennhrp[2649]: Received Registration Request from proto src 10.
0.0.3 to 10.0.0.1
opennhrp[2649]: Sending Registration Reply from proto src 10.0.0
.1 to 10.0.0.3 (1 bindings accepted, 0 rejected)
opennhrp[2649]: Received Registration Request from proto src 10.
0.0.2 to 10.0.0.1
opennhrp[2649]: Sending Registration Reply from proto src 10.0.0
.1 to 10.0.0.2 (1 bindings accepted, 0 rejected)
opennhrp[2649]: Received Registration Request from proto src 10.
0.0.3 to 10.0.0.1
opennhrp[2649]: Sending Registration Reply from proto src 10.0.0
.1 to 10.0.0.3 (1 bindings accepted, 0 rejected)
opennhrp[2649]: Received Registration Request from proto src 10.
```

Figura 30: Log opennhrp do HUB no momento de um ping entre os dois SPOKEs

Como pode-se observar na figura 31, após a sessão NHRP ser estabelecida entre os SPOKEs e o HUB (msg 1), o ping com destino SPOKE RUSSIA foi executado no SPOKE ARGENTINA (msg 2). Neste momento foi enviado um pedido de resolução de IP para chegar em RUSSIA ao HUB (msg 3). O HUB confirma se o SPOKE RUSSIA está acessível encaminhando o pedido de resolução à ele (msg 4). O SPOKE RUSSIA responde a solicitação ao HUB (msg 5), esta resposta é encaminhada ao SPOKE ARGENTINA (msg 6). Com a resposta, os SPOKEs fecham um túnel GRE entre eles e iniciam a comunicação (msg 7). Após um tempo de inatividade definido (msg 8), o túnel GRE entre os SPOKEs é desfeito (msg 9).

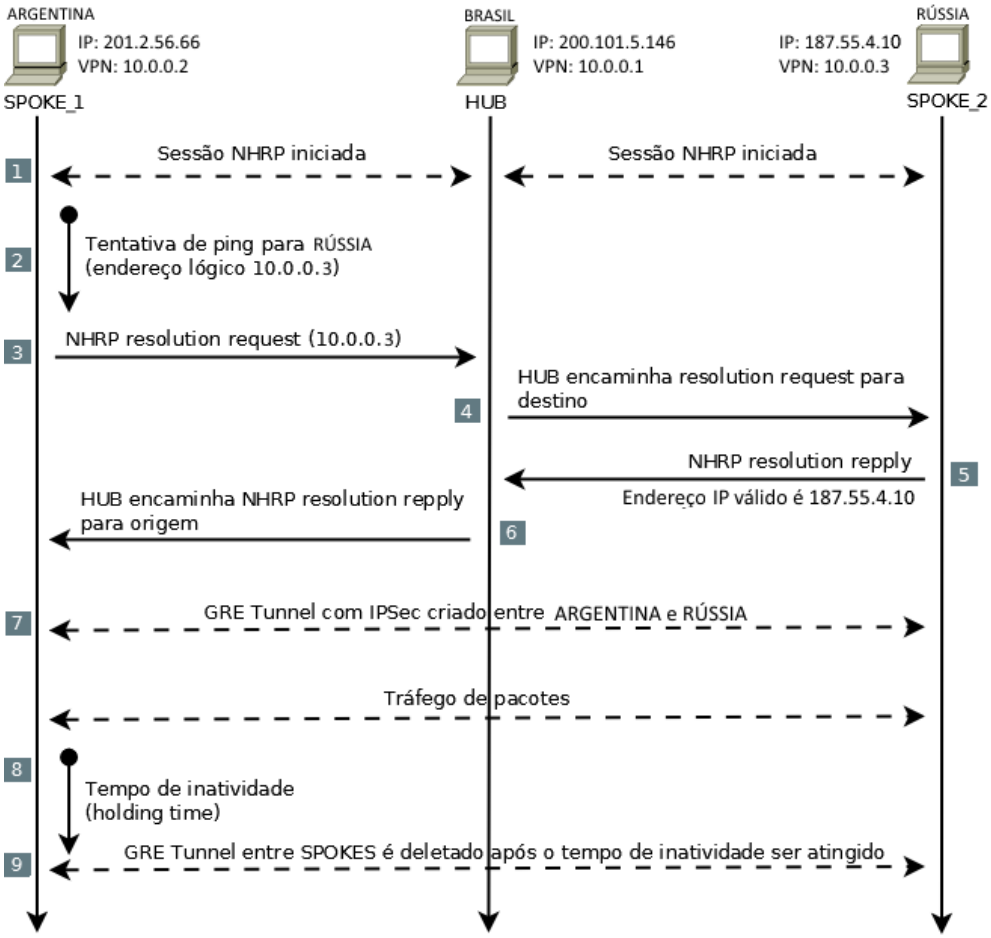


Figura 31: Log opennhrp do HUB no momento de um ping entre os dois SPOKEs

3.4 Conclusão

Neste capítulo foram relatados os experimentos básicos para testes da DMVPN com OpenNHRP. Os cenários investigados se comportaram conforme esperado, com os túneis se formando dinamicamente entre os SPOKEs. Foi observada uma indireção do NHRP durante a resolução dos endereços lógicos dos clientes, mas que não afetou o comportamento do cenário como um todo.

No próximo capítulo será feita uma análise do uso de um protocolo de roteamento dinâmico, o RIP, na construção de um modelo de redundância com DMVPNs.

4 Utilização de uma DMVPN atuando como redundância de uma VPN baseada em VRF

4.1 Introdução

Em um contexto de redes corportativas, é necessária uma alta confiabilidade nos pontos de acesso. Essa confiabilidade muitas vezes é obtida pelo uso de links de acesso redundantes. Neste capítulo será apresentado um breve estudo sobre redundâncias e uma discussão sobre a utilização de uma DMVPN servindo como um link redundante de uma VPN baseada em VRF.

4.2 Redundância em VPNs

(PINHEIRO, 2004)O termo redundância descreve a capacidade de um sistema em superar a falha de um de seus componentes através do uso de recursos redundantes, ou seja, um sistema redundante possui um segundo dispositivo que está imediatamente disponível para uso quando da falha do dispositivo primário do sistema. O que ocorre também no caso de redes de computadores, em que pode-se ter um acesso principal, por onde passará todo o tráfego, e um secundário, que ficará em *standby* até ocorrer alguma falha no acesso principal. Quando isso ocorrer, o acesso secundário deve assumir o tráfego para não haver perda de informações.

Um link redundante de uma VPN baseada em VRF muitas vezes não é viável devido aos custos, considerando por exemplo uma empresa em seu início. Muitas vezes também, não são necessárias todas as vantagens e confiabilidade do link principal em um link redundante, pois ele servirá somente para não perder totalmente a conectividade do link. Uma alternativa para estes casos, é o uso de DMVPNs, que possibilitam a formação de uma VPN redundante com menor custo de implementação.

O conceito de DMVPN foi introduzido pela Cisco e pode ser implementado fazendo uso dos roteadores que têm esse suporte. Além disso, foi mostrado que também é possível formar uma DMVPN fazendo uso de softwares livres e protocolos *opensource* que são capazes de atuar em conjunto com roteadores Cisco.

Uma DMVPN formada com a finalidade de atuar como link redundante de uma VPN baseada em VRFs seria feita partindo do princípio de que as falhas serão tratadas no CE. Isso ocorre porque uma VPN e uma DMVPN necessitam de tipos de serviços totalmente diferentes. Enquanto em uma VPN baseada em VRF são providos links da rede VPN da ISP, em uma DMVPN são necessários links com acesso à Internet, sendo estas duas redes isoladas uma da outra. A imagem 32 ilustra, de forma simplificada, como seria esta situação em um dos *sites* participantes, em que se tem o link redundante formado com uma DMVPN.

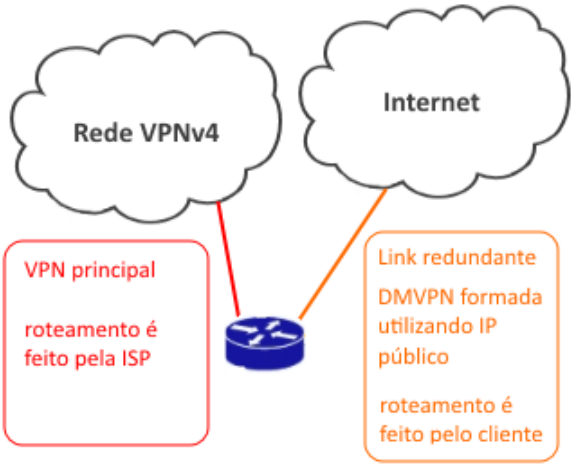


Figura 32: Forma simplificada de uma redundância com DMVPN.

Sendo assim, é possível a implementação dessa redundância fazendo uso de uma DMVPN. Considerando o cliente TABAJARA, ilustrado na figura 33, em que ele tem uma VPN contratada com uma ISP interligando 3 *sites*. Cada *site* tem uma rede local atrelada ao roteador do cliente, que nesse caso seria uma máquina Linux. Além desta VPN, o cliente contratou também, em cada ponto, links com IPs públicos, que serviram para a formação da DMVPN. Como a VPN e os links IP estão em redes distintas, o CE necessita realizar um tratamento do roteamento. Isso poderia ser feito de forma simples utilizando rotas estáticas, que teriam uma menor preferência de saída pela DMVPN. Assim, enquanto o link VPN estiver acessível, ele será o preferencial. No caso de uma falha, a rota estática secundária assumiria o tráfego, passando o acesso para a DMVPN. Ao ficar ativo novamente, o link principal reassumiria o tráfego.

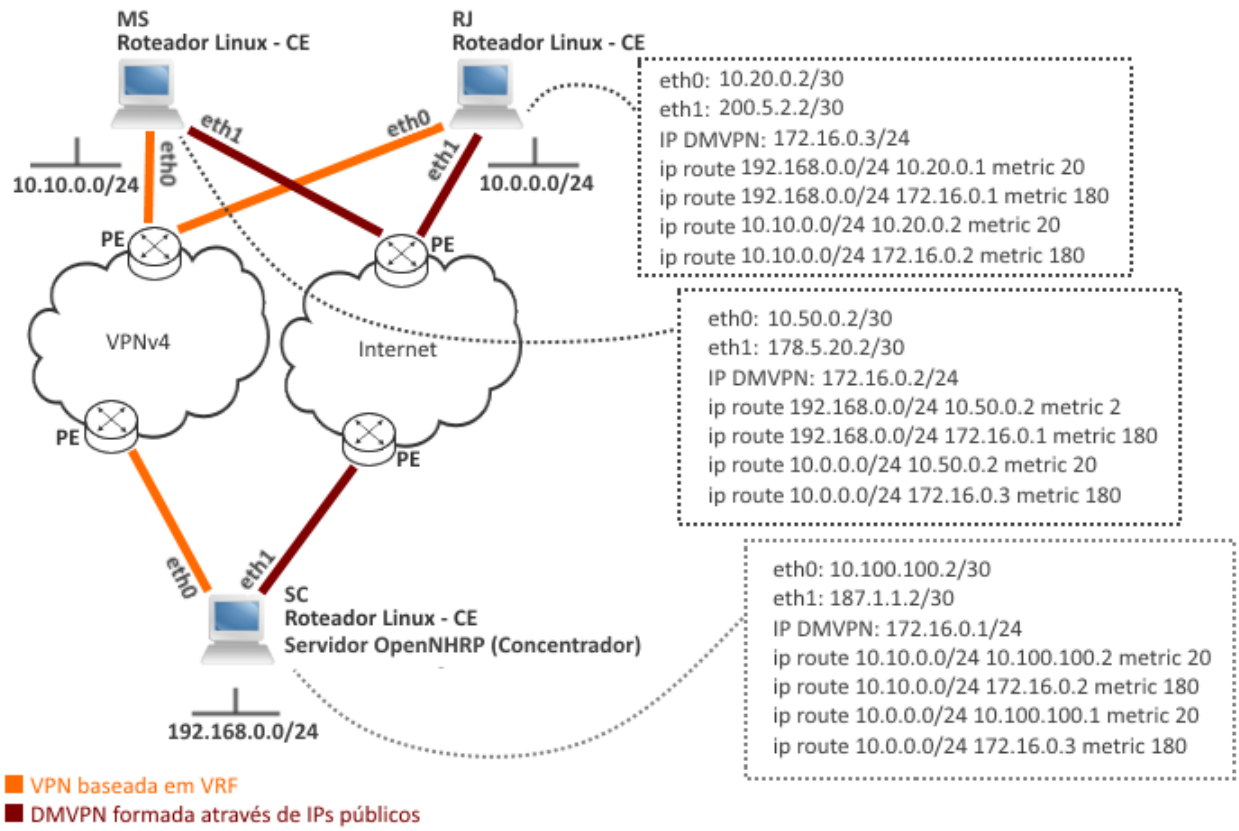


Figura 33: Cliente TABAJARA com uma redundância DMVPN simples utilizando rotas estáticas.

O uso de rotas estáticas não é o ideal, devido principalmente à escalabilidade. A distribuição de rotas poderia ser feita no CE através de um protocolo de roteamento dinâmico. Isso possibilitaria que o tráfego ocorresse por um link secundário quando o link principal não estivesse acessível.

Como protocolo de roteamento dinâmico, foi utilizado o RIP em uma topologia simples que é mostrada na figura 34, em que se tem dois *sites* interligados a uma núvem. Um deles tem dois links de acesso para esta núvem, como é possivel acompanhar também na figura 34. O protocolo foi configurado para dar uma maior preferência de saída pelo link PRINCIPAL. Ao desativar este link manualmente, alguns pocotes são perdidos, e em seguida, o link SECUNDÁRIO assume o tráfego. Ao reativar o link PRINCIPAL, o tráfego passa a sair por ele novamente.

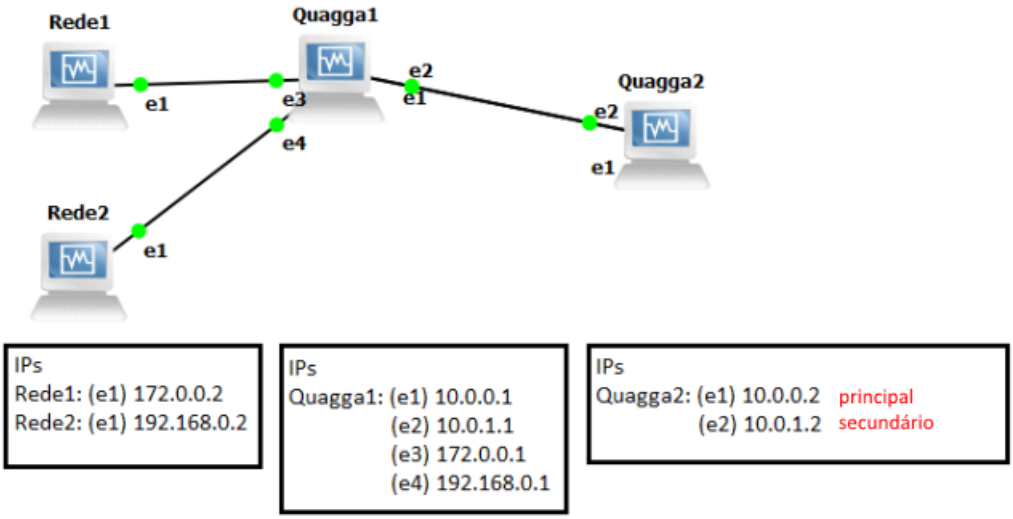


Figura 34: Cenário para teste de troca de acesso utilizando RIP.

Um dos objetivos deste trabalho era de avaliar o uso de diversos protocolos de roteamento dinâmico com a DMVPN como redundância. Foi elaborado um laboratório prelimiar com RIP que será apresentado na sequência.

4.3 Avaliação do RIP como protocolo de roteamento

4.3.1 Objetivos do experimento

Este cenário reproduz a topologia do cenário 2 do capítulo anterior, porém é acrescentado o RIP (MALKIN, 1998), um protocolo de roteamento dinâmico implementado pelo software Quagga. O objetivo deste experimento foi de identificar possíveis problemas envolvidos no uso de um protocolo de roteamento dinâmico no contexto da DMVPN. Note-se que o uso do protocolo RIP facilita a construção das tabelas de roteamento em sistemas maiores e pode vir no auxílio de uma possível integração de uma outra VPN na perspectiva de implementação de tolerância a falhas. Para a realização do experimento também foram acrescentadas interfaces loopback para simular redes locais nos sites dos clientes.

Tendo a DMVPN em funcionamento, foi instalado o Quagga em todos os *sites* participantes da DMVPN. Ele irá configurar automaticamente as interfaces de rede e utilizará o RIPv2 como protocolo de roteamento. Todas as LANs escolhidas nos *sites* serão redistribuídas dentro da DMVPN. A figura 35 ilustra o cenário que será investigado. Pode-se observar, por exemplo, que no SPOKE ARGENTINA existe uma rede local simulada

por uma interface loopback com endereço 22.22.22.22/32, sendo que este endereço deverá ser divulgado para os demais roteadores. Mantém-se o túnel mGRE associado à rede 10.0.0.0/24 e a interface física deste roteador é a eth1 com o IP público 201.2.56.66.

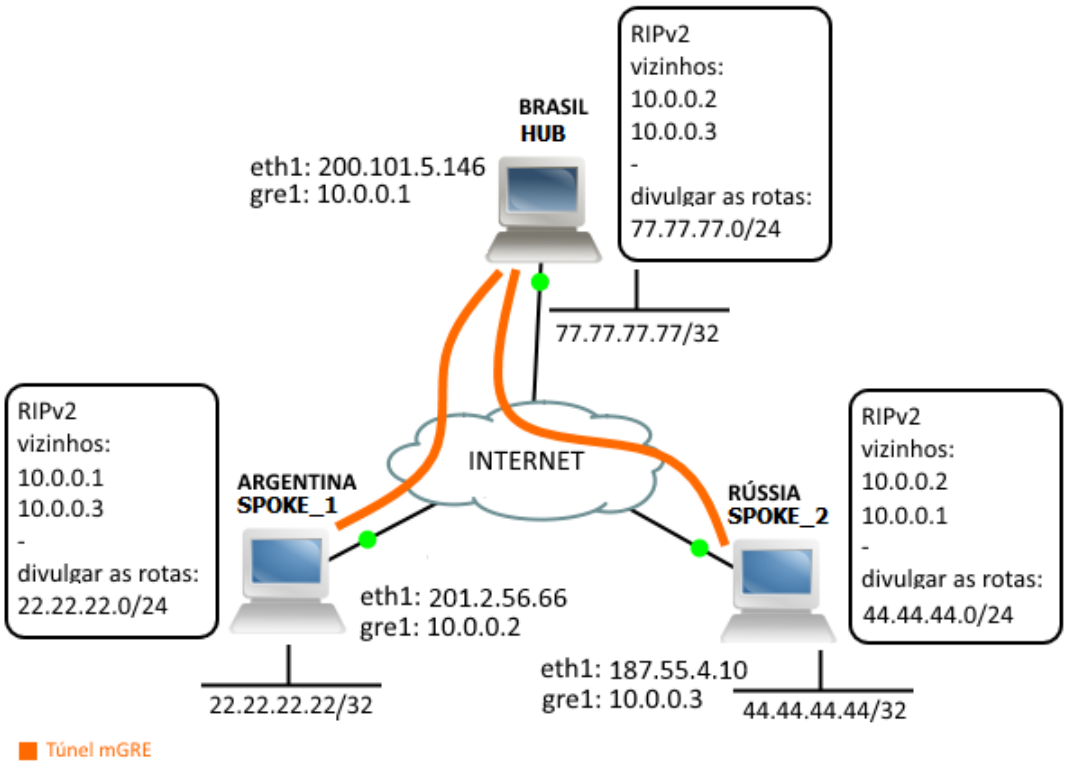


Figura 35: DMVPN em conjunto com Quagga. As rotas são divulgadas para os outros *sites* via RIP.

Dentro do cenário apresentado pretendeu-se inicialmente habilitar a divulgação das rotas unicamente no túnel, sendo que as redes a serem divulgadas são aquelas associadas aos *sites* do cliente. No entanto,em um primeiro teste, observou-se que o mGRE aparentemente não suportou o *multicast* que era esperado. Neste sentido, investigou-se uma alternativa para efetivar a operação do RIP. A solução encontrada foi o uso do comando *neighbor*, que permite a especificação do IP lógico de cada *site* remoto. Esta solução trouxe um problema de escalabilidade, uma vez que os endereços lógicos de todos os sites devem ser especificados com este comando. Na sequência é descrito o teste utilizando o comando *neighbor*.

4.3.2 Configurações e testes preliminares usando o multicast do mGRE

A topologia do cenário 2 do capítulo anterior foi utilizada para efetuar testes de multicast dos túneis mGRE. Os testes de conectividade obtiveram sucesso, porém a divulgação das rotas não ocorreu conforme esperado. Utilizar o protocolo RIP visou a automatização no roteamento da DMVPN. Foi notado que nos testes, a divulgação de rotas não estava acontecendo devido os roteadores da DMVPN estarem divulgando suas rotas para o seu vizinho, que no caso seria um roteador da ISP em que não se tem controle. Para um melhor entendimento, a figura 36 ilustra este problema, em que tem-se os roteadores de borda da ISP sendo, por padrão, vizinhos naturais dos participantes da DMVPN.

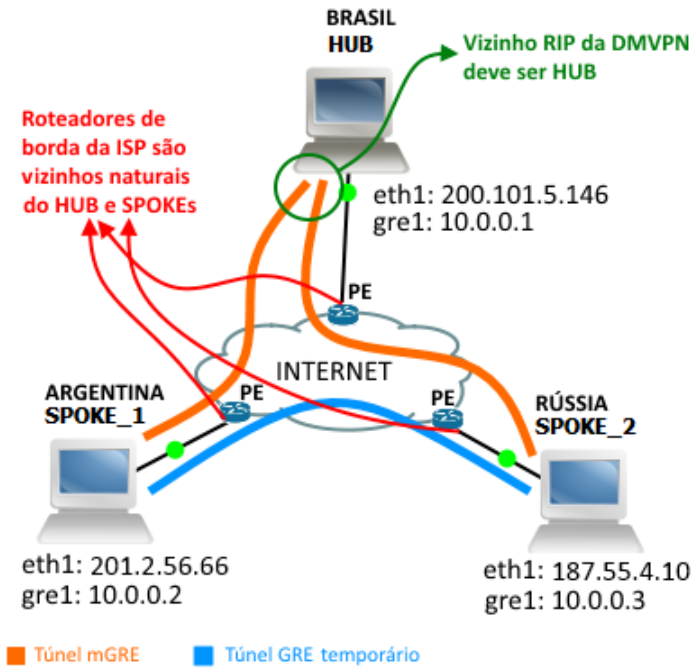


Figura 36: Identificação dos vizinhos RIP

Para contornar este problema, foi necessário especificar quais seriam os vizinhos RIP, que à princípio, seria o IP da interface túnel do HUB 10.0.0.1. A especificação de vizinhos é possivel conforme ilustrado na figura 37, em que o IP túnel do HUB é o vizinho. Ao receber as rotas,o HUB as divulgaria para os seus SPOKEs via atualizações de roteamento. Porém isso não ocorreu, as rotas eram divulgadas para o HUB, mas ele não as repassava para os SPOKEs. Com isso, foi confirmado que o multicast não ocorreu nos túneis mGRE, que não era o esperado. Conforme a figura 38 é possível acompanhar o resultado esperado, em que os SPOKEs aprendiam as redes uns dos outros através do vizinho HUB e, abaixo está o resultado na prática, em que os SPOKEs não aprenderam as redes um do outro.

```
router rip
 version 2
 timers basic 5 8 10
 network 44.44.44.0/24
 neighbor 10.0.0.1   especificando o HUB como vizinho
!
```

Figura 37: Especificação de vizinhos no roteador Quagga.

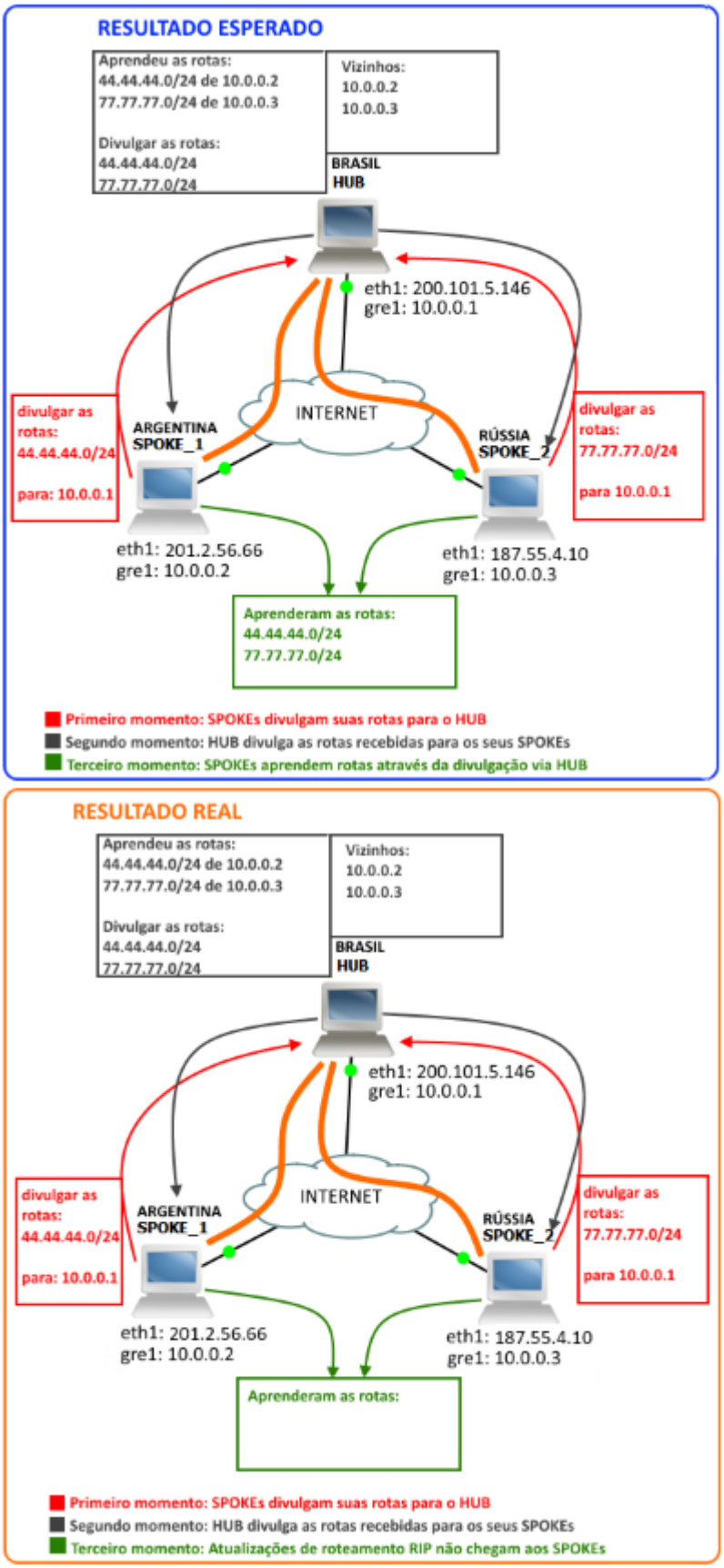


Figura 38: Resultado esperado do RIP x Resultado real obtido.

4.3.3 Configurações e testes usando o comando *neighbor*

A solução encontrada para o funcionamento com o RIP foi de especificar todos os vizinhos em todos os participantes da DMVPN. Conforme é ilustrado na figura 39, todos os participantes precisam conhecer todos os IPs privados da DMVPN. Desta forma, a automatização do roteamento não ficou ideal, porém auxilia no roteamento das outras redes atreladas aos participantes. Não foi possível a avaliação com roteadores Cisco devido a falta de licensas para uso de IOS.

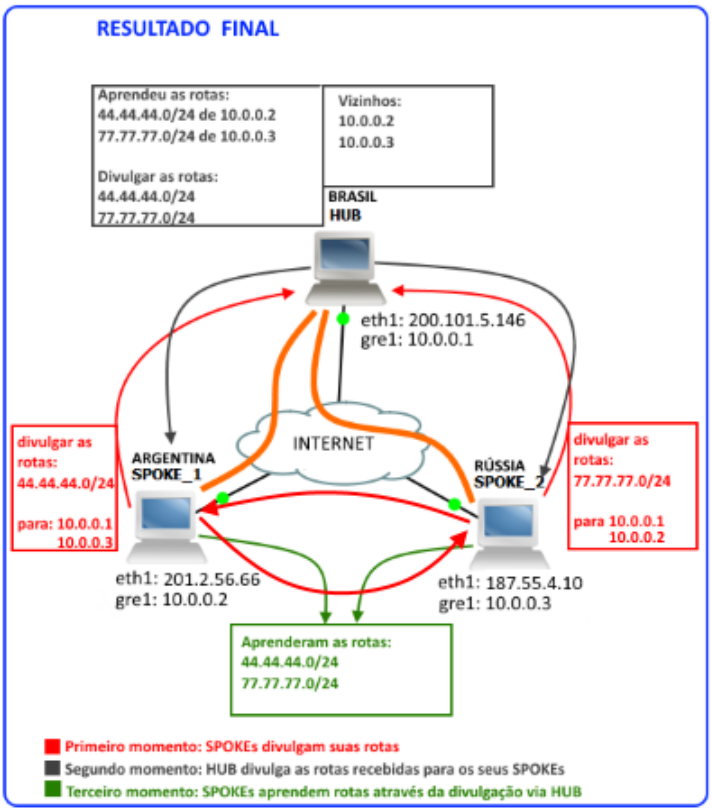


Figura 39: Configuração final RIP.

4.3.3.1 Configurações realizadas

Para o funcionamento do Quagga, foi necessária a instalação e a configuração do mesmo. Os serviços que o Quagga oferece são divididos em módulos, e a escolha de quais módulos serão utilizados é feita através do arquivo daemons. A quantidade de arquivos de configuração depende de quais foram ativados. Para este cenário, foram utilizados os módulos zebra e ripd, então a configuração foi feita em 2 arquivos: zebra.conf, ripd.conf. Além destes 2, também foi modificado o arquivo vtysh.conf, onde é configurado o usuário e senha do roteador Quagga.

Tanto no sistema Linux como no roteador Quagga é necessário ativar o "ip forwarding", que possibilita encaminhamento de pacotes IP. No linux isso é feito no arquivo /proc/sys/net/ipv4/ip_forward", mudando o seu valor para 1.

No arquivo zebra.conf foram especificadas as configurações das interfaces de rede, como por exemplo atribuindo-se o endereço IP. Portanto, os IPs passaram a ser configurados através do zebra, que é mostrado na figura 40. O "ip forwarding"também é configurado neste arquivo, como também as rotas estáticas.

```
interface eth2
 ip address 187.55.4.10/29
 multicast
 ipv6 nd suppress-ra
!
interface gre1
 multicast
 ipv6 nd suppress-ra
!
interface lo
 ip address 22.22.22.22/32
!
ip route 0.0.0.0/0 187.55.4.9
!
ip forwarding
!
line vty
```

Figura 40: Partes relevantes do arquivo zebra.conf do SPOKE RÚSSIA.

No segundo, ripd.conf, foram especificadas as configurações relacionadas ao RIP. É possível configurar a versão do protocolo, quais redes serão divulgadas e podem também ser especificados os vizinhos RIP. Neste caso, foi necessária a especificação dos vizinhos como sendo os IPs dos *sites* participantes da DMVPN através do comando *neighbor*, evitando-se utilizar a divulgação para o IP físico da ISP. A figura 41 mostra o arquivo ripd.conf no SPOKE RUSSIA e detalhamento de cada linha. Embora não tenha sido implementado, seria mais adequado especificar no comando network, a interface do cliente como sendo passiva.

```
router rip
 version 2
 timers basic 5 8 10
 network 44.44.44.0/24
 neighbor 10.0.0.1
 neighbor 10.0.0.3
!
```

Figura 41: Arquivo ripd.conf do SPOKE RÚSSIA.

No útimo arquivo, vtysh.conf, o Quagga foi configurado para guardar as configurações

em arquivos separados para melhor entendimento. Caso contrário, zebra e ripd estariam juntos em um só arquivo. Através do vtysh.conf também é configurado o usuário e senha do roteador Quagga.

4.3.4 Testes realizados

A divulgação de rotas com o comando neighbor obteve sucesso, a figura 42 mostra o console do roteador quagga ao verificar o status do RIP. É possível verificar as rotas recebidas [1], quais redes estão sendo distribuídas [2] e quais os vizinhos RIP [3].

```
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
C(i) 22.22.22.22/32     0.0.0.0               1 self              0
R(n) 44.44.44.44/32     10.0.0.3              2 10.0.0.3          0 00:03
R(n) 77.77.77.77/32     10.0.0.1              2 10.0.0.1          0 00:06
quagga-router# sho ip rip status
Routing Protocol is "rip"
  Sending updates every 5 seconds with +/-50%, next due in 3 seconds
  Timeout after 8 seconds, garbage collect after 10 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing:
  Default version control: send version 2, receive version 2
    Interface        Send  Recv   Key-chain
    lo               2     2
  Routing for Networks:
    22.22.22.0/24
    10.0.0.1
    10.0.0.3
  Routing Information Sources:
    Gateway          BadPackets BadRoutes  Distance Last Update
    10.0.0.1                 0         0       120   00:00:01
    10.0.0.3                 3         0       120   00:00:03
  Distance: (default is 120)
quagga-router#
```

Figura 42: Resultado dos comandos "show ip rip"e "show ip rip status"no quagga router do SPOKE ARGENTINA

Além disso, um teste de ping foi realizado partindo do SPOKE ARGENTINA tendo como destino um dos IPs divulgados através do RIP pelo SPOKE RÚSSIA. A figura 43 mostra este momento.

```
64 bytes from 44.44.44.44: icmp_seq=20 ttl=64 time=40.0 ms
64 bytes from 44.44.44.44: icmp_seq=21 ttl=64 time=44.0 ms
64 bytes from 44.44.44.44: icmp_seq=22 ttl=64 time=40.0 ms
64 bytes from 44.44.44.44: icmp_seq=23 ttl=64 time=60.0 ms
64 bytes from 44.44.44.44: icmp_seq=24 ttl=64 time=56.0 ms
64 bytes from 44.44.44.44: icmp_seq=25 ttl=64 time=52.0 ms
64 bytes from 44.44.44.44: icmp_seq=26 ttl=64 time=48.0 ms
64 bytes from 44.44.44.44: icmp_seq=27 ttl=64 time=44.0 ms
64 bytes from 44.44.44.44: icmp_seq=28 ttl=64 time=44.0 ms
64 bytes from 44.44.44.44: icmp_seq=29 ttl=64 time=60.0 ms
64 bytes from 44.44.44.44: icmp_seq=30 ttl=64 time=56.0 ms
64 bytes from 44.44.44.44: icmp_seq=31 ttl=64 time=64.0 ms
64 bytes from 44.44.44.44: icmp_seq=32 ttl=64 time=60.0 ms
64 bytes from 44.44.44.44: icmp_seq=33 ttl=64 time=48.0 ms
64 bytes from 44.44.44.44: icmp_seq=34 ttl=64 time=40.0 ms
64 bytes from 44.44.44.44: icmp_seq=35 ttl=64 time=40.0 ms
^C
--- 44.44.44.44 ping statistics ---
35 packets transmitted, 35 received, 0% packet loss, time 34155ms
rtt min/avg/max/mdev = 32.018/47.683/64.036/8.200 ms
quagga-router#
```

Figura 43: Resultado de um ping entre ARGENTINA e a LAN de RÚSSIA

Ainda assim, foi mostrado que o RIP não se comportou como esperado, não se adaptando muito bem ao problema de escalabilidade. Neste caso, poderiam ser avaliados outros protocolos de roteamento, como o BGP e OSPF, por exemplo. Porém, como não foram feitos exeprimentos com eles, não é possível dizer se o comportamento deles seria similar ao do RIP nos testes executados.

4.4 Discussão sobre os resultados

Não foi possível realizar experimentos exaustivos com o RIP, mas pode-se observar que é possível incorporar este protocolo como um elemento integrador das VPNs redundantes. Os seguintes pontos podem ser salientados:

1. Como o RIP periodicamente divulga as suas tabelas nas interfaces que são configuradas, os túneis serão utilizados durante período.

2. No laboratório com o RIP não se conseguiu utilizar o *multicast* do GRE. Isso trouxe um problema de escalabilidade na configuração dos SPOKEs, pois foi necessário utilizar o mecanismo *neighbor*, especificando os sites do cliente.

3. Para poder dar prioridade para a VPN principal, teve que ser forçada uma métrica maior para o acesso da DMVPN.

4. O RIP possui um mecanismo de trigger update que permite contornar o envio periódico das tabelas quando uma das interfaces se torna inoperante. Por outro lado, se cair um enlace, a identificação da queda é demorada devido aos temporizadores. Foi realizada uma mudança nos temporizadores a fim de essas quedas terem uma identificação mais ágil pelo RIP.

4.5 Conclusão

Embora não se tenha testado exaustivamente o protocolo RIP, neste capítulo foi apresentado um estudo e experimento preliminar sobre o uso do RIP como protocolo de roteamento em um esquema de redundância com DMVPN. Os resultados foram promissores. Deve-se salientar que existe pouca referência usando softwares opensource, como o OpenNHRP e o Quagga, o que limitou um maior aprofundamento do assunto.

5 *Conclusões*

Este trabalho visou estudar, avaliar e experimentar o uso de DMVPNs usando softwares opensource como uma alternativa de redundância à VPNs baseadas em BGP, MPLS e VRF, a fim de oferecer um serviço equivalente utiliazando softwares e ferramentas *opensource.*

Em uma primeira etapa, foram realizados estudos sobre *VPNs baseadas em BGP, MPLS e VRF* e *DMVPN* para aprofundamento do tema e embasamento teórico para a simulação prática.

Num segundo momento, foi criado um ambiente virtual simulando uma situação real com uma rede DMVPN, a fim de avaliar a complexidade de implementação, a compatibilidade dos softwares e protocolos utilizados e o seu uso em um ambiente prático, no qual não encontra-se maduro o suficiente para implementação em uma situação real.

Com os testes realizados, a proposta inicial de avaliar a complexidade e escalabilidade da implementação de uma DMVPN usando uma implementação *opensource* foi alcançada. Porém o objetivo de avaliar a capacidade da DMVPN atuar como redundância à uma VPN baseada e VRF nao foi alcançada devido ao tempo necessário para implementação e falta de equipamentos para testes. Contudo, foi feito um estudo avaliando este cenário.

Um útimo experimento foi feito utilizando o protocolo de roteamento RIP para divulgação das redes da DMVPN. Para trabalhos futuros, pode-se avaliar o problema de divulgação de rotas, realizar a avaliação do uso de outros protocolos de roteamento em uma DMVPN, bem como a integração prática de uma DMVPN com VPNs baseadas em BGP, MPLS e VRF.

APÊNDICE A – Arquivos de configuração após último experimento

A.1 HUB

A.1.1 OpenNHRP

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/opennhrp/opennhrp.conf
interface gre1
      cisco-authentication 1234
      holding-time 5
root@emerson-VirtualBox:/home/emerson#
```

Figura 44: opennhrp.conf no HUB

A.1.2 Quagga

A.1.2.1 Zebra

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/quagga/zebra.conf
!
! Zebra configuration saved from vty
!   2014/07/08 22:41:56
!
hostname Router
password zebra
enable password zebra
!
interface eth0
 ipv6 nd suppress-ra
!
interface eth1
 ip address 200.101.5.146/29
 ipv6 nd suppress-ra
!
interface gre0
 ipv6 nd suppress-ra
!
interface gre1
 ipv6 nd suppress-ra
!
interface lo
 ip address 77.77.77.77/32
!
ip route 0.0.0.0/0 200.101.5.145
!
ip forwarding
!
!
line vty
!
root@emerson-VirtualBox:/home/emerson#
```

Figura 45: zebra.conf no HUB

A.1.2.2 RIPD

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/quagga/ripd.conf
!
! Zebra configuration saved from vty
!   2014/07/08 22:41:56
!
hostname ripd
password zebra
log stdout
!
router rip
 version 2
 timers basic 5 8 10
 network 77.77.77.0/24
 neighbor 10.0.0.2
 neighbor 10.0.0.3
!
line vty
!
root@emerson-VirtualBox:/home/emerson#
```

Figura 46: ripd.conf no HUB

A.1.3 Ativar interface túnel no Linux

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/init.d/gre1_up
#!/bin/bash
ip tunnel add gre1 mode gre key 1234 ttl 255
ip addr add 10.0.0.1/24 dev gre1
ip tunnel change gre1 local 200.101.5.146
ip link set gre1 up
opennhrp -d
root@emerson-VirtualBox:/home/emerson#
```

Figura 47: gre1_up no HUB

A.2 SPOKE2

A.2.1 OpenNHRP

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/opennhrp/opennhrp.conf
interface gre1
      map 10.0.0.1/24 200.101.5.146 register cisco
      cisco-authentication 1234
      holding-time 5
interface lo
  shortcut-destination
root@emerson-VirtualBox:/home/emerson#
```

Figura 48: opennhrp.conf no SPOKE2

A.2.2 Quagga

A.2.2.1 Zebra

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/quagga/zebra.conf
!
! Zebra configuration saved from vty
!   2014/07/08 22:41:53
!
hostname SPOKE2
!
interface eth0
 ipv6 nd suppress-ra
!
interface eth1
 ipv6 nd suppress-ra
!
interface eth2
 ip address 187.55.4.10/29
 ipv6 nd suppress-ra
!
interface gre0
 ipv6 nd suppress-ra
!
interface gre1
 ipv6 nd suppress-ra
!
interface gretap0
 ipv6 nd suppress-ra
!
interface lo
 ip address 44.44.44.44/32
!
ip route 0.0.0.0/0 187.55.4.11
!
ip forwarding
!
!
line vty
!
root@emerson-VirtualBox:/home/emerson#
```

Figura 49: zebra.conf no SPOKE2

A.2.2.2 RIPD

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/quagga/ripd.conf
!
! Zebra configuration saved from vty
!   2014/07/08 22:41:53
!
hostname SPOKE2
log stdout
!
debug rip events
!
router rip
 version 2
 timers basic 5 8 10
 network 44.44.44.0/24
 neighbor 10.0.0.1
 neighbor 10.0.0.2
!
line vty
!
root@emerson-VirtualBox:/home/emerson#
```

Figura 50: ripd.conf no SPOKE2

A.2.3 Ativar interface túnel no Linux

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/init.d/gre1_up
#!/bin/bash
ip tunnel add gre1 mode gre key 1234 ttl 255
ip addr add 10.0.0.3/24 dev gre1
ip tunnel change gre1 local 187.55.4.10
ip link set gre1 up

root@emerson-VirtualBox:/home/emerson#
```

Figura 51: gre1_up no SPOKE2

A.3 Configurações comuns à todos

A.3.1 IPSecTools

A.3.1.1 PSK

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/racoon/psk.txt
*       1234
```

Figura 52: psk.txt em todos os *sites*.

A.3.1.2 Racoon

```
root@emerson-VirtualBox: /home/emerson
root@emerson-VirtualBox:/home/emerson# cat /etc/racoon/racoon.conf
log notify;
path pre_shared_key "/etc/racoon/psk.txt";
   remote anonymous {
      exchange_mode main,aggressive;
      lifetime time 24 hour;
      # nat_traversal on;
      script "/etc/opennhrp/racoon-ph1down.sh" phase1_down;
      proposal {
         encryption_algorithm 3des;
         hash_algorithm sha1;
         authentication_method pre_shared_key;
         dh_group 5;
      }
   }
   sainfo anonymous {
      lifetime time 12 hour;
      encryption_algorithm 3des, blowfish 448, rijndael;
      authentication_algorithm hmac_sha1, hmac_md5;
      compression_algorithm deflate;
   }
root@emerson-VirtualBox:/home/emerson#
```

Figura 53: racoon.conf em todos os *sites*.

Referências

CISCO. *BGP - Border Gateway Protocol (Parte 1)*. 2008. Disponível em: <http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a00800c95bb.shtml/>.

FARINACCI, D. et al. *Generic Routing Encapsulation (GRE)*. IETF, mar. 2000. RFC 2784 (Proposed Standard). (Request for Comments, 2784). Updated by RFC 2890. Disponível em: <http://www.ietf.org/rfc/rfc2784.txt>.

FERGUSON, P.; HUSTON, G. *What is a VPN? - Part I and II*. 2007. Disponível em: <www.cisco.com>.

FILIPETTI, M. *Cisco Dynamic Multipoint VPN (DMVPN)*. 2008. Disponível em: <blog.ccna.com.br/2008/10/28/dmvpn/>.

FOX, B.; PETRI, B. *NHRP Support for Virtual Private Networks*. IETF, dez. 1999. RFC 2735 (Proposed Standard). (Request for Comments, 2735). Disponível em: <http://www.ietf.org/rfc/rfc2735.txt>.

LUCIANI, J. et al. *NBMA Next Hop Resolution Protocol (NHRP)*. IETF, abr. 1998. RFC 2332 (Proposed Standard). (Request for Comments, 2332). Disponível em: <http://www.ietf.org/rfc/rfc2332.txt>.

MALKIN, G. *STD 56: RIP Version 2*. nov. 1998. See also RFC2453 (**??**). Obsoletes RFC1723 (**??**).

PATTERSON, M. *What is VRF: Virtual Routing and Forwarding*. 2009. Disponível em: <http://www.plixer.com/blog/netflow/what-is-vrf-virtual-routing-and-forwarding/>.

PINHEIRO, J. *Conceitos de Redundância e Contingência*. 2004. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_conceitos_de_redundancia.php>.

PREUSS, P. *DMVPN with Linux*. 2009. Disponível em: <patrickpreuss.wordpress.com/2009/02/14/dmvpn-with-linux/>.

REKHTER, Y.; LI, T.; HARES, S. *A Border Gateway Protocol 4 (BGP-4)*. IETF, jan. 2006. RFC 4271 (Draft Standard). (Request for Comments, 4271). Updated by RFCs 6286, 6608, 6793. Disponível em: <http://www.ietf.org/rfc/rfc4271.txt>.

ROSEN, E.; REKHTER, Y. *BGP/MPLS IP Virtual Private Networks (VPNs)*. IETF, fev. 2006. RFC 4364 (Proposed Standard). (Request for Comments, 4364). Updated by RFCs 4577, 4684, 5462. Disponível em: <http://www.ietf.org/rfc/rfc4364.txt>.

ROSEN, E.; VISWANATHAN, A.; CALLON, R. *Multiprotocol Label Switching Architecture*. IETF, jan. 2001. RFC 3031 (Proposed Standard). (Request for Comments, 3031). Updated by RFCs 6178, 6790. Disponível em: <http://www.ietf.org/rfc/rfc3031.txt>.

SAXENA, A. *DMVPN*. 2013. Disponível em: <https://supportforums.cisco.com/docs/DOC-1227/>.

SOUZA, S. *BGP - Border Gateway Protocol (Parte 1)*. 2011. Disponível em: <http://abrint.com.br/artigos/item/40-bgp-border-gateway-protocol-parte-1.html/>.

TAFT, B. P. *Conceitos de uma rede MPLS*. 2004. Disponível em: <http://www.gta.ufrj.br/grad/04_2/MPLS/>.

TYSON, J. *"How Stuff Works - Como funciona uma VPN"*. 2007. Disponível em: <informatica.hsl.uol.com.br/vpn.htm>.

UFRJ. *DMVPN*. 2012. Disponível em: <http://www.gta.ufrj.br/grad/04l_2/MPLS/funcionamento.htm/>.

WESTPHAL, R. *Estudo e Implementação de MPLS/BGP IPv6 VPNs em GNU/Linux*. 2011. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/37170/000819636.pdf?sequence=1/>.