

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
ESCOLA DE INFORMÁTICA APLICADA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

UM ESTUDO DE VPNs LAYER 3 MPLS-BASED UTILIZANDO
MULTIPROTOCOL BGP

Carlos Alberto de Melo Velez Filho

Thiago Sardinha Moreira

Orientador: Sidney Cunha de Lucena

Dezembro/2016

Projeto de Graduação apresentado à Escola de
Informática Aplicada da Universidade Federal do
Estado do Rio de Janeiro (UNIRIO) para obtenção
do título de Bacharel em Sistemas de Informação.

Carlos Alberto de Melo Velez Filho

Thiago Sardinha Moreira

UM ESTUDO DE VPNs LAYER 3 *MPLS-BASED* UTILIZANDO MULTIPROTOCOL BGP

Trabalho de conclusão de curso de graduação apresentado à Universidade Federal do Estado do Rio de Janeiro como requisito parcial para a obtenção do título de Bacharel(a) em Sistemas de Informação.

Aprovado em: _____ de _____ de _____.

BANCA EXAMINADORA

Sidney Cunha de Lucena - UNIRIO

Carlos Alberto Vieira Campos - UNIRIO

Morganna Carmem Diniz - UNIRIO

Agradecimentos de Carlos Alberto de Melo Velez Filho:

Dedico esta conquista primeiramente à meus amados avós, Luzia Camilo da Motta e Marcos José da Motta, que com seus inquestionáveis esforços, honestidade, dedicação e carinho promoveram a criação da melhor família que eu poderia pedir ao cosmos - Em especial para minha querida avó, que já não mais habita este plano, mas que com toda sua bondade, paciência e verdade com certeza estaria muito orgulhosa em ver a materialização deste dia.

Dedico também aos meus pais, Carlos Alberto de Melo Velez e Jurema da Motta Velez, por promoverem em mim uma personalidade forte e um espírito questionador, por terem criado a mim e ao meu irmão como pessoas de bem que somos, por me apoiarem nas escolhas pessoais e profissionais que fiz até aqui, e sobretudo por serem os melhores amigos que poderíamos ter.

Agradeço ao meu irmão, Pedro Paulo da Motta Velez, cujo laço de verdadeira irmandade, carinho e preocupação tão necessários no dia a dia, me deram também muita energia para continuar esta empreitada.

Um enorme agradecimento também à minha tia, Marcia Camilo da Motta Santos, pelo ombro amigo, pelo sentimento verdadeiro de felicidade dividido a cada uma de minhas conquistas, e pela preocupação constante com meu bem estar.

Gostaria também de agradecer à todos os espíritos de luz que comigo caminham neste plano, pelas evoluções que juntos conquistamos e pelos incontáveis ensinamentos, palavras de conforto e intenções trocadas até hoje.

Agradeço ao meu companheiro neste estudo, Thiago, pela amizade sincera ao longo de todos estes anos de universidade. e por ter embarcado comigo neste projeto final nada trivial ou confortável, mas tão engrandecedor.

Agradeço aos meus professores da UNIRIO, que marcaram uma época inesquecível da minha vida de maneiras muito diferentes, com qualidade de ensino de ótimo nível.

Agradecimentos especiais também aos amigos que estiveram comigo de maneira tão presente em alguns momentos nesta jornada - Sergio Feitosa, Raphael Gomes, Anna Gabi, João Felipe, Mel Cassiano, Michelle Cassiano, Thales Lopes, Tatiana Dias, Jessica Alves, Enrique Frade, Alysson Gomes - Por fazerem o cotidiano mais leve e me darem força, cada um no seu tempo e à sua maneira.

Por último mas nem um pouco menos importante, agradeço à Deus pela oportunidade de conviver com todas estas pessoas incríveis, e de poder materializar com este trabalho o fim de um ciclo de muitas dificuldades, mas também de enormes alegrias, aprendizagem e crescimento.

Agradecimentos de Thiago Sardinha Moreira

Gostaria de agradecer, em primeiro lugar, meus pais Maria da Glória Sardinha, Antonio Cláudio Rocha Moreira, Anna Cristina Pereira Couto e Roberto Wagner dos Santos Robillard pelo apoio incondicional durante toda a vida e, principalmente, para conseguir concluir esse capítulo tão importante em minha vida.

Quero agradecer também às pessoas mais próximas. Meu irmão Roberto, meus amigos Fabiene, Bruno, Victor, Leonardo, Leandro, Marcos, Felipe, Lohan, Nicolle, Bruno, Pedro, Marina, Nilso e tantos outros.

O apoio que tive no trabalho de meus gerentes Wesley e Luiz, que me liberaram num período crucial do ano e de todos os companheiros de equipe que puderam me ajudar.

Agradeço aos professores da UNIRIO pelo aprendizado e por me tornar o profissional que sou hoje.

Por fim, quero agradecer ao meu parceiro nesse trabalho, Carlos, pela oportunidade de trabalharmos em conjunto nesse projeto e, principalmente, pela amizade construída e pelos momentos dentro e fora da UNIRIO nos últimos 7 anos.

Resumo

Este estudo teve como objetivo definir os diversos conceitos necessários para o entendimento e implementação de VPNs de camada 3, utilizando MPLS e Multiprotocol BGP.

Por meio de um embasamento teórico sequencial, foram estudadas diferentes tecnologias aplicadas ao tema, com a realização de um estudo de caso com sistemas operacionais CISCO reais em um ambiente emulado ao final do trabalho.

Concluíram-se, neste experimento, as vantagens e aplicabilidade da utilização de MPLS em ambientes corporativos, a segurança provida pela manipulação de VRFs com Multiprotocol BGP, assim como a eficiência provida por uma arquitetura altamente escalável e não orientada à protocolos específicos.

Palavras-chave: MPLS, VPN, Multiprotocol BGP, GNS, VRF

Abstract

The objective of this study is to define the various concepts necessary for the understanding and implementation of layer 3 VPNs, using MPLS and Multiprotocol BGP.

Through a sequential theoretical basis, different technologies applied to the subject were studied, supported by real CISCO operating systems in an emulated environment at the end of work.

Therefore, the experiment concludes the advantages and applicability of MPLS in corporate environments, the security provided by the manipulation of VRFs with Multiprotocol BGP, as well as the efficiency provided by a highly scalable and non-protocol-oriented architecture.

Keywords: MPLS, VPN, Multiprotocol BGP, GNS, VRF

Lista de Ilustrações

Figura 2-1.	Cabeçalho da mensagem BGP <i>open</i>	8
Figura 2-2.	Modelo básico de uma VPN	11
Figura 2-3.	Funcionamento de uma MPLS VPN	14
Figura 2-4.	Topologia MPLS	18
Figura 2-5.	Cabeçalho MPLS	19
Figura 3-1.	A tela Inicial do GNS3	23
Figura 3-2.	Diferentes métodos de criação de uma instância de roteador no GNS3	25
Figura 3-3.	<i>Slots</i> e adaptadores para inclusão em uma instância de roteador emulado	26
Figura 3-4.	Topologia do experimento	28
Figura 3-5.	Backbone MPLS	29
Figura 3-6.	Roteador R5	30
Figura 3-7.	Roteador R7	34
Figura 3-8.	Topologia - MPLS CORE + Cliente A	36
Figura 3-9.	Elementos envolvidos nas configurações para o Cliente A	36
Figura 3-10.	Elementos envolvidos nas configurações para o Cliente A	37
Figura 3-11.	Configurações R5 e R1	39
Figura 3-12.	Topologia - MPLS CORE + Cliente B	44
Figura 3-13.	Elementos envolvidos nas configurações para o Cliente B	44
Figura 3-14.	Elementos envolvidos nas configurações para o Cliente B	45
Figura 3-15.	Configurações R5 e R2	48
Figura 4-1.	Core MPLS	52
Figura 4-2.	Core MPLS e PEs do Cliente A	60
Figura 4-3.	Core MPLS e PEs do Cliente B	69
Figura 4-4.	Falha de links no experimento	84

Lista de Tabelas

Tabela 1-1.	Distâncias administrativas	5
Tabela 3-1.	Plano de endereçamento do experimento	27

Lista de Siglas e Abreviaturas

VPN – *Virtual Private Network*

MPLS – *Multiprotocol Label Switching*

BGP – *Border Gateway Protocol*

MBGP – *Multiprotocol Extensions for BGP*

IGP – *Interior Gateway Protocol*

EGP – *Exterior Gateway Protocol*

OSPF – *Open Shortest Path First*

VRF – *Virtual Routing and Forwarding*

CEF – *Cisco Express Forwarding*

GNS – *Graphical Network Simulator*

AS – *Autonomous System*

RD – *Route Distinguisher*

RT – *Route Target*

VC – *Virtual Circuit*

CE – *Customer Edge*

PE – *Provider Edge*

LSP – *Label Switched Paths*

LDP – *Label Distribution Protocol*

LER – *Label Edge Router*

SSH – *Secure Shell*

L2TP – *Layer 2 Tunnelling Protocol*

RBI – *Routing Information Base*

TCP – *Transmission Control Protocol*

UDP – *User Datagram Protocol*

ATM – *Asynchronous Transfer Mode*

SSL/TLS – *Secure Sockets Layer / Transport Layer Security*

IP – *Internet Protocol*

FEC – *Forwarding Equivalence Class*

Sumário

1 Introdução	0
1.1 Apresentação	0
1.2 Objetivo do trabalho e metodologia	1
1.3 Estrutura do texto	1
2 Roteamento IP e VPNs	3
2.1 Control Plane x User Plane	3
2.2 Roteamento IP	3
2.2.1 Estático x Dinâmico	4
2.2.2 Intradomínio x Interdomínio	6
2.3 OSPF	7
2.4 BGP	8
2.5 Multiprotocol BGP	9
2.6 VPN	10
2.6.1 Conceito	10
2.6.2 Formas de Implementação	11
2.6.3 VPN com MPLS	13
2.7 VRF	17
2.8 MPLS	17
2.8.1 CEF Cisco	20
3 Experimento	22
3.1 GNS e Dynamips	22
3.2 Detalhes do experimento	24
3.2.1 Propósito do Experimento	24
3.3 Implementação do experimento	28
3.3.1 Configuração do MPLS no CORE backbone	28

3.3.2 Configuração do roteamento BGP CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-A.....	35
3.3.3 Configuração do roteamento OSPF CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-A	43
4 Resultados e análises	52
4.1 Resultados do CORE MPLS no Backbone - Core OSPF e MPLS LDP, com configuração de todos os roteadores da nuvem MPLS.....	52
4.2 Resultados de roteamento BGP CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-A	59
4.3 Resultados de roteamento OSPF CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-B	68
4.4 Testes adicionais.....	77
4.4.1 Testes de separação de tabelas de VRFs	78
4.4.2 Testes de peers Multiprotocol BGP	79
4.4.3 Testes de falhas de links	83
5 Conclusão e trabalhos futuros	90
Referências Bibliográficas	91
Anexos.....	93

1 Introdução

1.1 Apresentação

Em um mundo globalizado como o dos dias atuais, a integração entre serviços, usuários e diferentes plataformas tem se feito cada vez mais necessária. O acesso e interconexão entre redes geograficamente distantes tem sido amplamente estudado desde o surgimento da internet, em busca de saídas para as exigências de um mercado que consome cada vez mais dados e fica cada vez mais interativo.

Este crescente volume de tráfego e informações, aliado à grande demanda por qualidade de serviços que eles proporcionam, gerou a necessidade de pensar em tecnologias que apoiassem meios mais seguros, flexíveis e ágeis na transmissão de dados, sejam eles de âmbito pessoal ou corporativo.

As primeiras soluções implementadas envolviam ligações físicas de forma praticamente direta entre as partes, exigindo infraestrutura cada vez mais custosa a cada novo integrante da conexão.

Em resposta à esta necessidade surgiram as chamadas *Virtual Private Networks* (VPN), que nada mais são do que redes artificiais funcionando sobre *backbones* privados e/ou a Internet. Este serviço surge como alternativa à interconexões dedicadas entre diferentes redes, uma vez que utiliza a rede pública para estender o domínio entre duas ou mais redes locais (LAN).

Quando usadas com MPLS, VPNs permitem que diversas redes se interconectem transparentemente através da rede intermediária do seu Service Provider. A rede de um Service Provider pode suportar diversas VPNs IP diferentes entre si. Cada uma delas é vista como uma rede privada para seus usuários, separadas de todas as outras redes. Em uma VPN, cada rede pode enviar pacotes IP para qualquer outra rede nesta mesma VPN.
(CISCO SYSTEMS, Document ID: 13733 – Configuring a MPLS VPN, Página 1)

VPN, porém, é apenas o conceito. Existem diversas formas de implementação, cada uma com pontos positivos e negativos que devem ser levados em conta pensando na estrutura da rede, que tipo de dado será trafegado e como esse acesso deverá ser feito.

Dentre as diversas soluções diferentes para a criação de VPNs, estudaremos nessa monografia o serviço de VPNs baseadas em MPLS, um mecanismo de encapsulamento que vem ganhando notoriedade no mercado por conta de sua alta escalabilidade.

O MPLS é uma tecnologia que visa aumentar a velocidade de transmissão de pacotes em uma rede sem levar em consideração o protocolo de rede que o pacote a ser transferido está utilizando, o que o torna altamente escalável. Ele também aumenta a velocidade desta transmissão com a manipulação de labels entre camadas de enlace e rede, e satisfaz conceitos de segurança com seus mecanismos de tunelamento. VPNs que se utilizam de MPLS fazem uso destes benefícios para criar arquiteturas capazes de suportar uma grande pluralidade de clientes e serviços, e conseguem, com isso, fazer uma melhor alocação de seus recursos por meio de estudos de engenharias de tráfego.

1.2 Objetivo do trabalho e metodologia

Neste trabalho de conclusão de curso, serão estudadas as tecnologias envolvidas na criação das chamadas VPNs de camada 3. O estudo tem por objetivo construir um cenário experimental em um emulador de redes para testar a configuração de VPNs de camada 3 utilizando MPLS e Multiprotocol BGP.

Para isso, será realizado um embasamento teórico sequencial dos protocolos que são pré-requisitos para a criação destas VPNs em cenários reais, discutindo suas funcionalidades e os benefícios e desvantagens trazidos com eles.

Ao final do estudo, após este embasamento conceitual, será demonstrada a viabilidade dos conceitos estudados, por meio da implementação passo a passo do ambiente emulado mencionado, com diferentes cenários de testes utilizando sistemas operacionais reais CISCO. Esta rede simulará um provedor de serviços vendendo um serviço de interconexão para dois diferentes clientes usando uma mesma nuvem MPLS.

1.3 Estrutura do texto

Este trabalho está dividido em cinco partes.

A primeira parte é introdutória, e apresenta de forma breve os objetivos e a organização do estudo.

A segunda parte, ou capítulo 2, fala sobre conceitos de roteamento, indicando diferenças sobre planos de controle e de dados e meios de implementação de rotas, além de apresentar protocolos de roteamento dinâmico e como os mesmos podem

influenciar no conceito de VPNs. Este capítulo visa conceituar todos os tópicos abordados no experimento que será visto no capítulo 3, incluindo o conceito de VPNs, demonstrando sua utilidade e expondo conceitos de tunelamento, encriptação e encaminhamento de dados nas mesmas, MPLS, alvo do trabalho como um todo na elaboração das VPNs MPLS, abordando conceitos de sua arquitetura, funcionalidades e mecanismos proporcionados por ele.

O terceiro capítulo é aquele que descreve e apresenta o experimento, suas nuances e configurações de todos os protocolos e conceitos estudados até ele. Para isso, ele apresenta o *software* usado no estudo para emulação do ambiente de testes, sua configuração e as funcionalidades que o mesmo proporciona para criação do ambiente estudado, além de um *overview* do experimento em si.

A parte seguinte, ou capítulo 4, é aquele que apresenta considerações sobre o ambiente emulado, os benefícios percebidos com ele e constatações gerais sobre o que foi criado, visando a experimentação da topologia do capítulo 3 por meio de testes. Nele, é descrito todo o processo de configuração realizado, sequencialmente.

Em seguida, há um capítulo para conclusões e elucidação de possíveis trabalhos futuros, o capítulo 5, em que são apontados os desafios para uma futura continuação deste desenvolvimento.

As Referências Bibliográficas referem todos os locais de pesquisa necessários para a elaboração desta monografia. E finalmente, surgem os Anexos referenciados no trabalho.

2 Roteamento IP e VPNs

2.1 *Control Plane x User Plane*

O roteamento IP é implementado, operado e controlado pelo roteador. Ele é usado quando um elemento, em dada rede local, envia um pacote para um destino que está em uma rede externa. Por rede externa, entende-se qualquer rede que requer a transmissão de dados por um ou mais roteadores antes de encontrar seu destino.

Cada roteador de uma rede mantém consigo uma tabela de endereços e detalhes de roteadores ou outras redes às quais ele já esteve conectado anteriormente, chamada tabela de roteamento.

O plano de controle é a parte de uma rede que carrega tráfego de sinalização e é responsável por este roteamento, que inclui a descoberta dos caminhos possíveis em uma rede e a escolha para onde o tráfego será enviado. Ele lida com a troca de tabelas de roteamento entre roteadores, e com atualização da topologia que cria com essas tabelas. O plano de dados é o movimento efetivo dos pacotes sobre o caminho escolhido pelo plano de controle, a qual chamamos encaminhamento de pacotes.

2.2 Roteamento IP

O processo de roteamento ocorre através da descoberta de redes pelo roteador, conforme visto anteriormente. Esse processo acontece através das comunicações que possui com os roteadores vizinhos, ou através da configuração manual de um administrador de sistemas. Com as rotas aprendidas, o roteador pode criar a tabela de roteamento, utilizada como referência para o encaminhamento dos pacotes.

Quando um pacote é recebido em uma interface do roteador, ele procura pelo endereço IP de destino deste pacote em sua tabela de roteamento. Se for encontrado uma combinação com o dado IP de destino, o pacote é encaminhado para o roteador correspondente ou para uma lista de roteadores, pela qual ele deve passar até chegar no seu nó de destino.

2.2.1 Estático x Dinâmico

Existem maneiras diferentes usadas pelos roteadores para construção de suas tabelas de roteamento. A maneira pela qual ela é construída determina se é realizado roteamento estático ou dinâmico. A principal característica do roteamento estático é a necessidade de intervenção manual no processo, através de regras criadas por um administrador. O roteamento dinâmico é justamente o oposto, onde existe um protocolo de roteamento que realiza a criação da tabela de roteamento de acordo com informações de roteadores vizinhos que também utilizam o mesmo protocolo de roteamento.

No roteamento estático, as rotas não se alteram dinamicamente de acordo com alterações na topologia da rede, e precisam ser alteradas manualmente.

Por conta disto, o custo de sua manutenção está diretamente relacionado à complexidade e tamanho da rede em que ele está implementada: A cada nova rede adicionada ou alterada em sua topologia, será necessário operar manualmente informações sobre ela na tabela de roteamento. Por este alto custo operacional, o roteamento estático não é viável em grandes ambiente de produção, mas ainda é utilizado para ambientes menores.

Por não gerar uso de banda para troca de informações entre roteadores com roteamento estático implementado, há uma redução considerável do processamento de suas CPUs, se considerarmos o roteamento dinâmico. Ele é indicado para ambientes domésticos, conjuntos de redes pequenas e para topologias que não serão alteradas ao longo do tempo.

No roteamento dinâmico, são usados protocolos de roteamento para o envio e descoberta de rotas. Os protocolos de roteamento utilizados em redes pertencem a duas categorias: *Interior Gateway Protocol* (IGP) e *Exterior Gateway Protocol* (EGP). Protocolos IGP são usados para troca de informações entre roteadores pertencentes a um mesmo Sistema Autônomo (*Autonomous System* - AS).

Um AS é uma rede ou grupo de redes gerenciadas pelo mesmo administrador de rede controlada por uma única entidade administrativa e que utiliza uma política de roteamento única e bem definidas, possuindo um número único global que o define, chamado Autonomous System Number, ou ASN [Cisco Systems, 2016].

Já protocolos EGP são utilizados para comunicação entre roteadores pertencentes a Sistemas Autônomos diferentes.

Distâncias Administrativas são métricas utilizadas para classificar a confiabilidade das informações recebidas por um roteador através de seus vizinhos. A Distância Administrativa é um número inteiro entre zero e 255, sendo que quanto menor mais confiável é a rota^[1]. Ou seja, zero é a rota mais confiável e 255 indica uma rota inalcançável. A tabela 1-1 demonstra os valores e os protocolos relacionados.

Tabela 1-1. Distâncias administrativas

Protocolo	Distância Administrativa
Diretamente conectado	0
Rota estática	1
EIGRP Rota sumarizada	5
External BGP	20
Internal EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
ODR	160
External EIGRP	170
Internal BGP	200
Desconhecida	255

Fonte: Site da CISCO [1]

As VPNs estudadas neste trabalho terão seus domínios de roteamento estendidos via *backbone* comum, fazendo uso de roteamento estático e dinâmico. A seguir, serão definidos os diferentes tipos de domínios existentes em uma rede, e a relação entre eles.

2.2.2 Intradomínio x Interdomínio

Domínios administrativos são coleções de sistemas intermediários e sub-redes operadas por uma mesma autoridade administrativa ou organização. Esta definição leva em consideração que os nós das topologias administradas pela organização são confiáveis entre si.

Domínios de roteamento são coleções de sistemas finais e intermediários que operam de acordo com os mesmos procedimentos de roteamento, e que estão sob um mesmo domínio administrativo. Basicamente usam a mesma métrica de roteamento, o mesmo protocolo de distribuição, e o mesmo algoritmo de seleção de caminhos, por definição.

Estes domínios estão hierarquicamente relacionados, visto que um domínio administrativo pode possuir diversos domínios de roteamento. O contrário, no entanto, não se aplica.

Um roteamento de domínio intra-administrativo está relacionado à interconexão de múltiplos domínios de roteamento dentro de uma mesma administração, de forma equivalente à um IGP. Com isso, é assumido um nível moderado de confiança, já que pontos como endereçamento, manutenção de *policies* e *troubleshooting* interno são feitos pela própria entidade administradora.

A natureza das interações entre dois domínios de roteamento pode variar entre níveis mais entrelaçados de roteamento à níveis mais modeláveis, baseados em *policies*. Com o roteamento de domínios inter-administrativos, no entanto, se faz necessária a criação de um plano técnico unificado coordenado entre as organizações.

O roteamento inter-administrativo trata do controle do fluxo de informações em um nível altamente estruturado entre as organizações, que pode requerer acordos multilaterais. Os problemas enfrentados neste modelo envolvem conceitos - legais, de segurança, controle de acesso - que devem ser mapeados antes de qualquer implementação.

O conceito de Sistema Autônomo está fortemente ligado à domínios de roteamento, uma vez que um AS pode contar múltiplos domínios, e por definição, representa um IGP e um ponto de administração de *policies*. Sendo assim, seu parâmetro *Autonomous System Number* (ASN) pode também ser visto como um número de domínio administrativo.

2.3 OSPF

Open Shortest Path First (OSPF) é um protocolo de roteamento intra-domínio que se utiliza de vetores de estados de link, ao invés dos vetores de distância de seus antecessores. Isso significa dizer que o OSPF propaga avisos que servem como atualizações dos seus estados de link – os chamados *Link-State Advertisements* (LSA) – no lugar de versões atualizadas de sua tabela de roteamento. Como somente os LSAs são trocados no lugar da tabela inteira, o OSPF converge em uma velocidade que deve ser levada em consideração.

O algoritmo *Link-State* de busca pelo melhor caminho utilizado pelo OSPF se chama *Shortest Path First* (SPF). O SPF funciona de modo diferente dos algoritmos vetor-distância, que requerem que os nós de uma rede informassem mudanças de topologia somente a seus vizinhos. Este tipo de algoritmo possui menor complexidade computacional, mas não é aplicável para redes robustas, visto a limitação quanto ao número de nós utilizados e o reprocessamento de mensagens que ocorre em protocolos desse tipo, como o RIP.

Com SPF, cada roteador em uma área OSPF contém uma *database* de estados de link idêntica à todos os outros roteadores. Esta base de dados é uma lista de todas as interfaces utilizáveis de vizinhos alcançáveis nesta área. Cada rota contém o identificador de interface, o número do enlace e a distância ou métrica e, com essas informações, os roteadores descobrem a melhor rota possível.

Quando ocorre uma alteração em um dos enlaces da rede, os nós adjacentes percebem e avisam aos seus vizinhos. Para os vizinhos saberem se este aviso é novo ou velho, é necessário um campo no pacote com número da mensagem ou sua hora. Portanto, quando um nó recebe uma mensagem, primeiro é feita a verificação da existência ou não desta rota, se ela não existir é adicionada. Se existe, compara-se o número da mensagem recebida com a rota da tabela. Se o número da mensagem recebida for maior que a da tabela, a rota é substituída, caso contrário, a rota da tabela é transmitida como uma nova mensagem. Se os números forem iguais nada é feito. Este processo é chamado de *flooding*.

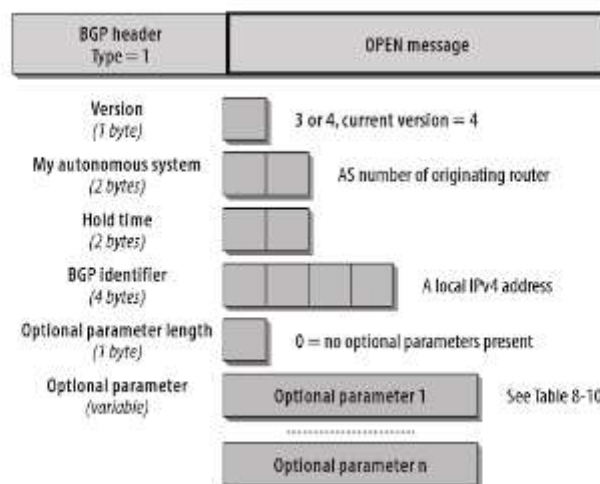
2.4 BGP

Border Gateway Protocol (BGP) é um protocolo de roteamento que gerencia a transferência de pacotes em sistemas autônomos. Ele decide rotas para transferência de pacotes baseado em regras, caminhos e políticas de rede que devem ser configuradas por um administrador da rede.

BGP é um protocolo de vetor de caminho. Cada roteador mantém uma tabela de roteamento com registros indicando a rede de destino, o próximo roteador e o caminho para chegar até o destino. Além desta tabela, o protocolo também usa uma base de informação de roteamento (*Routing Information Base* - RBI), que contém informação das redes externas diretamente conectadas, assim como dos outros roteadores internos.

O roteamento que ocorre apenas dentro de um AS é chamado de BGP interno (*Internal BGP* - iBGP). Quando é utilizado para conectar um AS a outros sistemas autônomos, é chamado de BGP externo (*External BGP* - eBGP). Para o funcionamento correto do protocolo, a conexão entre os nós deve ser configurada manualmente com as conexões entre si definidas em cada ponta, pois o BGP não possui descoberta automática. A figura 2-1 demonstra como o cabeçalho BGP entrega essas informações.

Figura 2-1. Cabeçalho da mensagem BGP open



Fonte: (HAGEN, 2006, p. 206)

O protocolo BGP utiliza mensagens para estabelecer conexões, trocar informações de rota, notificar a ocorrência de algum erro e checar a se outro dispositivo BGP ainda está disponível.

Existem quatro tipos de mensagem BGP. Todas elas têm o mesmo cabeçalho, de tamanho fixo. O cabeçalho possui indicadores de sincronização e autenticação, um campo que indica o tamanho do pacote e o tipo de mensagem que está sendo enviada.

A mensagem *open* é trocada entre dois roteadores BGP logo após uma conexão TCP é estabelecida entre eles para tentar estabelecer uma conexão BGP. As mensagens *update* são enviadas para trocar informações de rotas, indicando quais devem ser retiradas e as propriedades das rotas ainda vigentes. Mensagens do tipo *keepalive* são enviadas para determinar se uma conexão ou roteador falhou ou não está mais disponível. Essas mensagens são enviadas em intervalos de tempo suficientes para que o *hold time* da conexão não expire. Por fim, quando um roteador BGP identifica que ocorreu uma condição de erro, ele manda uma mensagem do tipo *notification*. Logo em seguida, a sessão BGP e a conexão TCP entre os roteadores é fechada.

2.5 Multiprotocol BGP

A limitação do uso IPv4 *unicast* do protocolo BGP gerou uma extensão, o Multiprotocol BGP (*Multiprotocol Extensions for BGP* - MBGP). O MBGP possibilita que diferentes tipos de endereçamentos sejam utilizados em paralelo. Ele suporta tanto IPv4 quanto IPv6, tanto *unicast* quanto *multicast* [HAGEN, 2006].

Porém, para o funcionamento do MPLS L3, não seria possível com esses protocolos identificar unicamente todos os dispositivos externos a menos que todos tivessem um endereço público único. Para solucionar essa possível ambiguidade, o MBGP fornece a família de endereços VPN-IPv4. Um endereço VPN-IPv4 consiste de um valor que identifica a VPN, chamado de *route distinguisher* (RD), que serve de prefixo para o endereço IPv4, providenciando uma forma de identificá-lo unicamente [PEPELNJAK, 2002].

Somente os roteadores da borda do provedor de serviços em questão - que à frente serão apresentados como *provider edge* (PE) - precisam dar suporte à extensão VPN-IPv4. Quando este recebe uma rota IPv4 de um dispositivo dentro de uma VPN, ele encapsula o endereço num VPN-IPv4, adicionando o *route distinguisher* no prefixo da rota. Na volta, ele converte novamente, removendo o identificador da VPN e anunciando a rota para seus roteadores CE conectados.

Este encapsulamento de endereços extras funcionam adicionando cabeçalhos nas mensagens que só serão lidos novamente por roteadores também aptos e configurados para tal, e transmitidos sob um meio que não terá visibilidade sobre estes endereçamento.

O RD é transmitido em conjunto com a rota pelo MBGP quando rotas são atualizadas entre PEs.

Por exemplo, se temos duas rotas em duas VRFs diferentes que dentro de uma rede 192.168.0.0/24, é necessário um RD para saber sobre qual rota pertence à qual VRF.

Route Target (RT), diferente do RD, é utilizado para compartilhar rotas entre VRFs. Adicionando um RD à rota nós criamos um endereço global único na tabela de VPN-IPv4 do MBGP, mas para um PE dentro do *backbone* quais dessas rotas pertencem à qual VRF, um PE que sabe essa resposta deve exportar o RT para ele.

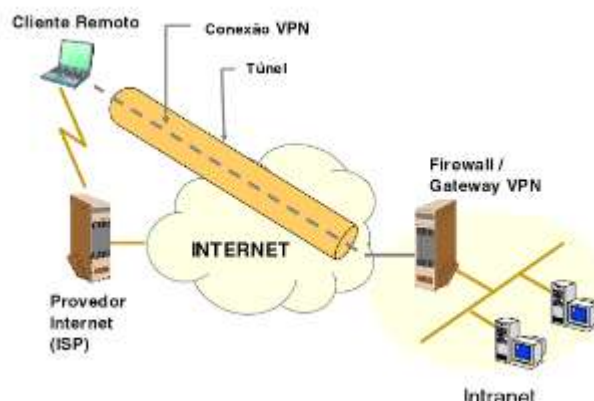
2.6 VPN

2.6.1 Conceito

Virtual Private Network (VPN), demonstrada na figura 2-2, é uma tecnologia que permite que uma rede privada se estenda por uma rede pública, normalmente a Internet. Ela possibilita que seus usuários troquem informações por redes públicas como se estivessem diretamente conectados à uma rede privada [GUIMARÃES, 2006].

VPNs são comumente associadas a soluções empresariais com o intuito de prover acesso seguro e eficiente entre filiais, escritórios ou até mesmo prover acesso remoto de seus funcionários como se estivessem na mesma rede dos recursos da empresa, como *data storages*, *data centers*, aplicações, serviços e arquivos.

Figura 2-2. Modelo básico de uma VPN



Fonte: (GUIMARÃES et al, 2006, p. 81)

Com a crescente preocupação com segurança da informação, alguns serviços que tradicionalmente levam em conta apenas qualidade e velocidade passaram a utilizar VPNs. É o caso de serviços VOIP e transferência de arquivos. Mesmo com o *overhead* causado pelo encapsulamento desses dados em pacotes TCP e técnicas de encriptação, a influência negativa na perda de pacotes, latência e *jitter* é desprezível.

Para prover uma conexão segura e privada, VPNs utilizam protocolos de tunelamento e encriptação para garantir apenas acessos autenticados. Estes protocolos permitem que qualquer tipo de dado trafegado possa ser encapsulado em pacotes IP e transferido através de uma rede que não suporte algum protocolo em particular, como por exemplo, transmitir pacotes IPv6 numa rede que suporta apenas IPv4.

Técnicas de encriptação garantem que somente pessoas autorizadas consigam compreender o conteúdo da mensagem, mesmo que ela seja interceptada, proporcionando privacidade da informação trafegada.

2.6.2 Formas de Implementação

Inicialmente, muitas redes realizavam conectividade remota do mesmo “estilo” das VPNs usando *leased lines* com *Frame Relay* ou ATM, provisionadas por uma rede normalmente operada por uma *carrier* contratada para tal. Esse tipo de conexão vem sendo substituída por VPNs baseadas em IP e IP/MPLS, devido à alta redução de

custos e consequente incremento de banda para o tráfego de dados de *user plane*, de fato. Como exemplos de métodos diferentes para implementação de VPNs que não aqueles com MPLS, citamos:

2.6.2.1 SSH

Secure Shell (SSH) é um protocolo que permite acesso a um equipamento remoto de uma maneira segura, de maneira autenticada e comunicação entre as partes criptografada através de uma rede não segura. SSH é amplamente utilizado por administradores de rede para gerenciamento de aplicações e sistemas de forma remota, permitindo que possam se logar em outros equipamento, transferir arquivos e executar comandos.

Um uso menos difundido do SSH é a capacidade de ser utilizado como protocolo de tunelamento, provendo tráfego de informações não encriptadas por meio de um canal seguro, porém com certas limitações. O protocolo permite configurar um cliente SSH para atuar como *proxy*. Configurando uma aplicação para utilizar esse *proxy*, todo o tráfego que entra nele é direcionado por uma conexão SSH, o que é conhecido como tunelamento SSH. Porém, diferente de VPNs, cada aplicação deve ser configurada separadamente para utilizar o *proxy* criado. Não é assegurado que toda informação trafegada será enviada pelo túnel SSH.

2.6.2.2 SSL/TLS

VPNs com base em SSL/TLS oferecem uma maneira mais simples de utilizar a tecnologia, sem precisar de um programa cliente VPN de terceiros, qualquer navegador amplamente utilizado pode ser configurado para realizar acesso remoto, aplicações e redes internas. O tráfego entre o navegador e a VPN é encriptado utilizando o protocolo SSL ou TLS, seu sucessor. VPNs SSL podem conectar localidades que tenham problemas de NAT com o IPsec ou *firewalls*, por exemplo.

2.6.2.3 L2TP

Layer 2 Tunnel Protocol é um protocolo para VPN que não utiliza nenhuma encriptação por si só. Por esse motivo, normalmente a tecnologia é referida em conjunto com o protocolo IPsec, uma extensão do protocolo IP que garante

privacidade dos dados com autenticação e encriptação de cada pacote IP que trafega na rede.

Por passar a informação pela conversão de dois protocolos, L2TP e IPsec, a sobrecarga do tráfego com essa tecnologia é maior que a dos outros protocolos com encriptação embutida. Além disso, não é possível trafegar a informação por uma porta que não seja a 500 (UDP). Isso torna mais difícil driblar o bloqueio de muitos *firewalls*.

2.6.2.4 OpenVPN

OpenVPN não é um protocolo em si, ainda que comumente seja descrito como. Ele é, na verdade, um programa open source que implementa uma VPN. Porém, OpenVPN utiliza versões abertas de SSL/TLS para encriptação e um protocolo próprio para tunelamento.

Apesar de, diferente das tecnologias anteriores, forçar a instalação de *softwares* de terceiros, as vantagens do OpenVPN são a customização do aplicativo e a livre escolha da porta onde será aberto o túnel. Com diversas opções de encriptação, é possível escolher entre maior segurança ou menos sobrecarga dos dados trafegados. A escolha da porta é importante pois pode dificultar que a informação seja interceptada por *firewalls*. É possível definir a porta 443 (TCP) como padrão e, com a encriptação SSL, o tráfego é facilmente confundido com o padrão HTTPS.

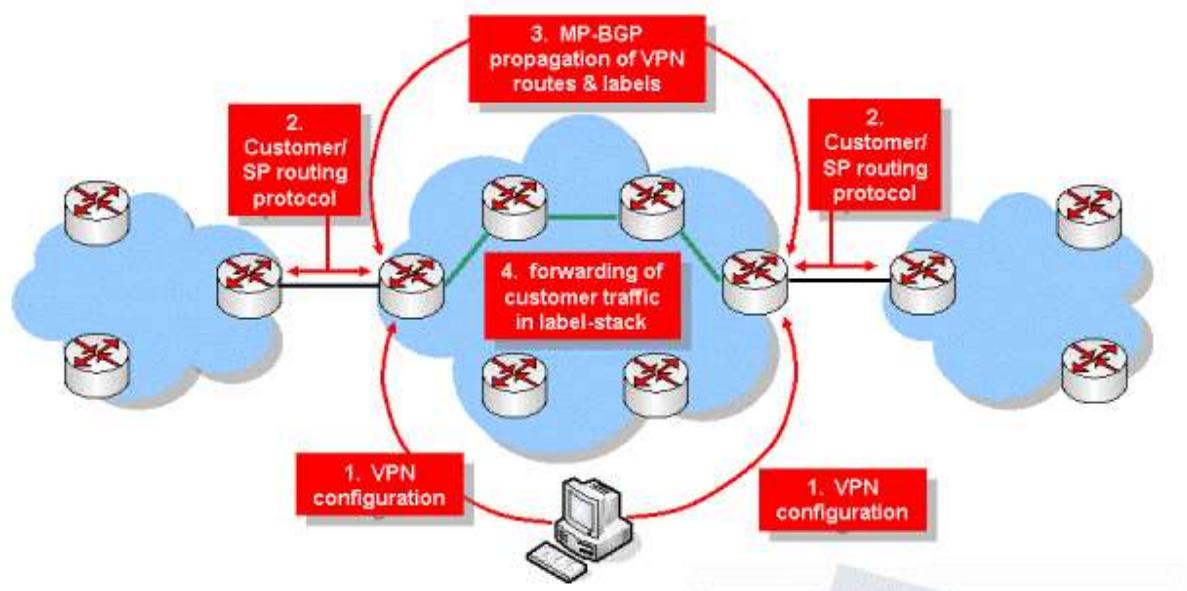
2.6.3 VPN com MPLS

MPLS e VPN são tecnologias distintas que não precisam uma da outra para ser utilizadas. Porém, existe certa confusão com as nomenclaturas empregadas para definir VPNs que utilizam MPLS difundida principalmente pela multiplicidade de termos que empresas normalmente empregam para definir o mesmo serviço.

Como descrito, VPN é uma rede privada que se estende por uma rede pública, normalmente a internet, aplicando protocolos de tunelamento e técnicas de encriptação para tornar esse acesso seguro. MPLS é uma tecnologia usada para aumentar a velocidade de transmissão de pacotes em uma rede sem se importar com o protocolo de rede que o pacote transferido está utilizando. Definimos genericamente MPLS VPN como uma VPN que é configurada em torno de uma rede baseada em

MPLS, geralmente de um provedor de serviços [LEWIS, 2006]. A figura 2-3 explica o funcionamento de uma MPLS VPN [Technology Training Limited, 2016].

Figura 2-3. Funcionamento de uma MPLS VPN



Fonte: Site do Technology Training Limited [7]

MPLS VPN engloba implementações diferentes, cada uma operando em uma camada do modelo OSI e com um objetivo distinto possuindo vantagens e desvantagens para cada tipo de tráfego que será manipulado. Existem, primariamente, dois tipos de VPNs que se utilizam de MPLS, que são as chamadas VPNs de camada 3 e VPNs de camada 2.

2.6.3.1 Layer 3 MPLS VPN (VPRN)

MPLS VPN de camada 3 são conhecidas como *Virtual Private Routed Networks* (VPRN).

Nessa implementação, cada VPN necessita de um equipamento para fazer sua borda com os PEs, necessariamente roteadores ou computadores, chamados de *Customer Edge* (CE). *Switches Layer 2* são transparentes e não servem como dispositivo de borda para o provedor de serviços .

Os dispositivos CE podem usar protocolos de roteamento distintos, como OSPF e BGP, para comunicação com o roteador PE para carregar prefixos IP na rede do provedor de serviços. O uso de VRF possibilita que roteador PE mantenha uma tabela

de roteamento diferente para cada cliente, além de uma tabela padrão para casos onde o endereço desejado não seja encontrado na tabela de roteamento do cliente.

Os PEs trocam informação das tabelas de roteamento com outros roteadores dentro da rede MPLS via MBGP. Porém, como um mesmo endereço pode se repetir para várias VPNs, as rotas são disponibilizadas com o RD, que identifica a VPN a qual eles pertencem.

Quando um CE se conecta à rede VPN, o PE procura em suas tabelas de roteamento qual pertence à VPN que foi informada. Ao encontrar, ele pode definir a etiqueta MPLS para endereçar o pacote ao túnel correto.

2.6.3.2 Layer 2 MPLS VPN (VPLS)

MPLS VPN de camada 2 são conhecidas como *Virtual Private LAN Services* (VRLS). Os clientes em uma VPLS atuam como se estivessem na mesma LAN, independente da localidade onde estão.

No VPLS, cada VPN é associada a um PE que recebe apenas endereços da camada 2, como endereços MAC. A partir desse endereço, o PE identifica em uma tabela interna chamada *Virtual Forwarding Instance* (VFI) a qual VPN ele possui. Os PEs aprendem endereço de camada 2 do mesmo jeito que *switches* convencionais, exceto pelo fato que os *frames* são recebidos através de Circuitos Virtuais (*Virtual Circuits* - VCs) e não portas físicas.

É necessário criar uma malha de VCs entre os PEs que estão conectados aos sites que fazem parte de cada VPN. Os VCs devem ser configurados nos dois sentidos do tráfego quando necessário, pois funcionam de forma unidirecional. As VPNs dos clientes são identificadas por um ID único formado por 32 bits. Dentro da rede MPLS, os *frames* são transferidos sempre com a etiqueta que determina o VC e a que identifica o LSP a qual o *frame* pertence.

Os roteadores do núcleo da rede MPLS não aprendem endereços de camada 2 porque fazem a comutação baseados nas etiquetas associadas aos frames.

Caso um mesmo site de um mesmo cliente participe de duas VPNs distintas, o tráfego de ambas deve ser separado conectando cada uma em portas diferentes do roteador PE. Alternativamente, o tráfego pode ser multiplexado sobre a mesma conexão física utilizando dois VLAN IDs diferentes.

O IETF não define um padrão de implementação para VPLS, mas aceita duas metodologias, definidas em dois *drafts*. O de Kompella utiliza BGP tanto para descoberta de outros PEs quanto para estabelecer o *full-mesh* entre todos os PEs na mesma VPLS. Já o de Martini utiliza LDP para estabelecer o *full-mesh*, porém não existe descoberta automática de novos PEs na rede.

2.6.3.3 VPRN x VPLS

Apesar de oferecer o mesmo serviço de VPN, as duas implementações possuem vantagens e desvantagens em sua aplicação, manutenção e escalabilidade se comparadas uma com a outra.

Na VPLS, como a camada de rede não é analisada, o roteamento fica por conta do cliente. Isso é vantajoso por aceitar qualquer protocolo de camada de rede utilizado e, para o cliente, ter mais liberdade para definir o tráfego. Por outro lado, implementar QoS é difícil e, na maioria das vezes, inclui um *overhead* aos *frames* para solucionar. Já VPRN, por compreender a camada 3, sabe com antecedência o tipo de serviço a qual ele pertence e pode direcionar o tráfego da melhor maneira.

Utilizando VPRN, a adição de uma novo site para o cliente não necessita reconfiguração da rede MPLS existente. Já no caso de VPLS, é necessário definir os VCs que determinam o caminho que o pacote deve seguir para os endereços de camada 2 do novo site.

Como VPLS é uma rede *full-mesh*, a falha de um dispositivo pode resultar em perda de conectividade de algum site na rede. Em VPRN, as falhas são menos críticas porque os roteadores se comunicam via MBGP dentro do domínio MPLS e alteram os LSPs necessários.

Em resumo, VPLSs é um serviço que dá mais autonomia para o cliente, é mais rápido e possui um custo a curto prazo menor. São indicadas para quem quer mais privacidade com seus IPs ou não possui um IP público. Além disso, disponibilizam melhor serviços de tempo real. Já VRPNs são mais robustas, confiáveis e, a longo prazo, a manutenção é mais fácil. Definições de QoS são mais fáceis de ser estabelecidas, roteando cada tipo de tráfego por um caminho diferente.

2.7 VRF

Virtual Routing and Forwarding (VRF) é uma tecnologia inclusa em roteadores que implementam o protocolo IP que permite que múltiplas instâncias da tabela de roteamento coexistam no mesmo roteador simultaneamente. VRFs são comumente utilizadas por ISPs para criar VPNs para diferentes consumidores, por isso a tecnologia também é conhecida como *VPN Routing and Forwarding*.

Com VRF, podemos separar um mesmo roteador em diversos roteadores lógicos onde cada interface trabalha com uma tabela de roteamento diferente. Desse modo, é possível que rotas idênticas possam ser encaminhadas para endereços distintos baseados em tabelas de roteamento configuradas separadamente para cada VRF. Apesar de não ser um conceito exclusivo de MPLS, VRFs são normalmente associadas a este, justamente por ser uma das ferramentas que auxiliam os projetistas de redes MPLS por serem completamente compatíveis com MBGP.

Podemos configurar uma VRF para ser de único protocolo ou para funcionar com VPN-IPv4. Na definição da VRF definimos as importações e exportações de RD cujas tabelas devem endereçar. Desse modo, cada tabela de roteamento pode endereçar uma VPN distinta, facilitando e agilizando o processo.

2.8 MPLS

Multiprotocol Label Switching (MPLS) é um protocolo projetado para agilizar o transporte de pacotes entre os roteadores de uma rede. Ele é mais um protocolo de *tunneling*, que faz o transporte de dados, operando entre a camada de dados e a camada de rede.

Pelo seu alto rendimento e compatibilidade com diversos tipos de tráfego, o MPLS elimina a dependência em tecnologias particulares e permite que muitos dos pacotes sejam transferidos na camada 2 do modelo OSI, comumente referida como enlace, evitando que um passo a mais no processo de roteamento seja feito ao permitir que os cabeçalhos de camada de rede sejam ignorados e buscas mais custosas em tabelas de roteamento sejam evitadas^[4].

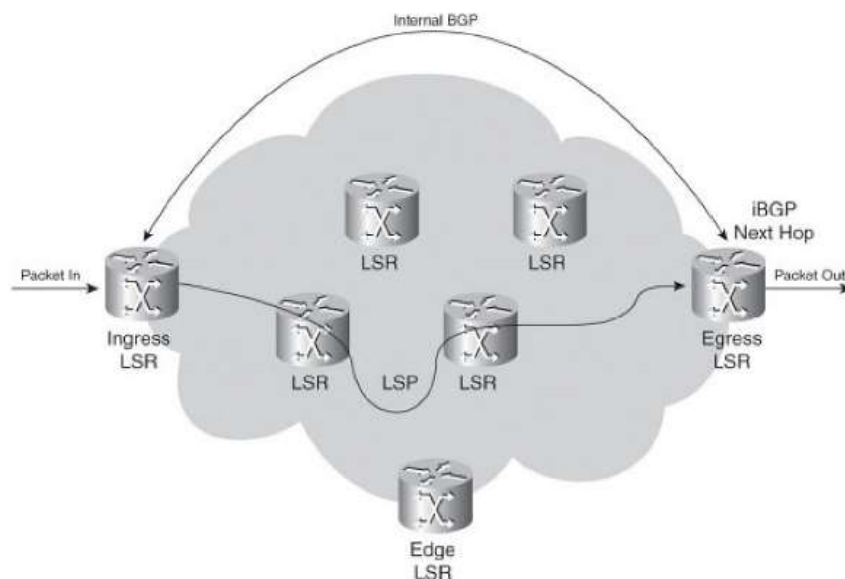
O protocolo MPLS define seus próprios caminhos dentro da rede e os roteadores que recebem os pacotes nas bordas da rede MPLS utilizam outro protocolo chamado *Label Distribution Protocol* (LDP), que se encarrega de adicionar uma ou

mais *labels* (etiquetas) aos pacotes, baseadas no destino destes [SearchEnterpriseWAN, 2016]. Essa etiqueta determina previamente o caminho que o pacote irá seguir, sem necessidade de avaliar o conteúdo do pacote. Os caminhos também são definidos com antecedência para cada tipo de tráfego transmitido.

Os caminhos pré-determinados dentro de uma rede MPLS são chamados de *Label Switched Paths* (LSP). Cada LSP é uma via de mão única, sendo necessário configurar um para cada direção do tráfego caso necessário.

Quem define qual LSP um pacote deve seguir são os roteadores na borda da rede MPLS, também chamados de *Label Edge Routers* (LER). LERs que recebem pacotes entrando na rede MPLS são chamados de LER de ingresso, enquanto os que encaminham o pacote na saída da rede são chamados de LER de egresso. Cada LSP possui um LER de ingresso e um de egresso e podem possuir outros roteadores no meio do caminho, no núcleo da rede MPLS, conhecidos como *Label Switching Routers* (LSR)^[6]. Na figura 2-4, a topologia de uma rede MPLS [GHEIN, 2007].

Figura 2-4: Topologia MPLS



Fonte: (GHEIN, 2007, p. 173)

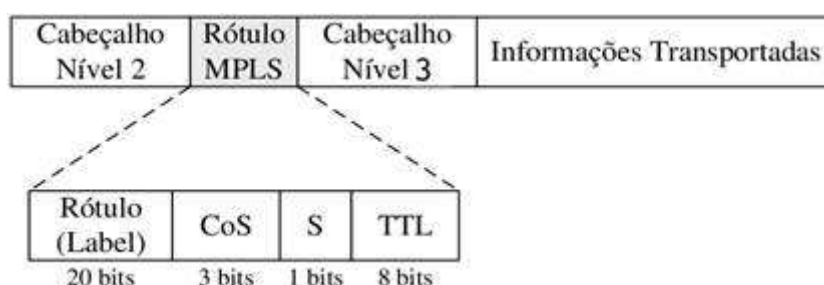
Uma das razões para se utilizar de MPLS é ter suas rotas do *core* baseadas nestas *labels*, ao invés de pacotes IP. Ao chegar em um LER, o pacote é analisado e Classes de Encaminhamento Equivalente (*Forwarding Equivalence Class* - FEC) são determinadas. FECs são as possibilidades de encaminhamento do pacote através da rede^[6]. A seguir, são adicionadas etiquetas ao cabeçalho MPLS do pacote indicando

o próximo passo no caminho do pacote dentro da LSP. Quando um pacote sem etiqueta entra num roteador de acesso e necessita de passar num túnel MPLS, o roteador determina o FEC do pacote e insere uma ou mais etiquetas no *header* MPLS que foi criado para o efeito. Quando este pacote sai do túnel MPLS, este *header* é removido.

O MPLS utiliza, assim, *label stacking* (empilhamento de etiquetas) para poder suportar túneis de conexões dentro de conexões^[6] [Teleco, 2016]. Quando um pacote contendo etiquetas é recebido por um roteador MPLS, sua etiqueta mais externa é analisada. Cabe ao roteador, então, analisar e procurar em suas tabelas de endereçamento qual o próximo passo no transporte do pacote baseado na etiqueta, sem olhar o cabeçalho de rede. Depois de decidido o próximo passo, o roteador pode trocar a etiqueta analisada (*swap*), adicionar outra etiqueta acima dessa encapsulando o pacote em mais um nível MPLS (*push*) ou remover a etiqueta (*pop*). A figura 2-5 demonstra como o cabeçalho MPLS transmite essas informações^[4].

Se a etiqueta removida for última do pacote, ele deixa o túnel MPLS. Essa ação é normalmente realizada pelos LERs de egresso, a menos que a rede MPLS esteja configurada para executar o *Penultimate Hop Popping*^[4]. Neste caso, a última etiqueta é retirada sempre no último LSR antes do LER de egresso. Essa configuração é utilizada normalmente para diminuir o processamento nos LERs.

Figura 2-5: Cabeçalho MPLS



Fonte: Site da Teleco, 2016 [8]

As vantagens de utilizar uma rede baseada em MPLS incluem:

- Engenharia de Tráfego (QoS): os cabeçalhos MPLS são incluídos nos roteadores da borda da rede MPLS, o que possibilita identificar o tipo de tráfego que está sendo transferido e, por meio das etiquetas, direcionar o pacote por um caminho diferente,

permitindo que cada classe de dados (voz, vídeo, dados, etc) tenha uma característica individual de performance. Tráfego de dados padrão pode ser direcionado para um caminho de menor prioridade, enquanto dados de *streaming* em tempo real utilizam um caminho de prioridade maior ou de menos utilizado. Isso também permite que se especifique diferentes latências, limites mínimos de perda de pacotes e *jitter* para diferentes caminhos dentro da rede MPLS.

- Processamento de tráfego: uma vantagem inerente da forma como a rede MPLS opera tem como consequência a diminuição do processamento dos pacotes no núcleo da rede, já que a parte pesada do processamento é feita nas bordas da rede onde o fluxo de pacotes é menor, tornando o núcleo mais rápido e, portanto, mais eficiente.
- Expansão: a inclusão de novas VPNs na rede MPLS apresenta baixo custo de implementação e fácil configuração em relação à outras tecnologias de redes privadas. Caso seja necessária uma nova filial para se conectar à VPN, basta que esta tenha uma conexão com a rede MPLS já estabelecida.
- Acordo de Nível de Serviço (*Service Level Agreement* - SLA): é um compromisso oficial, geralmente um contrato, onde um provedor de serviço e seu cliente acordam sobre algum aspecto da qualidade do serviço, como disponibilidade e qualidade. No caso das redes MPLS, provedores de serviço geralmente garantem que a indisponibilidade do serviço será mínima. É possível analisar a performance de forma contínua e detectar instantaneamente erros para suporte e correção rápida.

2.8.1 CEF Cisco

Cisco Express Forwarding (CEF) é uma tecnologia de camada 3 para comutação de pacotes IP usada majoritariamente em redes de grande escala para melhorar a performance de modo geral em redes CISCO^[12] [CISCO SYSTEMS, 2016].

CEF é utilizado para aumentar a velocidade na comutação de pacotes reduzindo a sobrecarga e *delays* que outras técnicas de roteamento apresentam. Ele possui uma tabela chamada *Forwarding Information Base* (FIB) similar à de tabela roteamento da maioria dos protocolos de roteamento que mantém apenas o próximo passo para cada endereço IP. Outro componente do protocolo é a tabela de adjacências, que mantém informações de camada 2 para cada entrada na tabela FIB, evitando a necessidade de mais buscas mais custosas.

Para o funcionamento de MPLS nos roteadores Cisco, é necessário que o CEF esteja habilitado e devidamente configurado. CEF permite que o encaminhamento de dados requeridos para o funcionamento da rede MPLS seja feito de forma correta. Isso não significa que CEF é necessário para o funcionamento de MPLS ou que funcione apenas em roteadores Cisco. Outros vendedores disponibilizam seus próprios protocolos análogos que permitem comutação na camada 3 ou o roteamento é feito no próprio hardware do dispositivo, mas no caso específico entre equipamentos CISCO, se faz necessária sua utilização.

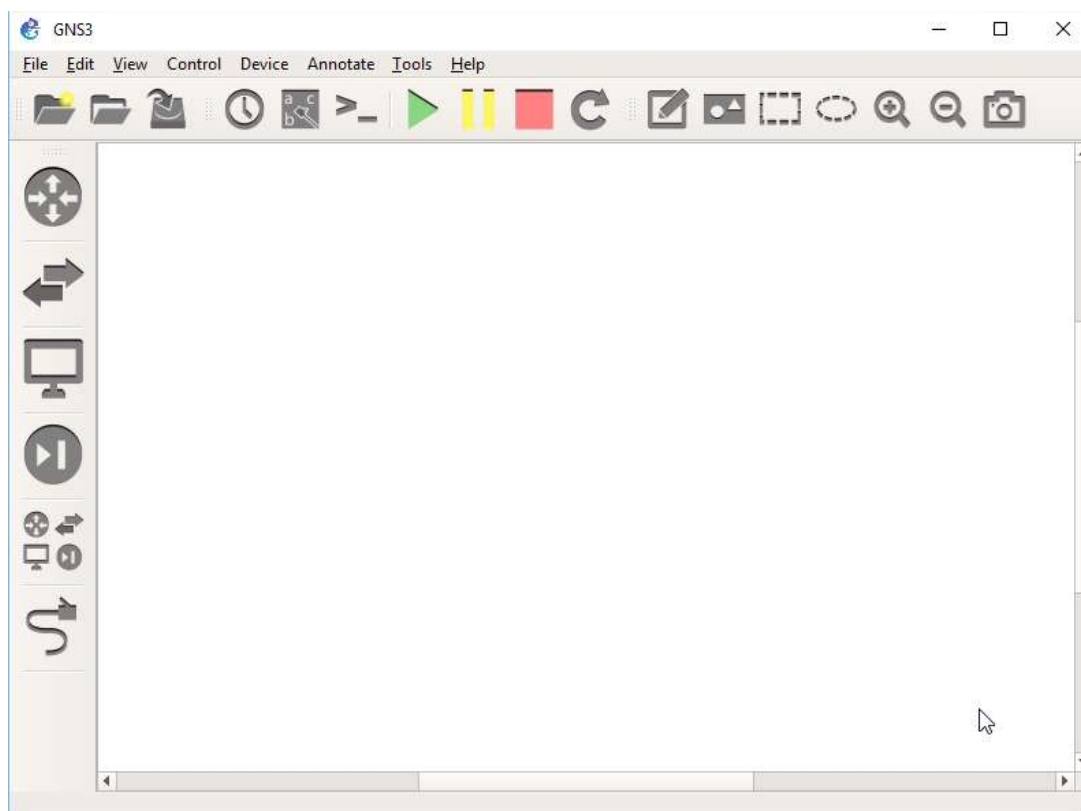
3 Experimento

3.1 GNS e Dynamips

O GNS3, abreviação para *Graphical Network Simulator*, é um emulador de redes *open source*, que permite a combinação de equipamentos reais e virtuais para criar redes complexas por meio de emulação de softwares reais da CISCO. Ele possui uma interface gráfica bastante intuitiva, e por meio da utilização de sistemas operacionais reais, torna possível a criação de múltiplos tipos de redes. Ele foi lançado em 2005 e vem passando por atualizações desde então. Hoje, o software encontra-se na versão 3.4.3 - e foi esta a versão usada no experimento a seguir.

O GNS3, demonstrado na figura 3-1, oferece maneiras diferentes para emular versões de IOSs, sendo a utilização do Dynamips o mais popular e indicado entre elas, por conta da ampla comunidade que usa e vem incrementando discussões sobre o software desde o seu lançamento, ainda em 2008. O Dynamips é um software de virtualização dedicado para rodar sistemas operacionais CISCO reais num computador, e foi com ele que o experimento foi criado. Também é possível utilizar o VMWare VirtualBox ou o Oracle VirtualBox para emular os softwares de *switches* e roteadores. O GNS cria uma interface para utilização do Dynamips que permite criar várias instâncias de equipamentos diferentes, assim como inclusão de placas nos mesmos e criação de links entre os equipamentos.

Figura 3-1. A tela Inicial do GNS3



O Dynamips trabalha emulando hardware de roteadores, *switches* e *switches* L3, incluindo uso de CPU, RAM, interfaces e console no servidor em que é utilizado. A imagem do sistema operacional utilizado não sabe que está sendo executada em um equipamento virtual, o que faz com que os IOS CISCO realmente tentem enviar pacotes por interfaces físicas, cabendo ao Dynamips interceptar estes pacotes de maneira a enviá-los para o próximo roteador virtual da topologia criada.

É possível utilizá-lo inclusive para encaminhar pacotes para interfaces LAN reais, fazendo com que outros equipamentos que não o computador executando o Dynamips recebam este tráfego. No estudo apresentado a seguir, não foram utilizadas estas funcionalidades externas à topologia criada, mas usando Dynamips foi possível conectar à console de cada um dos equipamentos, realizar telnet para os roteadores, e ter acesso a todos os comandos reais do sistema operacional escolhido.

3.2 Detalhes do experimento

3.2.1 Propósito do Experimento

O experimento teve como objetivo criar um *backbone* privado, que poderia ser administrado por um provedor de serviços, por exemplo, e por meio deste único *backbone* transmitir o tráfego de dois clientes diferentes utilizando VRFs distintas. A implementação do roteamento interno deste *backbone* utilizou OSPF, sincronizado com o LDP, que foi o protocolo de distribuição de labels escolhido na utilização do MPLS. Para a transmissão de tabelas de roteamento distintas, ou VRFs distintas, os roteadores das bordas do CORE *backbone* (PEs) conversam MBGP entre si, que acaba só sendo visto por estes dois e não pelos outros agentes do transporte MPLS, visto que este é um protocolo de camada menor.

O cenário dita que estes clientes, retratados nele como Cliente A e Cliente B visam transmitir dados usando VPNs *layer 3*, e tem seus roteadores de borda (CEs) ligados ao *backbone* em dois pontos diferentes por meio de roteadores em comum (PEs) nas bordas do *backbone*. Estas saídas diferentes dos CEs também sugerem que posições geográficas distintas poderiam estar sendo empregadas, e por isso o experimento também ilustrou diferentes bairros do município Rio de Janeiro e a cidade de Duque de Caxias para os 4 CEs envolvidos - Dois para o Cliente A e dois para o Cliente B.

Para comunicação entre PE-CE, o Cliente A se utilizou de BGP como protocolo EGP para transmissão de dados. O Cliente B se utilizou de OSPF.

O estudo envolve, assim, diferentes conceitos de roteamento visando segurança de dados, garantia de manutenção de serviço por parte do *service provider* com rotas de transbordo no *backbone* e integridade e sincronismo de dados. A topologia será apresentada nos tópicos a seguir e o plano de endereçamento está descrito na tabela 3-1.

Todos os comandos utilizados no estudo estão na documentação oficial da CISCO, referenciada ao final do trabalho em seus links oficiais. Foram usados, assim, os documentos com referências de comandos de MPLS Label Distribution Protocol (LDP), OSPF e BGP, que contemplam todo o conteúdo necessário para a configuração do experimento.

3.2.1.1 Equipamento escolhido

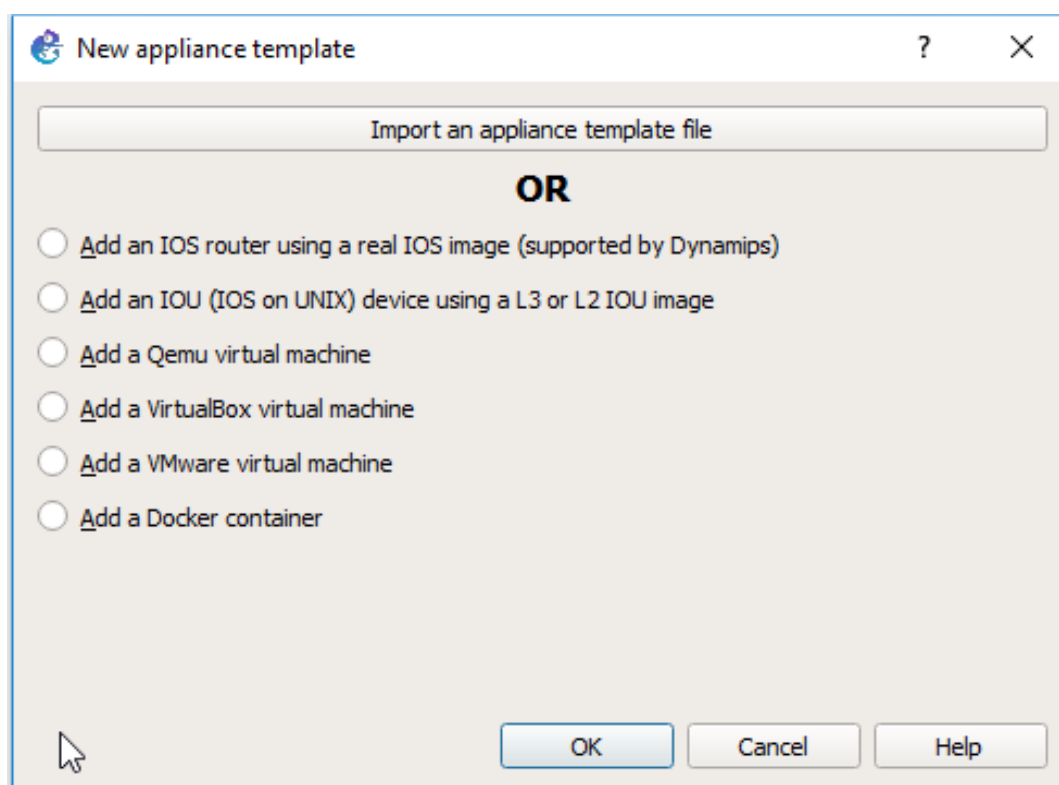
Todos os roteadores do experimento são emulações de CISCOs C3725 - versão de software c3725-124-25d. Este roteador da série 3700 foi escolhido por possuir um consumo médio de 128MBs de memória RAM por instância emulada, o que garantiu a possibilidade de criar 9 roteadores com tranquilidade em uma máquina com 16GB de RAM, e por suportar todos os protocolos mencionados nos capítulos anteriores.

Configuração *as is* do equipamento emulado:

R7000 CPU at 240MHz, Implementation 39, Rev 2.1, 256KB L2, 512KB L3 Cache
18 FastEthernet interfaces
DRAM configuration is 64 bits wide with parity enabled.
55K bytes of NVRAM.

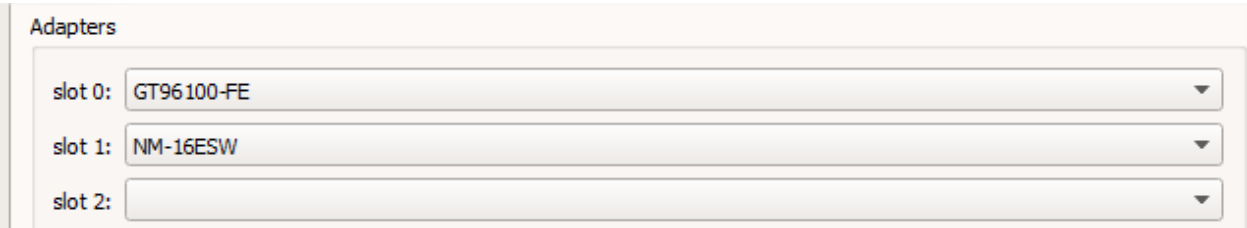
Vale ressaltar que o consumo médio de processamento de CPU por roteador do Dynamips na emulação dos sistemas operacionais é alto e tende a ocupar todo o processamento nativo do processador do computador, mas ele possui um controlador que mantém os roteadores que estão conectados na topologia em um estado de *idle* quando não estão em uso no momento. Este comportamento de balanceamento de carga de processamento impede a utilização de 100% da CPU do computador onde o GNS está sendo utilizado.

Figura 3-2. Diferentes métodos de criação de uma instância de roteador no GNS3



A criação de um *template* de roteador, demonstrada na figura 3-2, funciona como uma configuração global de um elemento que pode ser iniciado no GNS se dá de maneira bem prática, e conta com um *add-on* que garante a adição de placas com mais interfaces além das padrões de fábrica do modelo emulado. Visto que o modelo usado só possui duas portas *Fast Ethernet* por padrão, a adição de uma placa com mais portas se fez necessária para conversa entre mais roteadores. Todos os roteadores foram, assim, adaptados com uma placa NM-16ESW, que adiciona mais 16 portas *Fast Ethernet* em cada instância a ser criada, conforme a figura 3-3 ilustra.

Figura 3-3. Slots e adaptadores para inclusão em uma instância de roteador emulado



The image shows a configuration window titled "Adapters" with a light beige background. It contains three rows, each representing a slot. Each row has a label on the left and a dropdown menu on the right. The first row is labeled "slot 0:" and has a dropdown menu showing "GT96100-FE". The second row is labeled "slot 1:" and has a dropdown menu showing "NM-16ESW". The third row is labeled "slot 2:" and has an empty dropdown menu. Each dropdown menu has a small downward-pointing arrow on its right side.

Slot	Adapter
slot 0:	GT96100-FE
slot 1:	NM-16ESW
slot 2:	

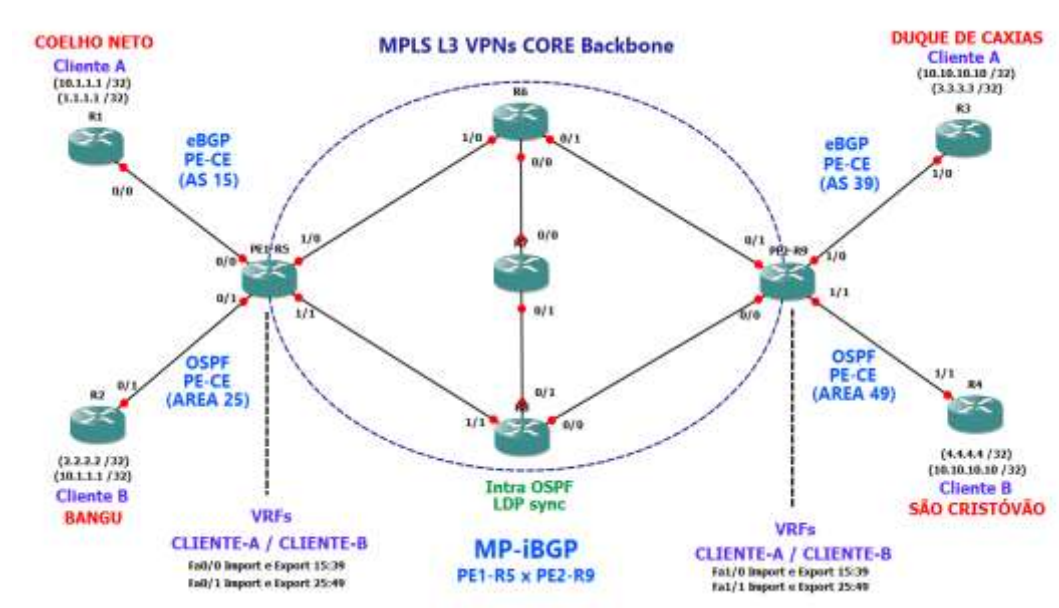
3.2.1.2 Mapeamento de links da rede emulada

Tabela 3-1. Plano de endereçamento do experimento

Roteador	Função do roteador	Tipo de interface	ID da Interface	IP da interface
R1	Customer Edge - CLIENTE A	Loopback	0	1.1.1.1
R1	Customer Edge - CLIENTE A	Loopback	1	10.1.1.1
R1	Customer Edge - CLIENTE A	FastEthernet	0/0	192.168.15.1
R2	Customer Edge - CLIENTE B	Loopback	0	2.2.2.2
R2	Customer Edge - CLIENTE B	Loopback	1	10.1.1.1
R2	Customer Edge - CLIENTE B	FastEthernet	0/1	192.168.25.2
R3	Customer Edge - CLIENTE A	Loopback	0	3.3.3.3
R3	Customer Edge - CLIENTE A	Loopback	1	10.1.1.1
R3	Customer Edge - CLIENTE A	FastEthernet	1/0	192.168.39.3
R4	Customer Edge - CLIENTE B	Loopback	0	4.4.4.4
R4	Customer Edge - CLIENTE B	Loopback	1	10.1.1.1
R4	Customer Edge - CLIENTE B	FastEthernet	1/1	192.168.49.4
R5	Provider Edge	Loopback	0	5.5.5.5
R5	Provider Edge	FastEthernet	0/0	192.168.15.5
R5	Provider Edge	FastEthernet	0/1	192.168.25.5
R5	Provider Edge	FastEthernet	1/0	172.16.56.5
R5	Provider Edge	FastEthernet	1/1	172.16.58.5
R9	Provider Edge	Loopback	0	9.9.9.9
R9	Provider Edge	FastEthernet	0/0	172.16.89.9
R9	Provider Edge	FastEthernet	0/1	172.16.69.9
R9	Provider Edge	FastEthernet	1/0	192.168.39.9
R9	Provider Edge	FastEthernet	1/1	192.168.49.9
R6	Provider	Loopback	0	6.6.6.6
R6	Provider	FastEthernet	0/0	172.16.67.6
R6	Provider	FastEthernet	0/1	172.16.69.6
R6	Provider	FastEthernet	1/0	172.16.56.6
R7	Provider	Loopback	0	7.7.7.7
R7	Provider	FastEthernet	0/0	172.16.67.7
R7	Provider	FastEthernet	0/1	172.16.78.7
R8	Provider	Loopback	0	8.8.8.8
R8	Provider	FastEthernet	0/0	172.16.89.8
R8	Provider	FastEthernet	0/1	172.16.78.8
R8	Provider	FastEthernet	1/1	172.16.58.8

3.2.1.3 Topologia do experimento

Figura 3-4. Topologia do experimento



3.3 Implementação do experimento

A implementação do experimento, cuja topologia pode ser vista na figura 3-4, foi dividida em 3 partes principais, que serão assim demonstradas a seguir para entendimento sequencial e construção de conhecimento passo a passo com os comandos utilizados, apoiados pela topologia apresentada acima. São eles:

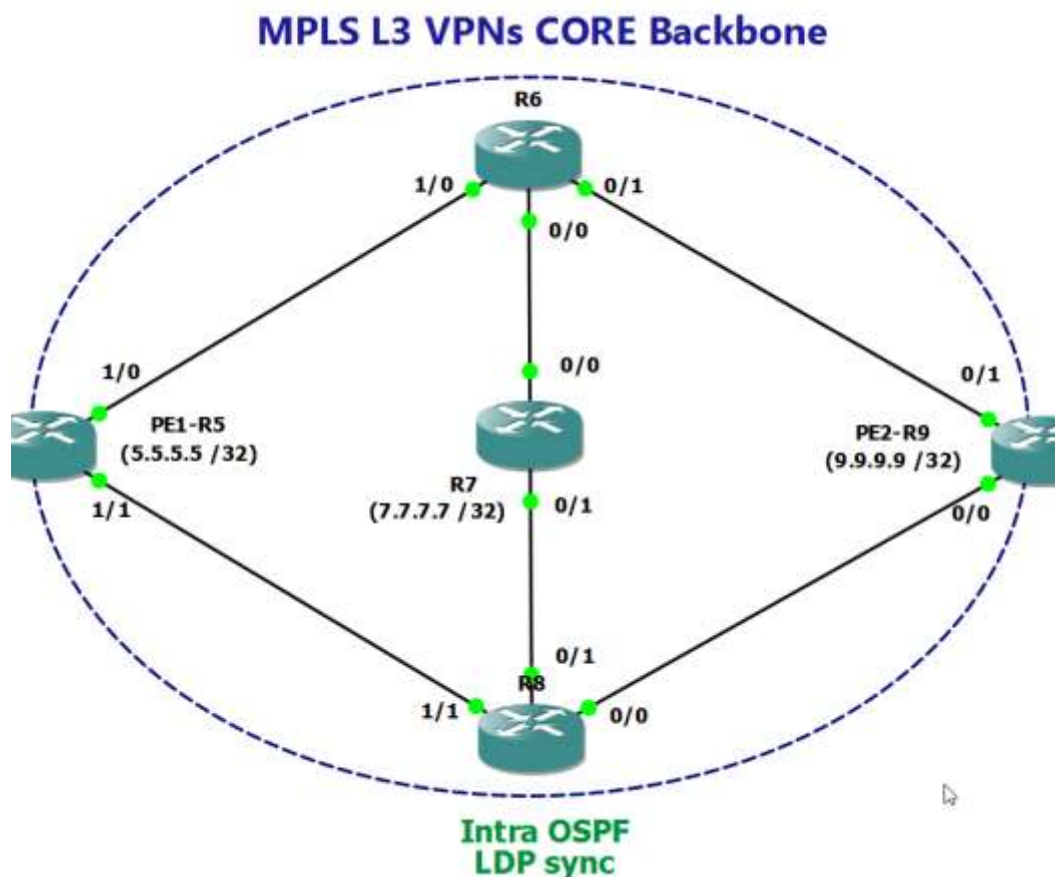
- Implementação do CORE MPLS no *Backbone* - Core OSPF e MPLS LDP, com configuração de todos os roteadores da nuvem MPLS;
- Implementação de roteamento BGP CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-A;
- Implementação de roteamento OSPF CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-B

3.3.1 Configuração do MPLS no CORE *backbone*

As configurações a seguir, relativas às integrações LDP/MPLS e OSPF do PE-R5, que foram realizadas de forma análoga no PE-R9, visam preparar o roteamento do *backbone* central (figura 3-5) e a nuvem MPLS como um todo, que será utilizada

no resto do experimento. Ainda neste capítulo, são mostradas também configurações do R7, a serem replicadas em R6 e R8.

Figura 3-5. Backbone MPLS



Abaixo é criada uma interface de *loopback* para o roteador PE-R5 (figura 3-6), para fins de conectividade. Todos IPs /32 (255.255.255.255).

Os roteadores de toda a topologia terão IPs de *loopback* com duas finalidades-chave:

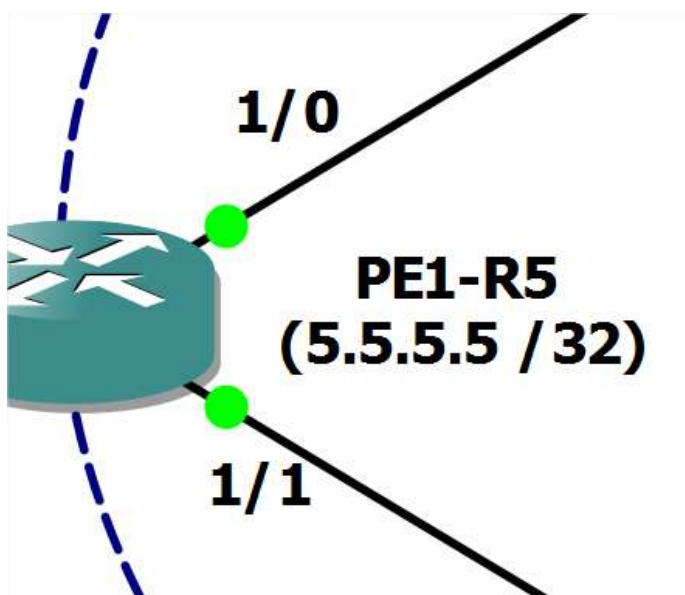
- 1) Ser endereços que identificam unicamente os roteadores.
- 2) Ser usados como identificadores para propriedades específicas de alguns protocolos, como será visto a seguir.

O comando *configure terminal* entra em modo de configuração global para implementação de informações na tabela de configurações chamada *running configuration*, que é a tabela de configurações usadas pelos roteadores CISCO no momento da execução dos comandos. Existe também um arquivo de configurações chamado *startup config*, que é aquele que tem informações salvas para eventuais

boots do sistema, e que geralmente se originam do *running config* via intervenção manual para salvar as configurações.

```
PE1-R5#
PE1-R5#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
PE1-R5(config)#
PE1-R5(config)#interface loopback 0
PE1-R5(config-if)#
PE1-R5(config-if)#ip address 5.5.5.5 255.255.255.255
PE1-R5(config-if)#
```

Figura 3-6 - Roteador R5



Saindo do modo de configuração Global:

```
PE1-R5(config-if)#exit
PE1-R5(config)#
```

Configuração de MPLS com uso explícito de LDP:

```
PE1-R5(config)#
PE1-R5(config)#mpls label protocol ?
    ldp Use LDP (default)
    tdp Use TDP
PE1-R5(config)#mpls label protocol ldp
PE1-R5(config)#
```

Especificando uma interface preferida para como LDP router ID com o comando *mpls ldp router-id*.

O LDP Router ID é uma interface que identifica unicamente o roteador dentro de uma nuvem MPLS.

Caso não seja definido de forma explícita, o roteador escolhe uma interface conforme documentação da CISCO.

É escolhido o maior IP de loopback. Na ausência de um deste tipo, é escolhido o maior IP operacional no roteador.

Por usarmos interfaces de loopback, esta será a escolhida no R5 e nos demais roteadores da nuvem do experimento: R9, R6, R7 e R8.

```
PE1-R5(config)#
PE1-R5(config)#mpls ldp router-id loopback 0
PE1-R5(config)#
```

Configurar o CEF, explicado no capítulo 2.8.1, é necessário para funcionamento em equipamentos Cisco.

```
PE1-R5(config)#
PE1-R5(config)#ip cef
```

Os comandos abaixo definem intervalos de *Label ID* para serem utilizados pelo processo MPLS.

Todos os roteadores da nuvem MPLS terão intervalos de 100 possíveis *labels*, começando sempre com o dígito do seu número de roteador.

Neste caso, o PE é o R5, e terá um range de 500-599. Isso significa que no cabeçalho Layer 2 das mensagens enviadas, os primeiro 20 bits definem o *Label ID*.

```
PE1-R5(config)#mpls label ?
  protocol  Set platform default label distribution protocol
  range     Label range
PE1-R5(config)#mpls label range 500 599
```

O próximo passo é a criação de um processo OSPF (neste caso, configurado como *router ospf 1*), de modo a configurar o roteamento interno da nuvem.

Todo roteador em uma rede OSPF precisa também de um endereço IPv4 para identificar unicamente este roteador na conversa OSPF.

Este endereço, conhecido como *router ID*, será o IP de *loopback*, conforme mencionado anteriormente.

```
PE1-R5#
PE1-R5#conf t
```

```

PE1-R5#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
PE1-R5(config)#
PE1-R5(config)#router ospf 1
PE1-R5(config-router)#
PE1-R5(config-router)#router-id 5.5.5.5
PE1-R5(config-router)#

```

O comando abaixo mantém o LDP em sincronia com o IGP que está em cada uma das interfaces que ambos falam.

Isto impede, por exemplo, que caso ocorra um problema temporário com o LDP em alguma interface conhecida por ele, que o IGP continue a enviar tráfego de *labels* pelo seu último caminho:

```

PE1-R5(config-router)#
PE1-R5(config-router)#mpls ldp sync

```

Então, adicionamos um IP na interface FastEthernet1/0 e ligamos a mesma com o comando *no shutdown*, que conversa com o R6.

Os IPs desta nuvem possuem prefixos 172.16.XY.Z, onde:

- XY são os pares de roteadores que estão conversando, neste caso 5 e 6 - 56
- Z é o roteador em questão, neste caso, o 5:

```

PE1-R5(config)#
PE1-R5(config-router)#interface f1/0
PE1-R5(config-if)#
PE1-R5(config-if)#no shut
PE1-R5(config-if)#
*Mar  1 01:29:52.707: %LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet1/0, changed state to up
PE1-R5(config-if)#
PE1-R5(config-if)#ip add
PE1-R5(config-if)#ip address 172.16.56.5 255.255.255.0
% IP addresses may not be configured on L2 links.

```

O aviso na última linha do trecho de código acima refere-se à impossibilidade de configurar IPs em interfaces habilitadas somente como *Layer 2*.

Para habilitar funcionalidades *Layer 3* nas mesmas, devemos usar o comando *no switchport*:

```

PE1-R5(config-if)#
PE1-R5(config-if)#no switchport
PE1-R5(config-if)#

```

```

PE1-R5(config-if)#ip address 172.16.56.5 255.255.255.0
PE1-R5(config-if)#
*Mar  1 01:33:58.631: %LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet1/0, changed state to up
*Mar  1 01:33:59.623: %TDP-5-INFO: Default-IP-Routing-Table: TDP ID
removed
PE1-R5(config-if)#

```

Adicionamos, então, um IP na interface FastEthernet1/1 que conversa com o R8 e ligamos a mesma (com o comando *no shutdown*),:

```

PE1-R5(config)#int f1/1
PE1-R5(config-if)#
PE1-R5(config-if)#no switchport
PE1-R5(config-if)#
PE1-R5(config-if)#ip add 172.16.58.5 255.255.255.0
PE1-R5(config-if)#
PE1-R5(config-if)#no shut
PE1-R5(config-if)#
*Mar  1 01:35:41.095: %LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet1/1, changed state to up

```

Habilitamos, na sequência, o MPLS e a Area 0 OSPF em ambas as interfaces.

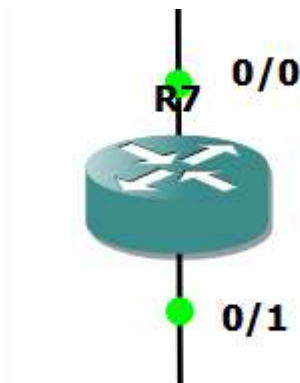
```

PE1-R5(config-if)#
PE1-R5(config-if)#mpls ip
PE1-R5(config-if)#
PE1-R5(config-if)#ip ospf 1 area 0
PE1-R5(config-if)#
PE1-R5(config-if)#int f1/0
PE1-R5(config-if)#
PE1-R5(config-if)#mpls ip
PE1-R5(config-if)#
PE1-R5(config-if)#ip ospf 1 area 0
PE1-R5(config-if)#

```

Após configurar as interfaces dos PEs que conversam com o *backbone* MPLS, para que o CORE tenha tunelamento LDP fim-a-fim, ficam pendentes as configurações dos roteadores restantes - R6, R7 e R8 - que também terão configurações semelhantes, salvos alguns detalhes adicionais. Abaixo, seguem configurações do R7 (figura 3-7), o único roteador central que não possui conexões diretas com os PEs, e que foram redistribuídas de maneira análoga no R6 e R8.

Figura 3-7. Roteador R7



O IP *loopback* de R7 foi configurado como /32 e também para falar na área OSPF 0, caso contrário, a vizinhança MPLS não é estabelecida.

```
R7#
R7#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#
R7(config)#int lo0
R7(config-if)#
R7(config-if)#ip add 7.7.7.7 255.255.255.255
R7(config-if)#
R7(config-if)#ip ospf 1 area 0
R7(config-if)#
R7(config-if)#exit
R7(config)
```

Abaixo, a configuração de MPLS com uso explícito de LDP no R7.

O *Router ID* definido como a interface de *loopback* anteriormente, com CEF ativado e *label range* variando de 700 a 799.

```
R7(config)#
R7(config)#mpls label protocol ldp
R7(config)#
R7(config)#mpls ldp router-id lo0
R7(config)#
R7(config)#mpls label range 700 799
R7(config)#
R7(config)#ip cef
R7(config)#
```

Abaixo, a configuração global de OSPF, configurando o *router ID* e ativando o sincronismo com o LDP assim como visto anteriormente.

Também é executado o comando *mpls ldp autoconfig*, que ativa a configuração que faz com que o LDP seja configurado automaticamente em todas as interfaces associadas com a instância de IGP utilizado (neste caso, o OSPF):

```
R7(config)#
R7(config)#router ospf 1
R7(config-router)#
R7(config-router)#router-id 7.7.7.7
R7(config-router)#
R7(config-router)#mpls ldp sync
R7(config-router)#
R7(config-router)#mpls ldp autoconfig
```

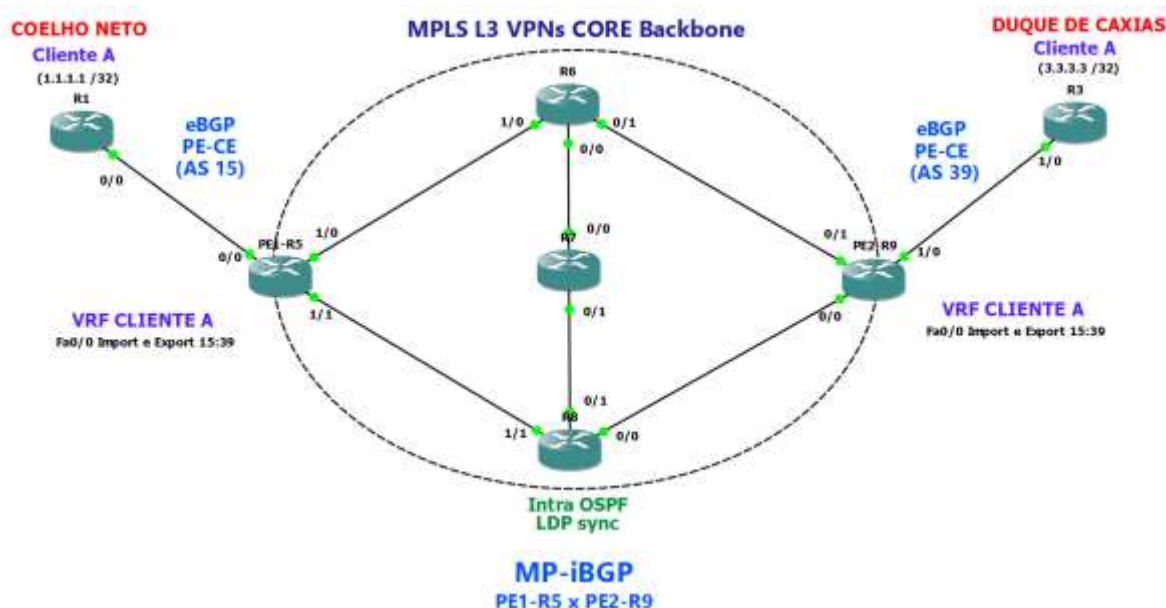
A seguir, configurações das interfaces que conversam com R6 e R8.

```
R7(config-router)#
R7(config-router)#int f0/0
R7(config-if)#
R7(config-if)#ip ospf 1 area 0
R7(config-if)#
R7(config-if)#ip add 172.16.67.7 255.255.255.0
R7(config-if)#
R7(config-if)#int f0/0
R7(config-if)#ip ospf 1 area 0
R7(config-if)#
R7(config-if)#ip add 172.16.67.7 255.255.255.0
```

3.3.2 Configuração do roteamento BGP CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-A

As configurações a seguir, feitas no PE-R5, tem por finalidade criar as instâncias de VRF do Cliente A e realizar as configurações de Multiprotocol BGP que só serão estudadas e entendidas pelos PEs, de forma que o tráfego passe pela nuvem MPLS configurada no capítulo anterior sem leitura deste tráfego. Elas foram feitas de maneira semelhante no PE-R9 para funcionamento da topologia. Na figura 3-8, o *backbone* e os PEs do Cliente A.

Figura 3-8. Topologia - MPLS CORE + Cliente A



Em R1 são feitas as configurações padrão de endereçamento de interfaces para conversar com PE-R5 (figura 3-9), assim como configurações de roteamento BGP, que neste caso atua como IGP de modo a ter o fim-a-fim estabelecido com R3. Suas configurações são também análogas àsquelas realizadas em R3 (figura 3-10). Os roteadores R6, R7 e R8 não serão mais configurados até o fim do experimento.

Figura 3-9. Elementos envolvidos nas configurações para o Cliente A

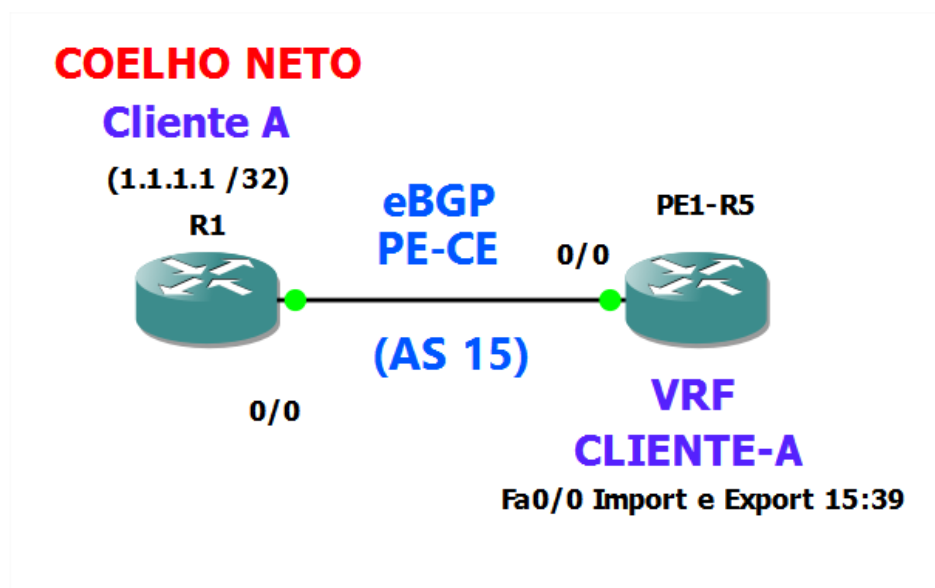
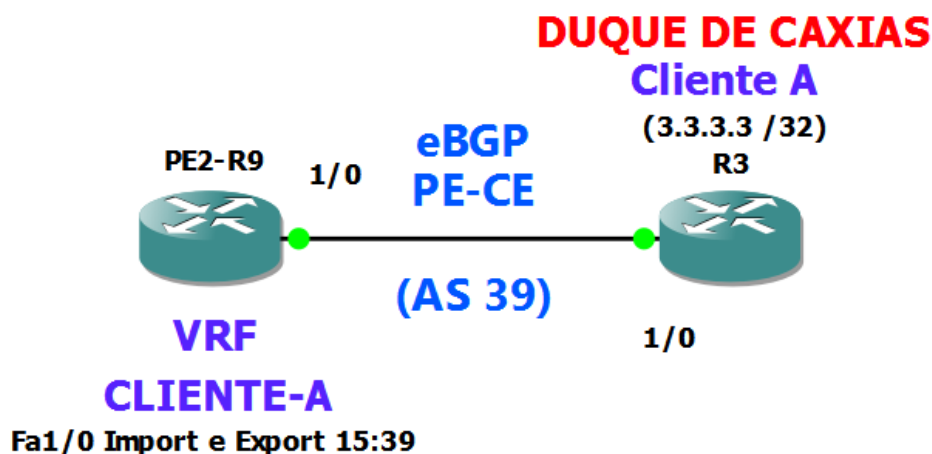


Figura 3-10. Elementos envolvidos nas configurações para o Cliente A



Como visto em capítulos anteriores, as VRFs serão instâncias de roteamento separadas nos PEs para divisão de tráfego dos dois diferentes clientes do experimento.

Definição da VRF do Cliente A, nomeada de CLIENTE-A:

```

PE1-R5#
PE1-R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
PE1-R5(config)#
PE1-R5(config)#ip vrf
PE1-R5(config)#ip vrf CLIENTE-A
PE1-R5(config-vrf)#
PE1-R5(config-vrf)do sh ip vrf
      Name                               Default RD           Interfaces
      CLIENTE-A                         <not set>
PE1-R5(config-vrf)do sh run | sec vrf
ip vrf CLIENTE-A

```

É necessário, após definições anteriores, especificar o RD dentro da configuração desta VRF para que endereços privados e idênticos em clientes diferentes não sejam confundidos.

O comando *route target* possui algumas variações para compreender a troca de informações entre VRFs.

O complemento *export* define que rotas contidas nesta VRF serão exportadas com um RT de 15:39 (que faz menção à ligação de R1-R5 e R3-R9).

O complemento *import* faz o papel inverso, aprende as rotas de VRFs que são exportadas também com RT 15:39 (no exemplo abaixo, os exports de PE-R9).

O complemento *both* pode ser usado caso o desejado seja utilizar os dois complementos com os mesmos parâmetros.

```

PE1-R5(config-vrf)#
PE1-R5(config-vrf)#rd ?
    ASN:nn or IP-address:nn  VPN Route Distinguisher

PE1-R5(config-vrf)#rd 15:39
PE1-R5(config-vrf)#
PE1-R5(config-vrf)#route-target ?
    ASN:nn or IP-address:nn  Target VPN Extended Community
    both                      Both import and export Target-VPN community
    export                    Export Target-VPN community
    import                    Import Target-VPN community

PE1-R5(config-vrf)#route-target export 15:39
PE1-R5(config-vrf)#
PE1-R5(config-vrf)#route-target import 15:39
PE1-R5(config-vrf)#
PE1-R5(config-vrf)#
PE1-R5(config-vrf)#do sh run | sec vrf
ip vrf CLIENTE-A
    rd 15:39
    route-target export 15:39
    route-target import 15:39
PE1-R5(config-vrf)#

```

Após definição das VRFs, é necessário especificar quais interfaces fazem o encaminhamento de tráfego das mesmas.

A interface de R5 que conversa com R1 é a FastEthernet0/0, sendo esta a única com *forwarding* da VRF CLIENTE-A:

```

PE1-R5(config)#int f0/0
PE1-R5(config-if)#
PE1-R5(config-if)#no shut
PE1-R5(config-if)#

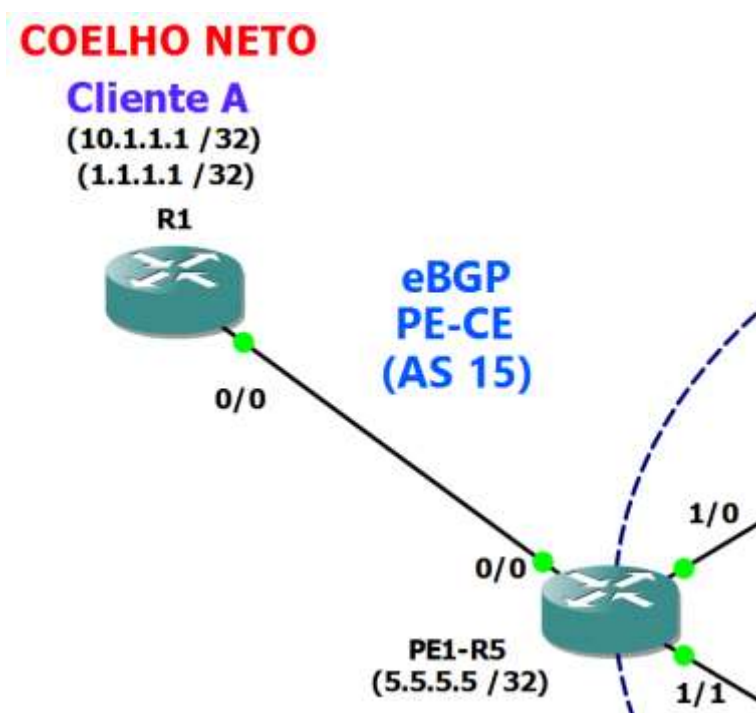
```

```

*Mar  1 00:27:38.087: %LINK-3-UPDOWN: Interface FastEthernet0/0,
changed state to up
*Mar  1 00:27:39.087: %LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet0/0, changed state
PE1-R5(config-if)#
PE1-R5(config-if)#ip vrf
PE1-R5(config-if)#ip vrf fo
PE1-R5(config-if)#ip vrf forwarding CLIENTE-A
PE1-R5(config-if)#
PE1-R5(config-if)#ip add
PE1-R5(config-if)#ip address 192.168.15.5 255.255.255.0
PE1-R5(config-if)#
PE1-R5(config-if)#no shut
PE1-R5(config-if)

```

Figura 3-11 - Configurações R5 e R1



No R1, neste meio tempo, são configurados o IP de *loopback* e da interface que conversa com PE-R5 - FastEthernet0/0 (figura 3-11).

Percebe-se que, no lado do cliente, as configurações de interface não precisam levar em consideração nenhum tipo de configuração de VRFs, ficando estas em total responsabilidade do provedor de serviços.

```

R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#
R1(config)#int lo0
R1(config-if)#
R1(config-if)#ip add 1.1.1.1 255.255.255.255
R1(config-if)#
R1(config-if)#int f0/0
R1(config-if)#
R1(config-if)#no shut
R1(config-if)#
R1(config-if)#ip address 192.168.15.1 255.255.255.0
R1(config-if)#

```

O próximo passo após configurações de VRFs é realizar a configuração do Multiprotocol iBGP que atuará entre R5 e R9.

O comando *router bgp 59* acessa o modo de configuração global do BGP para um AS específico, criando o mesmo caso ele não exista. No caso do experimento o AS do iBGP será 59.

Para que o roteador não aceite apenas endereços IPv4 *unicast*, o comando *no bgp default ipv4-unicast* é utilizado para preparar o roteador para ser multiprotocolo. Ele fará com que o roteador seja orientado à diferentes *address-families* com VPN IPv4.

Em seguida, foi utilizado o comando *bgp-log-neighbor-changes* para perceber mudanças de status dos pares BGP no ato na implementação do experimento e resolução de problemas ao decorrer do mesmo.

As configurações de vizinhanças feitas pelos comandos *neighbor 9.9.9.9 remote-as 59* e *neighbor 9.9.9.9 update-source lo0* estão definindo as relações multiprotocol BGP.

O 9.9.9.9, IP de *loopback* de R9, é usado como referência para permitir que ele reconheça e converse BGP com o roteador R5, contanto que seja configurado com o AS 59 - mesmo AS do R5.

O experimento só usa um AS no *backbone*, mas com isso percebe-se a possibilidade de criar engenharia de tráfego misturando ASs diferentes conforme a necessidade do provedor de serviços.

O comando *update-source* faz menção à interface *loopback* de R5, que é a interface que reconhecerá estas mudanças.

```
PE1-R5#conf t
Enter configuration commands, one per line. End with CNTL/Z.
PE1-R5(config)#router bgp 59
PE1-R5(config-router)#
PE1-R5(config-router)#no bgp default ipv4-unicast
PE1-R5(config-router)#
PE1-R5(config-router)#bgp log-neighbor-changes
PE1-R5(config-router)#
PE1-R5(config-router)#neighbor 9.9.9.9 remote-as 59
PE1-R5(config-router)#
PE1-R5(config-router)#neighbor 9.9.9.9 update-source lo0
PE1-R5(config-router)#
```

Nas configurações realizadas até agora foram criados ASs e especificações da vizinhança de R9, mas nenhuma configuração de IPv4 ou VPN IPv4 foi efetuada.

```
PE1-R5(config-router)#do sh run | sectionbgp
router bgp 59
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 9.9.9.9 remote-as 59
  neighbor 9.9.9.9 update-source Loopback0
```

Para entrar no modo de configuração de *address-families* e configurar uma sessão de roteamento usando explicitamente endereços com prefixos IPv4 padrão, é utilizado o comando *address-family ipv4 unicast*.

Com a ativação da vizinhança provida pelo comando *neighbor 9.9.9.9 activate*, o roteador R9 precisará somente reconhecer a vizinhança com o mesmo comando.

```
PE1-R5(config-router)#
PE1-R5(config-router)#address-family ipv4 unicast
PE1-R5(config-router-af)#
PE1-R5(config-router-af)#neighbor 9.9.9.9 activate
PE1-R5(config-router-af)#
```


A seguir, é definido o atributo estendido VPN IPv4 do *address-family* IPv4, o qual permite enviar e receber os atributos estendidos para MBGP. A ativação da vizinhança também é necessária neste caso.

O comando *neighbor 9.9.9.9 send-community both* define o envio tanto de *communities standard* e *extended* serão enviadas pelo MBGP.

Este roteador foi configurado com ambos (*both*) porque existe a necessidade de conversar sem atributos estendidos com R1 por parte de R5 e com R3 por parte de R9.

```
PE1-R5(config-router)#address-family vpnv4
PE1-R5(config-router-af)#
PE1-R5(config-router-af)#neighbor 9.9.9.9 activate
PE1-R5(config-router-af)#
PE1-R5(config-router-af)#neighbor 9.9.9.9 send-community both
PE1-R5(config-router-af)#
```

Como falado anteriormente, o Cliente A se utiliza de BGP para realizar seu roteamento dinâmico internamente, e até este ponto esta configuração não foi realizada.

Como o AS de R1 será aquele advertido à frente, é preciso que esta configuração seja realizada para que o *advertisement* da VRF no R5 possa ser feito.

De maneira análoga a como foi feito para o roteamento Multiprotocol iBGP, o AS 15 é configurado em R1 de modo a ter vizinhanças estabelecidas com o roteador R5, que conversa através do AS 59 com o R9.

```
R1#
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#
R1(config)#router bgp 15
R1(config-router)#
R1(config-router)#no bgp default ipv4-unicast
R1(config-router)#
R1(config-router)#neighbor 192.168.15.5 remote-as 59
R1(config-router)#
```

```

R1 (config-router) #address-family ipv4
R1 (config-router-af) #
R1 (config-router-af) #neighbor 192.168.15.5 activate
R1 (config-router-af) #

```

Existe, agora, a necessidade de entrelaçar a VRF que foi criada nos PEs ao BGP, porque é assim que ela que fará uma relação de *peer* com o outro PE. Esta VRF, então, precisa fazer o chamado *iBGP Peering*.

O comando *address-family ipv4 vrf CLIENTE-A* dentro da configuração global do BGP AS 59 define esta VRF como aquela que será associada aos consequentes comandos de configuração relacionados aos *address-families* seguintes.

O comando *neighbor 192.168.15.1 remote-as 15* define o AS remoto com o qual ela se relacionará, no caso o AS 15. Assim como nas definições anteriores, a ativação é necessária.

```

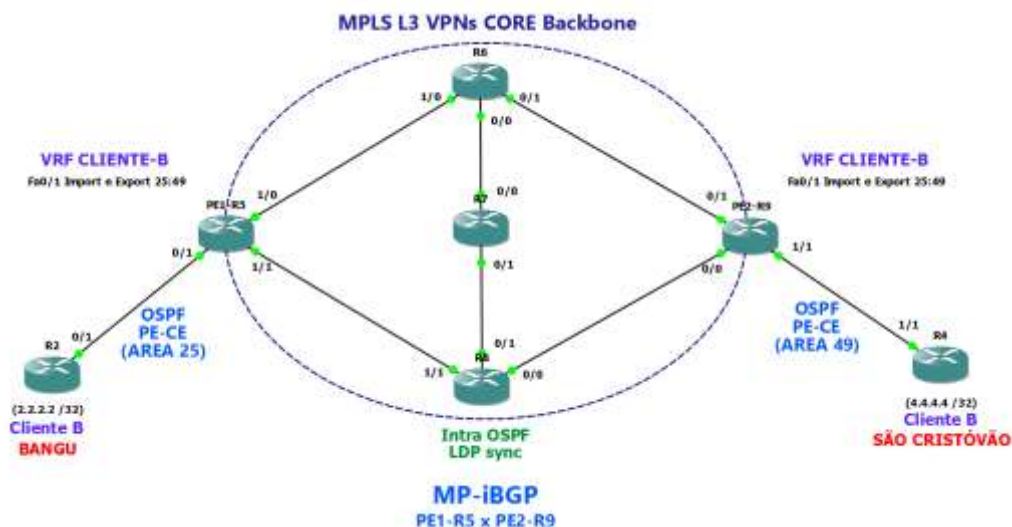
PE1-R5 (config-router) #
PE1-R5 (config-router) #address-family ipv4 vrf CLIENTE-A
PE1-R5 (config-router-af) #
PE1-R5 (config-router-af) #neighbor 192.168.15.1 remote-as 15
PE1-R5 (config-router-af) #
PE1-R5 (config-router-af) #neighbor 192.168.15.1 activate
PE1-R5 (config-router-af) #
PE1-R5 (config-router-af) #end

```

3.3.3 Configuração do roteamento OSPF CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-A

De uma maneira geral, a configuração inicial para VRFs do Cliente B (figura 3-13) no PE-R5 segue os mesmos passos que seguiu para a criação da VRF do Cliente A, análoga no PE-R9 (figura 3-14).

Figura 3-12. Topologia - MPLS CORE + Cliente B



No entanto, existem grandes diferenças na configuração que entrelaça o roteamento entre CE e PE para transmissão via MBGP entre os Pes (figura 3-12), conforme demonstrado a seguir.

Figura 3-13. Elementos envolvidos nas configurações para o Cliente B

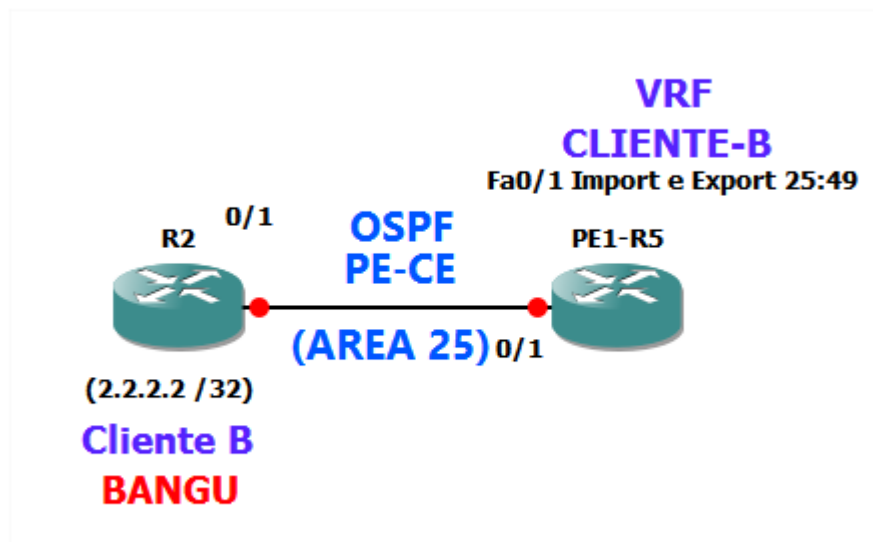
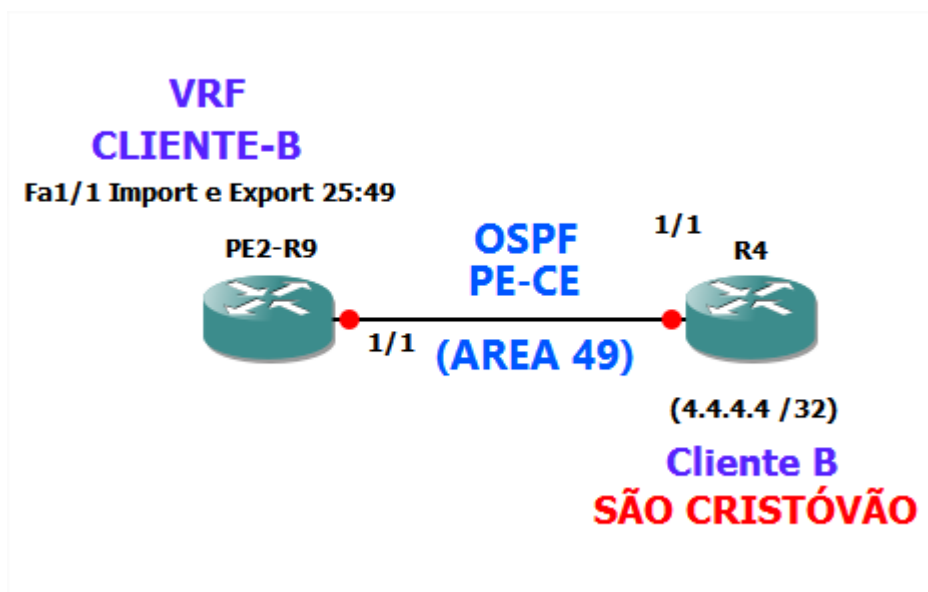


Figura 3-14. Elementos envolvidos nas configurações para o Cliente B



Até este ponto, não existem configurações no PE-R5 que sejam relacionadas ao roteamento do Cliente B, que nesta parte do estudo terá VRF configurada como CLIENTE-B:

```
PE1-R5#sh run | sec vrf
ip vrf CLIENTE-A
rd 15:39
route-target export 15:39
route-target import 15:39
ip vrf forwarding CLIENTE-A
address-family ipv4 vrf CLIENTE-A
neighbor 192.168.15.1 remote-as 15
neighbor 192.168.15.1 activate
no synchronization
```

Abaixo, a criação da VRF do Cliente B como CLIENTE-B e consequente criação de definição do seu *route distinguisher*, 25:49.

Logo após, o comando *route-target* é configurado com opção *both* fazendo menção à ambas as configurações de *import* e *export*.

O *export* define que rotas contidas nesta VRF serão exportadas com um RT de 25:49, e o *import* faz o papel inverso, de aprender as rotas de VRFs que são exportadas também com RD 25:49 (que, neste caso, serão do PE-R9).

```
PE1-R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
PE1-R5(config)#
PE1-R5(config)#ip vrf CLIENTE-B
PE1-R5(config-vrf)#
PE1-R5(config-vrf)#rd 25:49
PE1-R5(config-vrf)#
PE1-R5(config-vrf)#route-target both 25:49
PE1-R5(config-vrf)#
```

É visível, com o comando *show running-config | section vrf*, que a VRF CLIENTE-B foi definida globalmente no roteador, assim como a CLIENTE-A.

Faltam, no entanto, as configurações para encaminhamento das rotas aprendidas por esta VRF.

```
PE1-R5(config-vrf)#
PE1-R5(config-vrf)#do sh run | sec vrf
ip vrf CLIENTE-A
  rd 15:39
  route-target export 15:39
  route-target import 15:39
ip vrf CLIENTE-B
  rd 25:49
  route-target export 25:49
  route-target import 25:49
ip vrf forwarding CLIENTE-A
address-family ipv4 vrf CLIENTE-B
  no synchronization
address-family ipv4 vrf CLIENTE-A
  neighbor 192.168.15.1 remote-as 15
  neighbor 192.168.15.1 activate
  no synchronization
```

A seguir, especificamos quais interfaces fazem o encaminhamento de tráfego da VRF criada.

A interface de R5 que conversa com R2 (figura 3-11) é a FastEthernet0/1, sendo esta a única com *forwarding* da VRF CLIENTE-B. O endereçamento também é realizado neste momento.

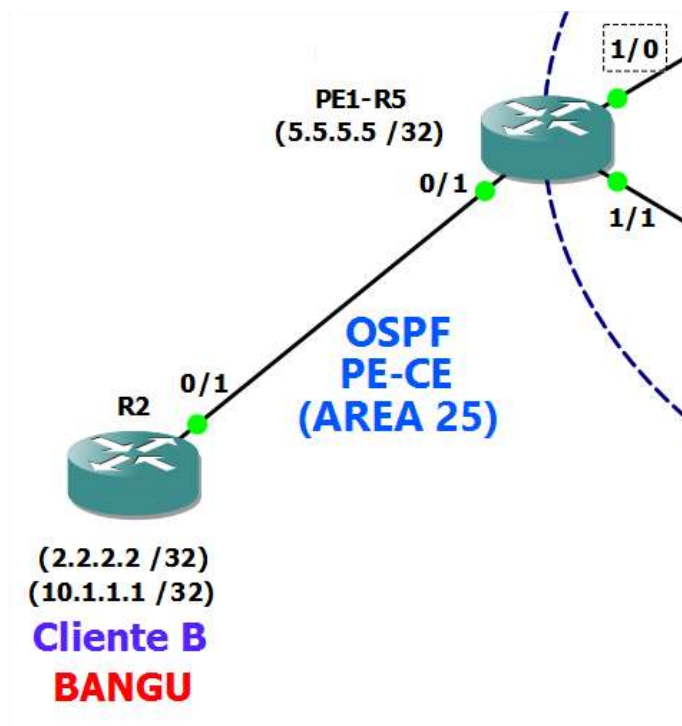
```

PE1-R5(config)#
PE1-R5(config)#int fastEthernet 0/1
PE1-R5(config-if)#
PE1-R5(config-if)#ip vrf forwarding CLIENTE-B
PE1-R5(config-if)#
PE1-R5(config-if)#ip add 192.168.25.5 255.255.255.0
PE1-R5(config-if)#
PE1-R5(config-if)#no shut
PE1-R5(config-if)#
*Mar  1 01:01:16.263: %LINK-3-UPDOWN: Interface FastEthernet0/1,
changed state to up
*Mar  1 01:01:17.263: %LINEPROTO-5-UPDOWN: Line protocol on
Interface FastEthernet0/1, changed state to up
PE1-R5(config-if)#
PE1-R5(config-if)#do sh run in Fa0/1
Building configuration...

Current configuration : 136 bytes
!
interface FastEthernet0/1
 ip vrf forwarding CLIENTE-B
 ip address 192.168.25.5 255.255.255.0
 shutdown
 duplex auto
 speed auto
end

```

Figura 3-15 - Configurações R5 e R2



Neste momento, a configurações iniciais de *loopback* em R2 são configuradas (figura 3-15):

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#int lo0
R2(config-if)#
R2(config-if)#ip address 2.2.2.2 255.255.255.255
R2(config-if)#
R2(config-if)#ip ospf 25 area 0
```

O endereçamento de interfaces de conversa com R5, incluindo configurações de OSPF, que atuarão como EGP via interface de saída FastEthernet0/1 neste roteador, pode ser realizado:

```
R2(config-if)#
R2(config-if)#int fa0/1
R2(config-if)#
```

```

R2(config-if)#no shut
R2(config-if)#
*Mar  1 00:28:50.363: %LINK-3-UPDOWN: Interface FastEthernet0/1,
changed state to up
*Mar  1 00:28:51.363: %LINEPROTO-5-UPDOWN: Line protocol on
Interface FastEthernet0/1, changed state to up
R2(config-if)#
R2(config-if)#ip add 192.168.25.2 255.255.255.0
R2(config-if

```

Assim como realizado para o Cliente A via BGP, é necessário criar um processo OSPF no PE que converse com o OSPF do CE do Cliente B.

No entanto, é necessário que ele seja mutuamente exclusivo para que atue somente na VRF CLIENTE-B, lembrando que o CORE MPLS já fala OSPF com PE-R5.

Para OSPF, isso é realizado com o mesmo comando *router ospf*, adicionando o *statement vrf* e definindo a VRF como CLIENTE-B.

O *process ID* será 25, sem prejuízo algum. E o *router ID* no PE-R5 não pode ser o IP de *loopback* 5.5.5.5, já usado como ID na conversa com o CORE, o que fez com que o ID fosse o IP da interface que conversa com R1 (192.168.25.5).

```

PE1-R5(config)#
PE1-R5(config)#router ospf 25 vrf CLIENTE-B
PE1-R5(config-router)#
PE1-R5(config-router)#router-id 192.168.25.5
PE1-R5(config-router)#
PE1-R5(config-router)#
PE1-R5(config-router)int fa0/1
PE1-R5(config-if)#
PE1-R5(config-if)#ip ospf 25 area 0

```

Até aqui, a conectividade entre R1 e R5 foi criada, mas a aprendizagem e redistribuição de rotas pela VRF CLIENTE-B ainda não acontece.

Para que isso seja possível, é necessário configurar a redistribuição via Multiprotocol BGP dentro desta configuração.

No caso do Cliente A, que usa BGP, ativar vizinhanças das rotas conectadas no lado do cliente com *router IDs* era o bastante, mas para o caso de OSPF é necessário redistribuir as rotas pelo AS do Multiprotocol iBGP.

Dentro da configuração do Processo OSPF 25:

```
PE1-R5(config)#
PE1-R5(config)#router ospf 25 vrf CLIENTE-B
```

São redistribuídas pelo AS 59 para o OSPF atuando como IGP.

Como configurado na primeira parte do experimento, o AS 59 é aquele do Multiprotocol BGP que atua entre o PE-R5 e o PE-R9.

O *statement metric 9* define o seu custo fixo, relativamente baixo pensando na tabela de distâncias administrativas demonstrada no subcapítulo 2.2.1.

```
PE1-R5(config-router)#
PE1-R5(config-router)#redistribute bgp 59 metric 9 subnets
PE1-R5(config-router)#
```

Após a redistribuição do BGP sobre OSPF, já realizada acima, é necessário também realizar a redistribuição do OSPF no BGP.

Do ponto de vista de BGP com relação à redistribuição de rotas, existem dois possíveis tipos de rotas OSPF: Internas, externas e *nssa-external*.

- Rotas *Internal* são aquelas internas ao AS específico
- Rotas *External* são aquelas externas ao AS específico

O experimento usa somente um AS na nuvem MBGP, então todos os tipos serão ativados, mas seria possível fazer engenharia de tráfego com estes diferentes tipos de rotas.

A relação a ser criada desta vez não é de vizinhança, mas sim de redistribuição, para finalizar o experimento.

```
PE1-R5(config-router)#router bgp 59
PE1-R5(config-router)#
PE1-R5(config-router)#address-family ipv4 vrf CLIENTE-B
PE1-R5(config-router-af)#
```

```
PE1-R5(config-router-af)#redistribute  ospf  25  match  external  
internal
```

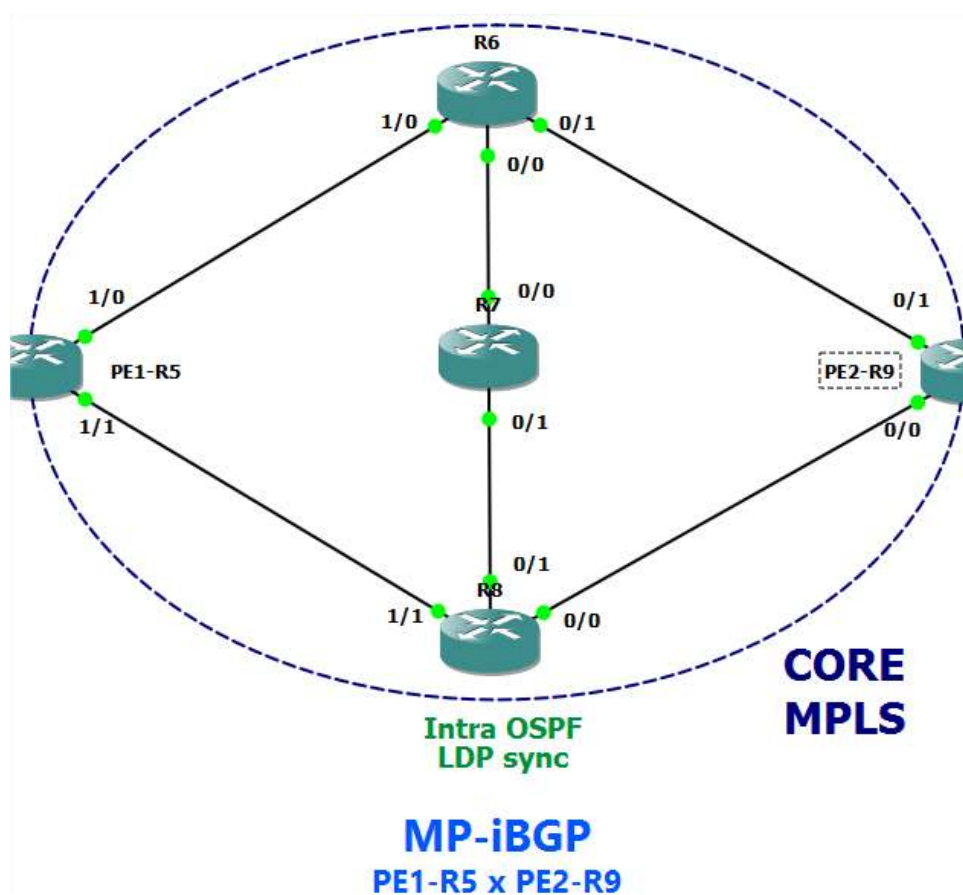
```
PE1-R5(config-router-af)#
```

4 Resultados e análises

4.1 Resultados do CORE MPLS no *Backbone* - Core OSPF e MPLS LDP, com configuração de todos os roteadores da nuvem MPLS

Testes iniciais realizados no R5 após configurar a interface 1/0, que conversa com o R6, e também a interface 1/0 do R6 (figura 4-1).

Figura 4-1. Core MPLS



O comando `show adjacency` vê detalhes das tabelas de adjacências CEF ou tabelas de adjacências de *Layer 3*, e as três colunas de resposta para o comando significam:

1. *Protocol*: o protocolo que é falado.
2. *Interface*: a interface de saída do roteador.
3. *Address*: o endereço do próximo salto.

No exemplo abaixo, na altura da execução do comando, percebe-se que somente as conexões entre R5 e R6 estão configuradas e, por isso, vemos TAGs LDP e IPs conversando somente para o R6, que possui o IP de final 6.

As Labels são nomeadas *TAGs* nas novas versões do ios CISCO:

```

PE1-R5#show adjacency
Protocol Interface Address
TAG      FastEthernet1/0 172.16.56.6(11)
IP       FastEthernet1/0 172.16.56.6(29)
PE1-R5#
PE1-R5#show adjacency detail
Protocol Interface Address
TAG      FastEthernet1/0 172.16.56.6(11)
                                0 packets, 0 bytes

C20C2BC0F100C20A2F0CF1008847
                                TFIB      03:46:21
                                Epoch: 0
                                172.16.56.6(29)
IP       FastEthernet1/0 0 packets, 0 bytes

C20C2BC0F100C20A2F0CF1000800
                                ARP      03:46:21
                                Epoch: 0

PE1-R5#

```

Após configuração da comunicação com o R8, por exemplo, a saída do comando é populada também com as informações da interface FastEthernet1/1:

```

PE1-R5#show adjacency detail
Protocol Interface Address
TAG      FastEthernet1/0 172.16.56.6(3)
                                0 packets, 0 bytes

C20C0D68F100C20A1F64F1008847
                                TFIB      04:02:28
                                Epoch: 0
                                172.16.56.6(5)
IP       FastEthernet1/0 0 packets, 0 bytes

C20C0D68F100C20A1F64F1000800

```

```

                                ARP          04:02:27
                                Epoch: 0
TAG          FastEthernet1/1    172.16.58.8(3)
                                0 packets, 0 bytes

C20E0544F101C20A1F64F1018847

                                TFIB          04:02:26
                                Epoch: 0
IP           FastEthernet1/1    172.16.58.8(5)
                                0 packets, 0 bytes

C20E0544F101C20A1F64F1010800

                                ARP          04:02:25
                                Epoch: 0

PE1-R5

```

O comando *debug mpls ldp messages* visa mostrar um fluxo específico de informações relacionadas com mensagens LDP enviadas de/para o *Router ID* do roteador onde o comando é executado.

Abaixo, são ativadas as visualizações de mensagens enviadas, recebidas e de eventos específicos (relacionados à mudanças de status):

```

PE1-R5#debug mpls ldp messages ?
    received  LDP received messages, excluding periodic Keep
Alive ("received
                all" to include periodic Keep Alive)
    sent      LDP sent messages, excluding periodic Keep
Alive ("sent all" to
                include periodic Keep Alive)

PE1-R5#
PE1-R5#debug mpls ldp messages sent
LDP sent PDUs, excluding periodic Keep Alive debugging is
on
PE1-R5#
PE1-R5#debug mpls ldp messages received
LDP received messages, excluding periodic Keep Alive
debugging is on
PE1-R5#
PE1-R5#debug mpls ldp transport events
LDP transport events debugging is on
PE1-R5#

```

É possível notar que, após configurar o MPLS em ambas as interfaces de R5 que falam com o CORE (FastEthernet1/0 e FastEthernet1/1) e ativar o *debugging*, o roteador está enviando HELLOs LDP.

Estes HELLOs são pacotes *multicast*, enviados para endereços 224.0.0.2 na porta 646 (UDP).

Isso significa que o R5 está procurando por quem esteja rodando LDP no mesmo segmento, para estabelecer sessões TCP e rodar LDP em conjunto:

```
PE1-R5#
*Mar  1 01:40:17.791: ldp: Send ldp hello; FastEthernet1/0,
src/dst 172.16.56.5/224.0.0.2, inst_id 0
*Mar  1 01:40:18.571: ldp: Send ldp hello; FastEthernet1/1,
src/dst 172.16.58.5/224.0.0.2, inst_id 0
PE1-R5#
PE1-R5#
*Mar  1 01:40:22.771: ldp: Send ldp hello; FastEthernet1/0,
src/dst 172.16.56.5/224.0.0.2, inst_id 0
*Mar  1 01:40:22.875: ldp: Send ldp hello; FastEthernet1/1,
src/dst 172.16.58.5/224.0.0.2, inst_id 0
PE1-R5#
PE1-R5#
*Mar  1 01:40:26.979: ldp: Send ldp hello; FastEthernet1/0,
src/dst 172.16.56.5/224.0.0.2, inst_id 0
*Mar  1 01:40:27.155: ldp: Send ldp hello; FastEthernet1/1,
src/dst 172.16.58.5/224.0.0.2, inst_id 0
```

Após configuração também no R6 e R8, percebem-se diversas mensagens sendo também recebidas (*Rcvd*) em R5.

```
PE1-R5#
*Mar  1 00:18:08.443: ldp: Rcvd ldp hello; FastEthernet1/0,
from 172.16.56.6 (6.6.6.6:0), intf_id 0, opt 0xC
*Mar  1 00:18:09.667: ldp: Rcvd ldp hello; FastEthernet1/1,
from 172.16.58.8 (8.8.8.8:0), intf_id 0, opt 0xC
*Mar  1 00:18:09.955: ldp: Send ldp hello; FastEthernet1/0,
src/dst 172.16.56.5/224.0.0.2, inst_id 0
*Mar  1 00:18:12.099: ldp: Send ldp hello; FastEthernet1/1,
src/dst 172.16.58.5/224.0.0.2, inst_id 0
PE1-R5#
```

```
*Mar  1 00:18:12.743: ldp: Rcvd ldp hello; FastEthernet1/0,
from 172.16.56.6 (6.6.6.6:0), intf_id 0, opt 0xC
*Mar  1 00:18:13.587: ldp: Rcvd ldp hello; FastEthernet1/1,
from 172.16.58.8 (8.8.8.8:0), intf_id 0, opt 0xC
*Mar  1 00:18:13.955: ldp: Send ldp hello; FastEthernet1/0,
src/dst 172.16.56.5/224.0.0.2, inst_id 0
*Mar  1 00:18:16.443: ldp: Send ldp hello; FastEthernet1/1,
src/dst 172.16.58.5/224.0.0.2, inst_id 0
```

O comando *show mpls ip bind* visa mostrar um fluxo específico de informações relacionadas às ligações aprendidas pelo LDP de forma sumarizada. Ele permite dizer os possíveis caminhos para alcançar MPLS IDs específicos dessa nuvem em que ele está inserido.

Os comandos abaixo demonstram a busca por possíveis saídas para o 9.9.9.9 partindo do R5.

O IP 9.9.9.9 é o prefixo de destino buscado (rede/máscara), que será referência para o resto dos atributos de saída do comando.

In Label é a label local designada pelo *Label Switch Router* e advertida para os outros LSRs.

Out Label é uma label remota aprendida por um vizinho LDP. Valores *imp-null* são os *nulls* implícitos, N/A.

O termo *inuse* indica que a *out label* está em uso para encaminhamento MPLS, ou seja, está instalada na *MPLS Forwarding Table* (LFIB):

```
PE1-R5#sh mpls ip bind 9.9.9.9 32
9.9.9.9/32
      in label:      506
      out label:     605      lsr: 6.6.6.6:0      inuse
      out label:     806      lsr: 8.8.8.8:0
PE1-R5#
```

A forma genérica, que demonstra todos os *binds* do roteador, segue abaixo, após configuração de todos os nós da nuvem MPLS:

```
PE1-R5#sh mpls ip bind
5.5.5.5/32
      in label:      imp-null
      out label:     804      lsr: 8.8.8.8:0
```

```

        out label:      604          lsr: 6.6.6.6:0
6.6.6.6/32
        in label:       504
        out label:      805          lsr: 8.8.8.8:0
        out label:      imp-null    lsr: 6.6.6.6:0          inuse
7.7.7.7/32
        in label:       505
        out label:      800          lsr: 8.8.8.8:0          inuse
        out label:      605          lsr: 6.6.6.6:0          inuse
8.8.8.8/32
        in label:       506
        out label:      imp-null    lsr: 8.8.8.8:0          inuse
        out label:      606          lsr: 6.6.6.6:0
9.9.9.9/32
        in label:       507
        out label:      806          lsr: 8.8.8.8:0          inuse
        out label:      600          lsr: 6.6.6.6:0          inuse

```

É possível, ainda, saber a rota exata de/para um par de IPs na nuvem confirmando a próxima interface e IPs do caminho com CEF. A partir do PE-R5, por exemplo:

```

PE1-R5#show ip cef exact-route 5.5.5.5 9.9.9.9
5.5.5.5          -> 9.9.9.9          : FastEthernet1/0 (next
hop 172.16.56.6)

```

Testes MPLS x OSPF também foram realizados para garantir o sincronismo de ambos. Na ausência de configuração OSPF na interface que conversa com R8, por exemplo, percebe-se falha na sincronização, conforme abaixo.

```

PE1-R5#sh mpls ldp igp sync
FastEthernet1/0:
  LDP configured; LDP-IGP Synchronization enabled.
  Sync status: sync achieved; peer reachable.
  Sync delay time: 0 seconds (0 seconds left)
  IGP holddown time: infinite.
  Peer LDP Ident: 6.6.6.6:0
  IGP enabled: OSPF 1
FastEthernet1/1:
  LDP configured; LDP-IGP Synchronization not enabled

```


Para evitar perda de pacotes por conta da falta de sincronização entre o LDP e o OSPF, após configuração do OSPF na interface faltante percebe-se que a interface que conversa com o R8 também possui sua sincronização sendo realizada:

```
PE1-R5#sh mpls ldp igp sync
FastEthernet1/0:
    LDP configured; LDP-IGP Synchronization enabled.
    Sync status: sync achieved; peer reachable.
    Sync delay time: 0 seconds (0 seconds left)
    IGP holddown time: infinite.
    Peer LDP Ident: 6.6.6.6:0
    IGP enabled: OSPF 1
FastEthernet1/1:
    LDP configured; LDP-IGP Synchronization enabled.
    Sync status: sync achieved; peer reachable.
    Sync delay time: 0 seconds (0 seconds left)
    IGP holddown time: infinite.
    Peer LDP Ident: 8.8.8.8:0
    IGP enabled: OSPF 1
```

Por fim, para ter visualização mais ampla quanto ao encaminhamento MPLS em toda esta nuvem configurada, usamos o comando *show mpls forwarding-table* para ver o conteúdo de toda a *Label Forwarding Information Base* (LFIB) do roteador.

Na LFIB, vemos os IPs de conversa entre os *peers* MPLS que não estão conectados diretamente ao R7, mas que já foram aprendidos por conta de todo o processo de descoberta demonstrado anteriormente.

A ideia é que assim, possam ser encontrados todos os possíveis destinos e caminhos para IPs da nuvem, partindo do R7, e comprovar a eficiência do *backbone*.

São pontos importantes no *display* do comando:

- *Local Tag*: *label* local designada pelo roteador
- *Outgoing Tag*: *label* designada pelo próximo salto para encaminhamento. A mensagem de *Pop Tag* significa que o próximo salto advertiu um *implicit null* para este destino. Sendo o penúltimo roteador antes do seu destino, as *labels* podem parar de ser utilizadas. Quando há um número de *label* neste campo, demonstra-se ali a *label* que será a mais externa no próximo salto (*Local Tag* no próximo) e assim sucessivamente até que os pacotes cheguem em seus destinos.

```
R7#show mpls forwarding-table
```


Percebe-se que a conectividade entre os nós, partindo de R5, testada via comandos de *ping*, a princípio não existe.

```
PE1-R5(config-if)#
PE1-R5(config-if)#do ping 192.168.15.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.15.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
PE1-R5(config-if)#
```

Isto acontece porque, uma vez que a interface de R5 que conversa com R1 foi configurada para encaminhar uma VRF específica, o caminho para esta rota não é aprendido pela tabela de roteamento global do roteador.

A conectividade não é corretamente testada se não for especificada a VRF específica a qual o endereço IP fala.

Podemos ver abaixo que não há menção ao endereço na tabela de roteamento global do roteador R5:

```
PE1-R5(config-if)#do sh ip route

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

5.0.0.0/32 is subnetted, 1 subnets
C      5.5.5.5 is directly connected, Loopback0
6.0.0.0/32 is subnetted, 1 subnets
O      6.6.6.6 [110/2] via 172.16.56.6, 01:12:48, FastEthernet1/0
```

```

172.16.0.0/24 is subnetted, 6 subnets
C      172.16.56.0 is directly connected, FastEthernet1/0
C      172.16.58.0 is directly connected, FastEthernet1/1
O      172.16.89.0 [110/11] via 172.16.58.8, 01:12:48, FastEthernet1/1
O      172.16.78.0 [110/11] via 172.16.58.8, 01:12:48, FastEthernet1/1
O      172.16.69.0 [110/11] via 172.16.56.6, 01:12:49, FastEthernet1/0
O      172.16.67.0 [110/11] via 172.16.56.6, 01:12:49, FastEthernet1/0
7.0.0.0/32 is subnetted, 1 subnets
O      7.7.7.7 [110/12] via 172.16.58.8, 01:12:49, FastEthernet1/1
        [110/12] via 172.16.56.6, 01:12:49, FastEthernet1/0
8.0.0.0/32 is subnetted, 1 subnets
O      8.8.8.8 [110/2] via 172.16.58.8, 01:12:51, FastEthernet1/1
9.0.0.0/32 is subnetted, 1 subnets
O      9.9.9.9 [110/12] via 172.16.58.8, 01:12:52, FastEthernet1/1
        [110/12] via 172.16.56.6, 01:12:52, FastEthernet1/0

```

A rota, no entanto, é percebida entre os caminhos conhecidos da VRF, conforme abaixo.

No exemplo, as rotas em R3 também já foram configuradas e temos os IPs da rede 192.168.39.0/24 sendo aprendidos via BGP, a ser demonstrado mais à frente:

```

PE1-R5#sh ip route vrf CLIENTE-A

Routing Table: CLIENTE-A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static
route
        o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

1.0.0.0/32 is subnetted, 1 subnets
B      1.1.1.1 [20/0] via 192.168.15.1, 00:00:40
3.0.0.0/32 is subnetted, 1 subnets

```

```

B      3.3.3.3 [200/0] via 9.9.9.9, 00:00:27
C      192.168.15.0/24 is directly connected, FastEthernet0/0
B      192.168.39.0/24 [200/0] via 9.9.9.9, 00:00:27

```

Para correto teste de conectividade, é necessário especificar a VRF para testes usando *ping*, que funciona conforme exemplificado abaixo:

```

PE1-R5#
R1(config)#ping 192.168.15.1
*Mar  1 01:22:58.631: %SYS-5-CONFIG_I: Configured from console by
console
PE1-R5#ping vrf CLIENTE-A 192.168.15.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.15.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/19/24
ms
PE1-R5#

```

No lado do Cliente A, como já foi demonstrado, não existe configuração de VRFs, então não há necessidade de especificações adicionais.

```

R1(config)#ping 192.168.15.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.15.5, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/19/24
ms
R1(config)#

```

Após configurações globais de VRFs e Multiprotocol BGP nos PEs, foram verificadas as definições por inteiro antes de alocar e ativar instâncias IPv4 nas VRFs criadas anteriormente:

```

PE1-R5(config-router-af)#do show running-config | sec bgp

```

Configuração Global - Estabelecimento do iBGP com o outro PEs:

```
router bgp 59
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 9.9.9.9 remote-as 59
  neighbor 9.9.9.9 update-source Loopback0
  !
```

Ativação desta relação de *peering* com um *address-family* IPv4

```
address-family ipv4
  neighbor 9.9.9.9 activate
  no auto-summary
  no synchronization
  exit-address-family
  !
```

Habilitação do atributo estendido do *address-family* VPN IPv4:

```
address-family vpnv4
  neighbor 9.9.9.9 activate
  neighbor 9.9.9.9 send-community both
  exit-address-family
  !
address-family ipv4 vrf CLIENTE-A
  no synchronization
  exit-address-family
PE1-R5(config-router-af) #
```

Tentativas de criação de encaminhamento com VRFs ainda não existentes não devem ser possíveis. A tentativa de criação com a VRF do Cliente B (ainda não criada) não é possível porque ela não existe.

```
PE1-R5(config-router-af) #
PE1-R5(config-router-af) #address-family ipv4 vrf CLIENTE-B
% VRF CLIENTE-B does not exist or does not have a RD
```

Quando na espera da configuração do roteamento BGP de R1 para conversa com R5, e espelhamento das configurações de R5 em R9 já concluídos, são realizadas ativações de debugs para identificar as mudanças de estado e envio ou recebimento de *keepalives* BGP no R5:

```
PE1-R5#
PE1-R5#debug bgp vpnv4 unicast keepalives
BGP keepalives debugging is on for address family: VPNv4 Unicast
PE1-R5#
PE1-R5#debug bgp vpnv4 unicast updates
BGP updates debugging is on for address family: VPNv4 Unicast
```

Com os *debugs* já ativados, é possível o envio e recebimento de *keepalives* de R9 (9.9.9.9) e a formação de adjacência com ele (*ADJCHANGE*):

```
PE1-R5#
*Mar  1 03:35:54.771: BGP: 9.9.9.9 sending KEEPALIVE (io)
*Mar  1 03:35:54.779: BGP: 9.9.9.9 received KEEPALIVE, length (excl.
header) 0
PE1-R5#
*Mar  1 03:36:00.883: %BGP-5-ADJCHANGE: neighbor 9.9.9.9 Down Peer
closed the session
PE1-R5#
*Mar  1 03:36:02.927: BGP: 9.9.9.9 sending KEEPALIVE (rcv_open)
*Mar  1 03:36:02.939: BGP: 9.9.9.9 received KEEPALIVE, length (excl.
header) 0
PE1-R5#
*Mar  1 03:36:02.939: %BGP-5-ADJCHANGE: neighbor 9.9.9.9 Up
```

Ao realizar as configurações devidas em R1, foi possível perceber também os *keepalives* e mudanças de adjacências de R1 (192.168.15.1):

```
PE1-R5#
*Mar  1 03:54:35.315: BGP: 192.168.15.1 sending KEEPALIVE (rcv_open)
*Mar  1 03:54:35.315: BGP: 192.168.15.1 received KEEPALIVE, length
(excl. header) 0
*Mar  1 03:54:35.315: %BGP-5-ADJCHANGE: neighbor 192.168.15.1 vpn vrf
CLIENTE-A Up
```

```
*Mar  1 03:54:35.315: BGP: 192.168.15.1 sending KEEPALIVE (updgrp)
*Mar  1 03:54:35.315: BGP: 192.168.15.1 sending KEEPALIVE (io)
```

Também é possível perceber a instalação das rotas da VRF CLIENTE-A sendo escritas em sua tabela de roteamento no R5 (1.1.1.1 salvo em CLIENTE-A *table*):

```
*Mar  1 03:54:35.363: BGP(2): Revise route installing 1 of 1 routes for
1.1.1.1/32 -> 192.168.15.1(CLIENTE-A) to CLIENTE-A IP table
*Mar  1 03:54:35.363: BGP(2): Revise route installing 1 of 1 routes for
192.168.15.0/24 -> 192.168.15.1(CLIENTE-A) to CLIENTE-A IP table
```

Podemos perceber os *updates* VPN IPv4 que mostram os atributos estendidos.

O Route Distinguisher 15:39 demonstrado em 15:39:1.1.1.1/32 faz com que a rota para 1.1.1.1 seja única, com próximo salto definido como 5.5.5.5, *label* 508 no momento da escrita na LFIB e *path* do AS de origem, 15:

```
*Mar      1  03:54:35.367:  BGP(2):  9.9.9.9  send  UPDATE  (format)
15:39:192.168.15.0/24, next 5.5.5.5, metric 0, path 15, extended community
RT:15:39
*Mar  1 03:54:35.371: BGP(2): 9.9.9.9 send UPDATE (prepend, chgflags:
0x820) 15:39:1.1.1.1/32, next 5.5.5.5, metric 0, path 15, extended community
RT:15:39
```

O antes e depois das configurações das rotas em R9 e R3, são facilmente percebidos nas tabelas das VRFs:

```
PE1-R5#
PE1-R5#sh ip route vrf CLIENTE-A
```

```
Routing Table: CLIENTE-A
```

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
```

```
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
E1 - OSPF external type 1, E2 - OSPF external type 2
```

```
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
```

```
level-2
```



```

        ia - IS-IS inter area, * - candidate default, U - per-user static
route
        o - ODR, P - periodic downloaded static route

```

Gateway of last resort is not set

```

        1.0.0.0/32 is subnetted, 1 subnets
B       1.1.1.1 [20/0] via 192.168.15.1, 00:37:12
C       192.168.15.0/24 is directly connected, FastEthernet0/0
PE1-R5#

```

-

```
PE1-R5#sh ip route vrf CLIENTE-A
```

Routing Table: CLIENTE-A

```

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS

```

level-2

```

        ia - IS-IS inter area, * - candidate default, U - per-user static
route
        o - ODR, P - periodic downloaded static route

```

Gateway of last resort is not set

```

        1.0.0.0/32 is subnetted, 1 subnets
B       1.1.1.1 [20/0] via 192.168.15.1, 00:00:40
        3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [200/0] via 9.9.9.9, 00:00:27
C       192.168.15.0/24 is directly connected, FastEthernet0/0
B       192.168.39.0/24 [200/0] via 9.9.9.9, 00:00:27
PE1-R5#

```

Também é possível confirmar de forma sumarizada todos os próximos saltos relacionados a VPN IPv4 no experimento até o momento:

```

PE1-R5#sh bgp vpnv4 unicast all
BGP table version is 23, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
                r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

```

Network	Next Hop	Metric	LocPrf	Weight	Path
Route Distinguisher: 15:39 (default for vrf CLIENTE-A)					
*> 1.1.1.1/32	192.168.15.1	0		0	15 ?
*>i3.3.3.3/32	9.9.9.9	0	100	0	39 ?
r> 192.168.15.0	192.168.15.1	0		0	15 ?
*>i192.168.39.0	9.9.9.9	0	100	0	39 ?

E, por fim, assim como o comando *show mpls forwarding* usado na primeira parte do experimento mostrava os saltos seguintes com estudo de labels LDP, a verificação final de conectividade fim a fim entre os roteadores do Cliente A também é possível com testes simples de pings contínuos:

```

R1#ping 3.3.3.3 repeat 50

Type escape sequence to abort.
Sending 50, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (50/50), round-trip min/avg/max = 20/50/88
ms
R1#

```

O comando *traceroute* mostra os nós por onde o tráfego está passando, englobando inclusive as labels LDPs incluídas no meio do encaminhamento.

Partindo do R1, por exemplo, o tráfego ICMP está passando pelo R5, R8 e R9 antes de chegar ao seu destino R3. Os IPs de interface de entrada nos roteadores são respectivamente 192.168.15.5, 172.16.58.8, 192.168.39.9 e 192.168.39.3.

Entre os nós 1 e 2, houve inserção de *labels* LDP. Entre os nós 2 e 3, ocorreu o *pop* da *label* 805, que deu lugar para a 905. O roteador R9, por sua vez, apontava

para o R3 com *implicit-null*, acabando então com o fluxo MPLS e entregando o tráfego sem *labels* no passo 4:

```
R1#traceroute 3.3.3.3

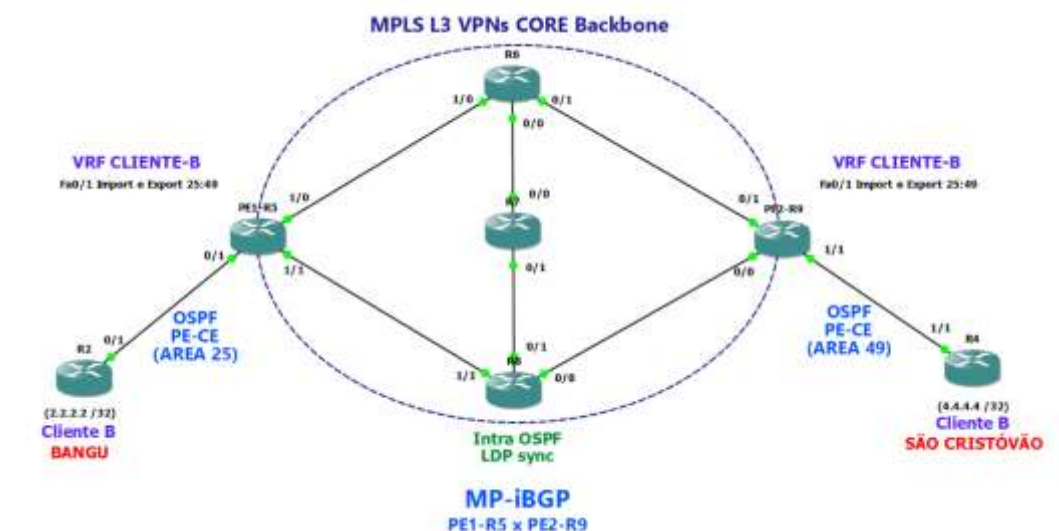
Type escape sequence to abort.
Tracing the route to 3.3.3.3

 0 192.168.15.5 8 msec 36 msec 20 msec
 1 172.16.58.8 [MPLS: Labels 805/908 Exp 0] 60 msec 80 msec 64 msec
 2 192.168.39.9 [AS 39] [MPLS: Label 908 Exp 0] 44 msec 60 msec 36
msec
 3 192.168.39.3 [AS 39] 60 msec 68 msec 68 msec
R1#
```

4.3 Resultados de roteamento OSPF CE-PE, VRFs e MP-iBGP nos PEs para conectividade fim-a-fim do CLIENTE-B

Antes da configuração de adjacências OSPF em PE-R5, R2 não conseguiria conectividade com ele, ainda que ambos tivessem seus IPs configurados nas respectivas interfaces (figura 4-3).

Figura 4-3. Core MPLS e PEs do Cliente B



```
R2(config-if)#do ping 192.168.25.5
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.25.5, timeout is 2
seconds:
```

```
.....
Success rate is 0 percent (0/5)
R2(config-if)#
R2(config-if)#
```

Ativamos a configuração OSPF em PE-R5:

```
PE1-R5(config-router)#do sh run | sec ospf
ip ospf 1 area 0
ip ospf 1 area 0
ip ospf 1 area 0
router ospf 25 vrf CLIENTE-B
router-id 192.168.25.5
ispf
log-adjacency-changes
router ospf 1
mpls ldp sync
router-id 5.5.5.5
log-adjacency-changes
PE1-R5(config-router)#
```

As adjacências se formam normalmente:

```
PE1-R5(config-if)#
*Mar  1 01:29:26.451: %OSPF-5-ADJCHG: Process 25, Nbr 2.2.2.2
on FastEthernet0/1 from LOADING to FULL, Loading Done
```

```
R2#show ip ospf neigh
```

	Neighbor ID	Pri	State	Dead Time	Address
Interface					
	192.168.25.5	1	2WAY/DROTHER	00:00:31	192.168.25.5
FastEthernet0/1					

E ainda que R2 ainda não aprendesse rotas por ausência da redistribuição de tráfego por parte do PE-R5 e falta de configuração das mesmas em R4 e R9:

```
R2#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF
inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-
user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

2.0.0.0/32 is subnetted, 1 subnets
C      2.2.2.2 is directly connected, Loopback0
C      192.168.25.0/24 is directly connected, FastEthernet0/1
```

A conectividade pode ser testada:

```
R2(config-if)#
R2(config-if)#do ping 192.168.25.5
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.25.5, timeout is 2
seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max =
20/22/24 ms
R2(config-if)#
```

Podemos verificar que R4 aprende as rotas em R2 após configurações de *redistribute* no roteador PE-R5 quando o entrelace de OSPF e iBGP é configurado.

```
R4#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter
area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-
user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    2.0.0.0/32 is subnetted, 1 subnets
      O   E2           2.2.2.2   [110/9]   via   192.168.49.9,   00:01:13,
FastEthernet1/1
      O   E2  192.168.25.0/24   [110/9]   via   192.168.49.9,   00:01:13,
FastEthernet1/1
    4.0.0.0/32 is subnetted, 1 subnets
      C           4.4.4.4 is directly connected, Loopback0
      C   192.168.49.0/24 is directly connected, FastEthernet1/1
R4#
```

Mesmo que a rota seja aprendida em R4, ainda não é possível fazer um *traceroute* ou testes de *ping* no elemento aprendido, conforme abaixo:

```
R4#traceroute 2.2.2.2

Type escape sequence to abort.
Tracing the route to 2.2.2.2
```

```

1 192.168.49.9 16 msec 20 msec 24 msec
2 * * *
3 * * *
4 * * *
5 * * *
6 * * *
7 * * *
8 * * *
9 * *

```

Isto acontece porque no PE-R9, até este momento, não foram adicionados IPv4 VRF *statements* para o Cliente B. São eles que permitem redistribuir as rotas via *redistribute* para outros roteadores:

```

PE2-R9(config-router)#do sh run | sec bgp
  redistribute bgp 59 metric 9 subnets
router bgp 59
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 5.5.5.5 remote-as 59
  neighbor 5.5.5.5 update-source Loopback0
  !
  address-family ipv4
    neighbor 5.5.5.5 activate
    no auto-summary
    no synchronization
  exit-address-family
  !
  address-family vpnv4
    neighbor 5.5.5.5 activate
    neighbor 5.5.5.5 send-community both
  exit-address-family
  !
  address-family ipv4 vrf CLIENTE-B
    no synchronization

```

```

exit-address-family
!
address-family ipv4 vrf CLIENTE-A
  neighbor 192.168.39.3 remote-as 39
  neighbor 192.168.39.3 activate
  no synchronization
exit-address-family

```

Sem *redistribute* no PE-R9, o OSPF não será replicado no MBGP, então R2 não percebe essas informações. Por isso, não existe menção a IPs 4.4.4.4 em R2:

```

R2#
R2#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter
area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-
user static route
        o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    2.0.0.0/32 is subnetted, 1 subnets
C      2.2.2.2 is directly connected, Loopback0
C    192.168.25.0/24 is directly connected, FastEthernet0/1
R2#

```

No R4, no entanto, percebemos a rota de R2 (2.2.2.2):

```

R4#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP

```


D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

ia - IS-IS inter area, * - candidate default, U - per-user static route

o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

2.0.0.0/32 is subnetted, 1 subnets

O E2 2.2.2.2 [110/9] via 192.168.49.9, 01:28:45, FastEthernet1/1

O E2 192.168.25.0/24 [110/9] via 192.168.49.9, 01:28:45, FastEthernet1/1

4.0.0.0/32 is subnetted, 1 subnets

C 4.4.4.4 is directly connected, Loopback0

C 192.168.49.0/24 is directly connected, FastEthernet1/1

Isto acontece em R4, diferentemente de R2, porque no PE-R5 o *redistribute* OSPF foi configurado:

```
PE1-R5#sh run | sec bgp
  redistribute bgp 59 metric 9 subnets
router bgp 59
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 9.9.9.9 remote-as 59
  neighbor 9.9.9.9 update-source Loopback0
  !
  address-family ipv4
    neighbor 9.9.9.9 activate
  no auto-summary
```

```

    no synchronization
exit-address-family
!
address-family vpnv4
    neighbor 9.9.9.9 activate
    neighbor 9.9.9.9 send-community both
exit-address-family
!
address-family ipv4 vrf CLIENTE-B
redistribute ospf 25 vrf CLIENTE-B match internal external 1
external 2 nssa-external 1 nssa-external 2
    no synchronization
exit-address-family
!
address-family ipv4 vrf CLIENTE-A
    neighbor 192.168.15.1 remote-as 15
    neighbor 192.168.15.1 activate
    no synchronization
exit-address-family

```

Configurações finais:

```

PE1-R5(config-router-af)#do sh run | sec bgp
    redistribute bgp 59 metric 9 subnets
router bgp 59
    no bgp default ipv4-unicast
    bgp log-neighbor-changes
    neighbor 9.9.9.9 remote-as 59
    neighbor 9.9.9.9 update-source Loopback0
!
address-family ipv4
    neighbor 9.9.9.9 activate
    no auto-summary
    no synchronization
exit-address-family
!
address-family vpnv4

```

```

    neighbor 9.9.9.9 activate
    neighbor 9.9.9.9 send-community both
exit-address-family
!
address-family ipv4 vrf CLIENTE-B
    redistribute ospf 25 vrf CLIENTE-B match internal external 1
external 2 nssa-external 1 nssa-external 2
    no synchronization
exit-address-family
!
address-family ipv4 vrf CLIENTE-A
    neighbor 192.168.15.1 remote-as 15
    neighbor 192.168.15.1 activate
    no synchronization
exit-address-family
PE1-R5(config-router-af)#

```

Neste ponto vemos que, com a VRF já acertada, a conectividade se dá com sucesso.

É feito, por fim, um teste final de conectividade por *ping* entre roteadores de clientes, e na sequência, *traceroute* para entendimento do caminho escolhido pelos pacotes.

```

R4#ping 2.2.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max =
44/58/84 ms
R4#
R4#trace 2.2.2.2

Type escape sequence to abort.
Tracing the route to 2.2.2.2

```

```

1 192.168.49.9 24 msec 8 msec 12 msec
2 172.16.69.6 [MPLS: Labels 604/511 Exp 0] 32 msec 40 msec 36
msec
3 192.168.25.5 [MPLS: Label 511 Exp 0] 40 msec 20 msec 40 msec
4 192.168.25.2 40 msec 40 msec 68 msec
R4#

```

4.4 Testes adicionais

Para provar o sucesso do experimento do ponto de vista de divisão de tráfego, foram criadas interfaces *loopback* com IPs 10.1.1.1 nos CEs R1 e R2, assim como IPs 10.10.10.10 nos CEs R3 e R4.

O intuito deste teste é comprovar o conceito de divisão final de tráfego com testes de conectividade adicionais, mostrando o caminho correto entre os CEs ainda que os endereços sejam os mesmos.

```

R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#
R1(config)#int lo1
R1(config-if)#
R1(config-if)#ip add
R1(config-if)#ip add 10.1.1.1 255.255.255.255

R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#
R2(config)#
R2(config)#int lo1
R2(config-if)#
R2(config-if)#ip add 10.1.1.1 255.255.255.255
R2(config-if)#
R2(config-if)#ip ospf 25 area 0

```

4.4.1 Testes de separação de tabelas de VRFs

Percebe-se facilmente que IPs 10.x.x.x se repetem em ambas, ainda que nunca conversem entre si, no entanto os IPs das primeiras *loopbacks* criadas, não.

A tabela do Cliente A possui as *loopbacks* de R1 e R3, enquanto a do Cliente B possui somente aquelas de R2 e R4, conforme esperado.

No entanto, ambas continuam possuindo as segundas *loopbacks* - 10.1.1.1 para R1 e R2 , e 10.10.10.10 para R3 e R4.

```

PE1-R5#sh ip route vrf CLIENTE-A

Routing Table: CLIENTE-A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter
area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-
user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    1.0.0.0/32 is subnetted, 1 subnets
B       1.1.1.1 [20/0] via 192.168.15.1, 00:21:17
    3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [200/0] via 9.9.9.9, 00:21:15
C     192.168.15.0/24 is directly connected, FastEthernet0/0
B     192.168.39.0/24 [200/0] via 9.9.9.9, 00:21:15
    10.0.0.0/32 is subnetted, 2 subnets
B       10.10.10.10 [200/0] via 9.9.9.9, 00:21:15
B       10.1.1.1 [20/0] via 192.168.15.1, 00:21:17

```

-

```
PE1-R5#sh ip route vrf CLIENTE-B
```

```
Routing Table: CLIENTE-B
```

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
```

```
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter
area
```

```
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external
type 2
```

```
        E1 - OSPF external type 1, E2 - OSPF external type 2
```

```
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 -
IS-IS level-2
```

```
        ia - IS-IS inter area, * - candidate default, U - per-
user static route
```

```
        o - ODR, P - periodic downloaded static route
```

```
Gateway of last resort is not set
```

```
        2.0.0.0/32 is subnetted, 1 subnets
```

```
        O          2.2.2.2 [110/11] via 192.168.25.2, 00:21:39,
FastEthernet0/1
```

```
        C    192.168.25.0/24 is directly connected, FastEthernet0/1
```

```
        4.0.0.0/32 is subnetted, 1 subnets
```

```
        B          4.4.4.4 [200/2] via 9.9.9.9, 00:21:23
```

```
        10.0.0.0/32 is subnetted, 2 subnets
```

```
        B          10.10.10.10 [200/2] via 9.9.9.9, 00:21:23
```

```
        O          10.1.1.1 [110/11] via 192.168.25.2, 00:21:39,
FastEthernet0/1
```

```
        B    192.168.49.0/24 [200/0] via 9.9.9.9, 00:21:23
```

4.4.2 Testes de peers Multiprotocol BGP

Visualizando a globalmente os *peers* estabelecidos nas relações de VPN IPv4, percebemos os dois clientes completamente separados, ainda que com IPs privados idênticos, conforme configurado anteriormente.

```

PE1-R5#sh bgp vpnv4 unicast all
BGP table version is 23, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best,
i - internal,
                r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete


      Network          Next Hop           Metric LocPrf Weight Path
Route Distinguisher: 15:39 (default for vrf CLIENTE-A)
*> 1.1.1.1/32         192.168.15.1             0                   0 15
?
*>i3.3.3.3/32         9.9.9.9                  0        100       0 39
?
*> 10.1.1.1/32        192.168.15.1            0                   0 15
?
*>i10.10.10.10/32     9.9.9.9                 0        100       0 39
?
r> 192.168.15.0       192.168.15.1            0                   0 15
?
*>i192.168.39.0       9.9.9.9                 0        100       0 39
?

Route Distinguisher: 25:49 (default for vrf CLIENTE-B)
*> 2.2.2.2/32         192.168.25.2            11               32768 ?
*>i4.4.4.4/32         9.9.9.9                 2        100         0 ?
*> 10.1.1.1/32        192.168.25.2            11               32768 ?
*>i10.10.10.10/32     9.9.9.9                 2        100         0 ?
*> 192.168.25.0       0.0.0.0                 0               32768 ?
*>i192.168.49.0       9.9.9.9                 0        100         0 ?
PE1-R5#
PE1-R5#
```

Estudando as VRFs do ponto de vista do BGP que atua entre R1 e PE-R5 (Cliente A), percebe-se que o *Extended Community* necessário para o BGP está bem definido como o RT, que neste caso é o 15:39:

```
PE1-R5#
PE1-R5#sh bgp vpnv4 unicast vrf CLIENTE-A 10.1.1.1
BGP routing table entry for 15:39:10.1.1.1/32, version 3
Paths: (1 available, best #1, table CLIENTE-A)
  Advertised to update-groups:
    2
  15
    192.168.15.1 from 192.168.15.1 (10.1.1.1)
      Origin incomplete, metric 0, localpref 100, valid,
external, best
      Extended Community: RT:15:39
      mpls labels in/out 509/nolabel
PE1-R5#
```

Do ponto de vista do OSPF, atuante entre R2 e PE-R5 (Cliente B), o RT também está sendo definido na sessão de *Extended Community*.

```
PE1-R5#sh bgp vpnv4 unicast vrf CLIENTE-B 10.1.1.1
BGP routing table entry for 25:49:10.1.1.1/32, version 20
Paths: (1 available, best #1, table CLIENTE-B)
  Advertised to update-groups:
    2
  Local
    192.168.25.2 from 0.0.0.0 (5.5.5.5)
      Origin incomplete, metric 11, localpref 100, weight 32768,
valid, sourced, best
      Extended Community: RT:25:49 OSPF DOMAIN
ID:0x0005:0x000000190200
      OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:192.168.25.5:0
      mpls labels in/out 512/nolabel
PE1-R5#
```


Na sequência, testamos as conectividades de IPs idênticos em diferentes clientes, que devem funcionar perfeitamente.

Em R4, a conectividade com R2 funciona:

```
R4#trace 2.2.2.2
```

```
Type escape sequence to abort.
```

```
Tracing the route to 2.2.2.2
```

```

  1 192.168.49.9 24 msec 8 msec 12 msec
  2 172.16.69.6 [MPLS: Labels 604/511 Exp 0] 32 msec 40 msec 36
msec
  3 192.168.25.5 [MPLS: Label 511 Exp 0] 40 msec 20 msec 40 msec
  4 192.168.25.2 40 msec 40 msec 68 msec
```

```
R4#ping 10.1.1.1
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max =
44/58/84 ms
```

```
R4#trace 10.1.1.1
```

```
Type escape sequence to abort.
```

```
Tracing the route to 10.1.1.1
```

```

  1 192.168.49.9 16 msec 20 msec 24 msec
  2 172.16.69.6 [MPLS: Labels 604/512 Exp 0] 88 msec 60 msec 52
msec
  3 192.168.25.5 [MPLS: Label 512 Exp 0] 60 msec 28 msec 40 msec
  4 192.168.25.2 20 msec 56 msec 48 msec
```

A ausência de conectividade entre os diferentes clientes também pode ser visualizada com testes de *ping*:

```
R1#ping 2.2.2.2
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
```

```
.....
```

```
Success rate is 0 percent (0/5)
```

```
R1#ping 4.4.4.4
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 4.4.4.4, timeout is 2 seconds:
```

```
.....
```

```
Success rate is 0 percent (0/5)
```

```
R1#
```

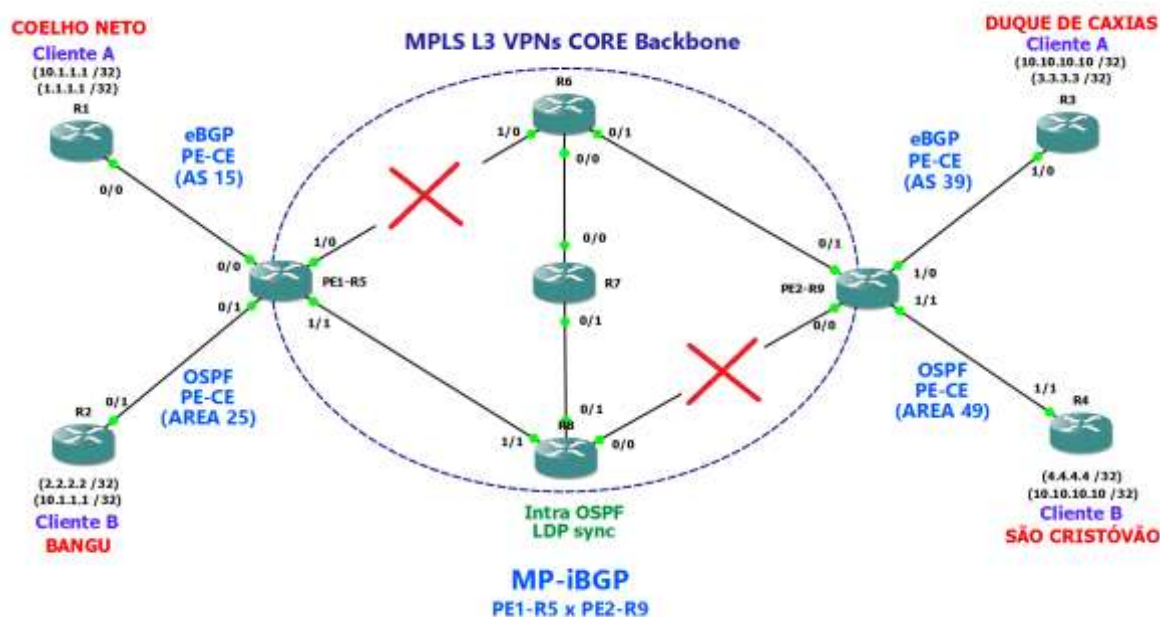
```
R1#
```

```
R1#
```

4.4.3 Testes de falhas de *links*

Foram simuladas também a queda de dois *links* do CORE MPLS, de modo a entender se a convergência do tráfego aconteceria normalmente ou não. A figura 4-4 ilustra quais links foram desabilitados para o teste.

Figura 4-4. Falha de links no experimento



Neste teste, os *links* físicos entre PE-R5 e R6 e entre R8 e PE-R9 são interrompidos para simular uma falha.

Com o modo de depuração MPLS ativado no roteador com o comando *debug mpls ldp messages*, que demonstrará as mensagens LDP enviadas e recebidas, e o *log adjacencies changes* do OSPF ativado, percebe-se que:

1. Cerca de 5 segundos após a queda do *link* com R6, o PE-R5 já percebe a vizinhança MPLS cair com ele e tenta enviar novas mensagens de *notify* para confirmar a queda (02:27:14.231).
2. Alguns milésimos de segundo depois, a vizinhança é reconhecida como *down* (02:27:14.239).

```

PE1-R5#debug mpls ldp messages received
LDP received messages, excluding periodic Keep Alive debugging
is on
PE1-R5#
PE1-R5#debug mpls ldp messages sent
LDP sent PDUs, excluding periodic Keep Alive debugging is on
PE1-R5#
*Mar  1 02:27:14.231: ldp: Sent notif msg to 6.6.6.6:0 (pp
0x654B9CC4)

```

```
*Mar 1 02:27:14.231: ldp: Sent notif msg to 6.6.6.6:0 (pp
0x654B9CC4)
```

```
PE1-R5#
```

```
*Mar 1 02:27:14.239: %LDP-5-NBRCHG: LDP Neighbor 6.6.6.6:0 (2)
is DOWN (Discovery Hello Hold Timer expired)
```

A tabela de roteamento reconhece a vizinhança OSPF como ausente com os *timers* de *expired* cerca de 25 segundos depois de todo este processo (02:27:39.635):

```
*Mar 1 02:27:39.635: %OSPF-5-ADJCHG: Process 1, Nbr 6.6.6.6 on
FastEthernet1/0 from FULL to DOWN, Neighbor Down: Dead timer expired
PE1-R5#
```

Na sequência, o mapeamento da LFIB MPLS é refeito na nuvem MPLS entre todos os roteadores.

```
*Mar 1 02:30:06.067: ldp: Rcvd init msg from 6.6.6.6 (pp 0x0)
*Mar 1 02:30:06.067: ldp: Sent init msg to 6.6.6.6:0 (pp 0x0)
*Mar 1 02:30:06.067: ldp: Sent keepalive msg to 6.6.6.6:0 (pp
0x0)
*Mar 1 02:30:06.127: ldp: Rcvd keepalive msg from 6.6.6.6:0 (pp
0x0)
*Mar 1 02:30:06.131: %LDP-5-NBRCHG: LDP Neighbor 6.6.6.6:0 (2)
is UP
*Mar 1 02:30:06.131: ldp: Sent address msg to 6.6.6.6:0 (pp
0x66023B8C)
*Mar 1 02:30:06.131: ldp: Sent 11 label mapping msgs to 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Rcvd address msg from 6.6.6.6:0 (pp
0x66023B8C)
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 24; prefix 6.6.6.6/32
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
```

```

*Mar 1 02:30:06.131: ldp: Pack msg; len 23; prefix
172.16.69.0/24
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 23; prefix
172.16.67.0/24
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 23; prefix
172.16.56.0/24
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 23; prefix
172.16.89.0/24
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 23; prefix
172.16.78.0/24
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 24; prefix 7.7.7.7/32
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 24; prefix 8.8.8.8/32
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 24; prefix 9.9.9.9/32
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 23; prefix
172.16.58.0/24
*Mar 1 02:30:06.131: ldp: Rcvd label mapping msg from 6.6.6.6:0
(pp 0x66023B8C)
*Mar 1 02:30:06.131: ldp: Pack msg; len 24; prefix 5.5.5.5/32
*Mar 1 02:30:06.131: ldp: Next msg not packable; Queue packed
msg buffer, len 166

```

4.4.3.1 Bindings MPLS e OSPF

Dada a falha dos *links*, todos os caminhos entre os roteadores dos clientes A e B passam agora por R7, antes não presente nos *traceroutes* por ter menor preferência frente à R6 e R8.

O caminho entre os PEs, agora único, é configurado por $R5 \leftarrow \Rightarrow R8 \leftarrow \Rightarrow R7 \leftarrow \Rightarrow R6 \leftarrow \Rightarrow R9$.

As ligações MPLS ficam, então, com *labels in use*, tendo somente um caminho para ligar os diferentes PEs.

```
PE1-R5#sh mpls ip bind 9.9.9.9 32
9.9.9.9/32
      in label:      507
      out label:     801      lsr: 8.8.8.8:0      inuse
```

```
R8#sh mpls ip bind 9.9.9.9 32
9.9.9.9/32
      in label:      801
      out label:     507      lsr: 5.5.5.5:0
      out label:     702      lsr: 7.7.7.7:0      inuse
```

```
R7#sh mpls ip bind 9.9.9.9 32
9.9.9.9/32
      in label:      702
      out label:     801      lsr: 8.8.8.8:0
      out label:     604      lsr: 6.6.6.6:0      inuse
```

```
R6#sh mpls ip bind 9.9.9.9 32
9.9.9.9/32
      in label:      604
      out label:     imp-null lsr: 9.9.9.9:0      inuse
      out label:     702      lsr: 7.7.7.7:0
```

Assim como diferentes *traceroutes* entre clientes, em diferentes sentidos.

- Sentido R1 \Rightarrow R3:

```
R1#traceroute 3.3.3.3
```

```
Type escape sequence to abort.
```

```
Tracing the route to 3.3.3.3
```

```

  1 192.168.15.5 16 msec 16 msec 24 msec
  2 172.16.58.8 [MPLS: Labels 801/908 Exp 0] 92 msec 68 msec 60
msec
  3 172.16.78.7 [MPLS: Labels 702/908 Exp 0] 72 msec 88 msec 48
msec
  4 172.16.67.6 [MPLS: Labels 604/908 Exp 0] 56 msec 68 msec 56
msec
  5 192.168.39.9 [AS 39] [MPLS: Label 908 Exp 0] 60 msec 40 msec
44 msec
  6 192.168.39.3 [AS 39] 72 msec 52 msec 60 msec
```

- Sentido R3 \Rightarrow R1:

```
R3#traceroute 1.1.1.1
```

```
Type escape sequence to abort.
```

```
Tracing the route to 1.1.1.1
```

```

  1 192.168.39.9 20 msec 16 msec 16 msec
  2 172.16.89.8 [MPLS: Labels 804/508 Exp 0] 64 msec 56 msec 64
msec
  3 192.168.15.5 [AS 15] [MPLS: Label 508 Exp 0] 40 msec 64 msec
40 msec
  4 192.168.15.1 [AS 15] 52 msec 44 msec 72 msec
```

- Sentido R2 \Rightarrow R4:

```
R2#traceroute 4.4.4.4
```

```
Type escape sequence to abort.
```

Tracing the route to 4.4.4.4

```

1 192.168.25.5 4 msec 44 msec 8 msec
2 172.16.56.6 [MPLS: Labels 605/908 Exp 0] 52 msec 56 msec 44
msec
3 192.168.49.9 [MPLS: Label 908 Exp 0] 20 msec 56 msec 24 msec
4 192.168.49.4 68 msec 28 msec 72 msec

```

- Sentido R4 \Rightarrow R2:

R4#traceroute 2.2.2.2

Type escape sequence to abort.

Tracing the route to 2.2.2.2

```

1 192.168.49.9 8 msec 8 msec 8 msec
2 172.16.69.6 [MPLS: Labels 606/511 Exp 0] 36 msec 40 msec 40
msec
3 192.168.25.5 [MPLS: Label 511 Exp 0] 40 msec 20 msec 40 msec
4 192.168.25.2 40 msec 44 msec 40 msec

```


5 Conclusão e trabalhos futuros

Nos dias atuais, onde diferentes tipos de serviço são trafegados por meio de meios físicos compartilhados, a busca por métodos cada vez mais eficazes para tratamento de dados com segurança e escalabilidade tem sido uma crescente. Este trabalho teve como objetivo demonstrar, por meio de um experimento prático, os conceitos necessários e a implementação de VPNs de camada 3 utilizando MPLS.

Com a realização deste estudo, fica clara a flexibilidade, escalabilidade, eficiência e segurança no uso de MPLS e Multiprotocol BGP para criação e transporte de VPNs de camada 3.

A comutação provida pelo MPLS, por suportar todos os diferentes tipos de encaminhamento (*unicast*, *unicast* com tipos de serviço e *multicast*), é utilizada como grande aliada em estudos avançados de engenharia de tráfego e QoS. O estudo realizado no capítulo 4 pôs em prática conceitos avançados que garantiram a divisão de tráfego de dois clientes que não se conheciam por meio de um mesmo *backbone*, e mostrou, assim, que este encaminhamento é possível em tem grandes vantagens relacionadas à segurança e processamento de dados nos roteadores envolvidos.

Como sugestão para trabalhos futuros, indica-se a emulação de diferentes tipos de tráfego neste *backbone* para realização de estudos de performance, assim como incremento contínuo das configurações com técnicas para avanço das configurações atuais, tanto no lado do CORE MPLS do provedor de serviço, quanto nos clientes atendidos:

O CORE poderia ser subdividido em mais ASs entre os PEs, com preferências para dados clientes ou tipos de tráfego (VoIP, SS7, tráfego de gerência, serviços Exchange), se precaver do tráfego dos clientes que se conectam à ele pelos PEs com configurações que mascarem o caminho em seu backbone, e utilizando diferentes equipamentos com finalidades fim mais específicas. Por outro lado, os clientes poderiam também tem mais subdivisões de roteadores atrás de seus CEs que conversassem ASs BGP e Áreas OSPF diferentes.

Referências Bibliográficas

1. Site da CISCO. **What is administrative distance?** Disponível em: <http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/15986-admin-distance.html>>. Acesso em: 10/12/2016.
2. HAGEN, Silvia. **IPv6 Essentials**. 2ª ed. Sebastopol: O'Reilly, 2006.
3. GUIMARÃES, Alexandre Guedes et al. **Segurança em Redes Privadas Virtuais - VPNs**. Rio de Janeiro: Brasport, 2006.
4. PEPELNJAK, Ivan; GUICHARD, Jim. **MPLS and VPN Architectures**. Indianapolis: Cisco Press, 2002.
5. LEWIS, Christopher S; PICKAVANCE, Steve. **Selecting MPLS VPN Services**: A guide to using and defining MPLS VPN Services. Indianapolis: Cisco Press, 2006.
6. GHEIN, Luc De. **MPLS Fundamentals**: A Comprehensive Introduction to MPLS Theory and Practice. Indianapolis: Cisco Press, 2007.
7. Site do Technology Training Limited. **MPLS Virtual Private Networks (VPNs)**. Disponível em: http://www.technology-training.co.uk/mplsvirtualprivatenetworksvpn_35.php>. Acesso em: 10/12/2016.
8. Site da Teleco. **O Protocolo MPLS**. http://www.teleco.com.br/tutoriais/tutorialmplscam/pagina_3.asp>. Acessado em: 06/12/2016.
9. CISCO SYSTEMS. **Comandos OSPF**. Disponível em: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/command/iro-cr-book/ospf-i1.html>. Acessado em: 27/11/2016.
10. CISCO SYSTEMS. **Comandos MPLS**. Disponível em: <http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mpls/command/mp-cr-book.html>>. Acessado em: 27/11/2016.
11. CISCO SYSTEMS. **Comandos BGP**. Disponível em: http://www.cisco.com/c/en/us/td/docs/ios/iproute_bgp/command/reference/irg_book.html>. Acessado em: 27/11/2016.
12. CISCO SYSTEMS. **Polarização CEF**. Disponível em: http://www.cisco.com/cisco/web/support/BR/111/1119/1119334_116376-technote-cef-00.html>. Acessado em: 29/11/2016.

13. CISCO SYSTEMS. **MPLS Label Distribution Protocol (LDP)**. Disponível em:
<http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t2/ftldp41.html>. Acessado em:
29/11/2016.
14. Site SearchEnterpriseWAN. **MPLS VPN Fundamentals Guide**. Disponível em:
<<http://searchenterprisewan.techtarget.com/guides/MPLS-VPN-fundamentals>>.
Acessado em: 30/11/2016.

Anexos

Como anexo, será disponibilizado também um documento com todas as configurações globais finais dos roteadores do experimento.

Este anexo tem como propósito servir de base para estudos futuros que queiram se utilizar das configurações envolvidas neste trabalho de maneira integral, uma vez que os capítulos de experimentação em si não contemplam as configurações finais de todos os equipamentos. Estes arquivos de configuração devem ser utilizados no GNS3 com a criação dos *links* da mesma maneira que é demonstrada na topologia do Capítulo 3.

Anexo – Configurações Globais finais dos roteadores

Configurações de PE-R5

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
  
!  
version 12.4  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname PE1-R5  
!  
boot-start-marker  
boot-end-marker  
!  
!  
no aaa new-model  
memory-size iomem 5
```

```
no ip icmp rate-limit unreachable
ip cef
!
!
!
!
ip vrf CLIENTE-A
rd 15:39
route-target export 15:39
route-target import 15:39
!
ip vrf CLIENTE-B
rd 25:49
route-target export 25:49
route-target import 25:49
!
no ip domain lookup
ip auth-proxy max-nodata-conns 3
ip admission max-nodata-conns 3
!
mpls label range 500 599
mpls label protocol ldp
!
!
!
!
!
!
!
!
!
```

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
ip tcp synwait-time 5  
!  
!  
!  
!  
!  
interface Loopback0  
ip address 5.5.5.5 255.255.255.255  
ip ospf 1 area 0  
!  
interface FastEthernet0/0  
ip vrf forwarding CLIENTE-A  
ip address 192.168.15.5 255.255.255.0  
duplex auto  
speed auto  
!  
interface FastEthernet0/1  
ip vrf forwarding CLIENTE-B  
ip address 192.168.25.5 255.255.255.0  
ip ospf 25 area 0  
duplex auto  
speed auto
```

```
!  
interface FastEthernet1/0  
  no switchport  
  ip address 172.16.56.5 255.255.255.0  
  ip ospf 1 area 0  
  mpls ip  
!  
interface FastEthernet1/1  
  no switchport  
  ip address 172.16.58.5 255.255.255.0  
  ip ospf 1 area 0  
  mpls ip  
!  
interface FastEthernet1/2  
!  
interface FastEthernet1/3  
!  
interface FastEthernet1/4  
!  
interface FastEthernet1/5  
!  
interface FastEthernet1/6  
!  
interface FastEthernet1/7  
!  
interface FastEthernet1/8  
!  
interface FastEthernet1/9  
!  
interface FastEthernet1/10  
!
```



```
interface FastEthernet1/11
!
interface FastEthernet1/12
!
interface FastEthernet1/13
!
interface FastEthernet1/14
!
interface FastEthernet1/15
!
interface Vlan1
  no ip address
!
router ospf 25 vrf CLIENTE-B
  router-id 192.168.25.5
  ispf
  log-adjacency-changes
  redistribute bgp 59 metric 9 subnets
!
router ospf 1
  mpls ldp sync
  router-id 5.5.5.5
  log-adjacency-changes
!
router bgp 59
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 9.9.9.9 remote-as 59
  neighbor 9.9.9.9 update-source Loopback0
!
address-family ipv4
```

```
neighbor 9.9.9.9 activate
no auto-summary
no synchronization
exit-address-family
!
address-family vpnv4
neighbor 9.9.9.9 activate
neighbor 9.9.9.9 send-community both
exit-address-family
!
address-family ipv4 vrf CLIENTE-B
redistribute ospf 25 vrf CLIENTE-B match internal external 1 external 2 nssa-external 1 nssa-external
2
no synchronization
exit-address-family
!
address-family ipv4 vrf CLIENTE-A
neighbor 192.168.15.1 remote-as 15
neighbor 192.168.15.1 activate
no synchronization
exit-address-family
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
!
!
```

```
mpls ldp router-id Loopback0
```

```
!
```

```
control-plane
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
line con 0
```

```
exec-timeout 0 0
```

```
privilege level 15
```

```
logging synchronous
```

```
line aux 0
```

```
exec-timeout 0 0
```

```
privilege level 15
```

```
logging synchronous
```

```
line vty 0 4
```

```
login
```

```
!
```

```
!
```

```
end
```

Configurações de PE-R9

```
!
```

```
!
```

!

!

!

!

!

!

!

!

!

!

!

version 12.4

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname PE2-R9

!

boot-start-marker

boot-end-marker

!

!

no aaa new-model

memory-size iomem 5

no ip icmp rate-limit unreachable

ip cef

!

!

!

!

```
ip vrf CLIENTE-A
rd 15:39
route-target export 15:39
route-target import 15:39
!
ip vrf CLIENTE-B
rd 25:49
route-target export 25:49
route-target import 25:49
!
no ip domain lookup
ip auth-proxy max-nodata-conns 3
ip admission max-nodata-conns 3
!
mpls label range 900 999
mpls label protocol ldp
!
!
!
!
!
!
!
!
!
!
!
!
!
!
```

```
!  
!  
!  
ip tcp synwait-time 5  
!  
!  
!  
!  
!  
!  
interface Loopback0  
  ip address 9.9.9.9 255.255.255.255  
  ip ospf 1 area 0  
!  
interface FastEthernet0/0  
  ip address 172.16.89.9 255.255.255.0  
  ip ospf 1 area 0  
  duplex auto  
  speed auto  
  mpls ip  
!  
interface FastEthernet0/1  
  ip address 172.16.69.9 255.255.255.0  
  ip ospf 1 area 0  
  duplex auto  
  speed auto  
  mpls ip  
!  
interface FastEthernet1/0  
  no switchport  
  ip vrf forwarding CLIENTE-A  
  ip address 192.168.39.9 255.255.255.0
```

```
!  
interface FastEthernet1/1  
  no switchport  
  ip vrf forwarding CLIENTE-B  
  ip address 192.168.49.9 255.255.255.0  
  ip ospf 49 area 0  
!  
interface FastEthernet1/2  
!  
interface FastEthernet1/3  
!  
interface FastEthernet1/4  
!  
interface FastEthernet1/5  
!  
interface FastEthernet1/6  
!  
interface FastEthernet1/7  
!  
interface FastEthernet1/8  
!  
interface FastEthernet1/9  
!  
interface FastEthernet1/10  
!  
interface FastEthernet1/11  
!  
interface FastEthernet1/12  
!  
interface FastEthernet1/13  
!
```

```
interface FastEthernet1/14
!
interface FastEthernet1/15
!
interface Vlan1
  no ip address
!
router ospf 49 vrf CLIENTE-B
  router-id 192.168.49.9
  ispf
  log-adjacency-changes
  redistribute bgp 59 metric 9 subnets
!
router ospf 1
  mpls ldp sync
  router-id 9.9.9.9
  log-adjacency-changes
!
router bgp 59
  no bgp default ipv4-unicast
  bgp log-neighbor-changes
  neighbor 5.5.5.5 remote-as 59
  neighbor 5.5.5.5 update-source Loopback0
!
address-family ipv4
  neighbor 5.5.5.5 activate
  no auto-summary
  no synchronization
  exit-address-family
!
address-family vpnv4
```



```
neighbor 5.5.5.5 activate
neighbor 5.5.5.5 send-community both
exit-address-family
!
address-family ipv4 vrf CLIENTE-B
  redistribute ospf 49 vrf CLIENTE-B match internal external 1 external 2 nssa-external 1 nssa-external
  2
  no synchronization
exit-address-family
!
address-family ipv4 vrf CLIENTE-A
  neighbor 192.168.39.3 remote-as 39
  neighbor 192.168.39.3 activate
  no synchronization
exit-address-family
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
!
!
mpls ldp router-id Loopback0
!
control-plane
!
!
!
```

```
!  
!  
!  
!  
!  
!  
!  
line con 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line aux 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line vty 0 4  
  login  
  !  
  !  
end
```

Configurações de R6

```
!  
!  
!  
!  
!  
!  
!  
!
```

!

!

!

!

!

!

!

!

!

!

!

!

!

!

version 12.4

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname R6

!

boot-start-marker

boot-end-marker

!

!

no aaa new-model

memory-size iomem 5

no ip icmp rate-limit unreachable

ip cef

!

!

!

!

no ip domain lookup

```
ip auth-proxy max-nodata-conns 3
```

```
ip admission max-nodata-conns 3
```

!

mpls label range 600 699

```
mpls label protocol ldp
```

!

!

!

!

!

!

!

!

!

!

!

!

!

1

!

!

!

!

```
ip tcp synwait-time 5
```

!

!

!

```
!  
!  
interface Loopback0  
ip address 6.6.6.6 255.255.255.255  
ip ospf 1 area 0  
!  
interface FastEthernet0/0  
ip address 172.16.67.6 255.255.255.0  
ip ospf 1 area 0  
duplex auto  
speed auto  
!  
interface FastEthernet0/1  
ip address 172.16.69.6 255.255.255.0  
ip ospf 1 area 0  
duplex auto  
speed auto  
!  
interface FastEthernet1/0  
no switchport  
ip address 172.16.56.6 255.255.255.0  
ip ospf 1 area 0  
!  
interface FastEthernet1/1  
!  
interface FastEthernet1/2  
!  
interface FastEthernet1/3  
!  
interface FastEthernet1/4  
!
```

```
interface FastEthernet1/5
!
interface FastEthernet1/6
!
interface FastEthernet1/7
!
interface FastEthernet1/8
!
interface FastEthernet1/9
!
interface FastEthernet1/10
!
interface FastEthernet1/11
!
interface FastEthernet1/12
!
interface FastEthernet1/13
!
interface FastEthernet1/14
!
interface FastEthernet1/15
!
interface Vlan1
  no ip address
!
router ospf 1
  mpls ldp sync
  mpls ldp autoconfig area 0
  router-id 6.6.6.6
  log-adjacency-changes
!
```

```
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
!
!
mpls ldp router-id Loopback0
!
control-plane
!
!
!
!
!
!
!
!
!
!
!
line con 0
exec-timeout 0 0
privilege level 15
logging synchronous
line aux 0
exec-timeout 0 0
privilege level 15
logging synchronous
line vty 0 4
```

```
login
```

```
!
```

```
!
```

```
end
```

Configurações de R7

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
version 12.4
```


service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname R7

!

boot-start-marker

boot-end-marker

!

!

no aaa new-model

memory-size iomem 5

no ip icmp rate-limit unreachable

ip cef

!

!

!

!

no ip domain lookup

ip auth-proxy max-nodata-conns 3

ip admission max-nodata-conns 3

!

mpls label range 700 799

mpls label protocol ldp

!

!

!

!

!

!

!

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
ip tcp synwait-time 5  
!  
!  
!  
!  
!  
!  
interface Loopback0  
  ip address 7.7.7.7 255.255.255.255  
  ip ospf 1 area 0  
!  
interface FastEthernet0/0  
  ip address 172.16.67.7 255.255.255.0  
  ip ospf 1 area 0  
  duplex auto  
  speed auto  
!  
interface FastEthernet0/1  
  ip address 172.16.78.7 255.255.255.0  
  ip ospf 1 area 0  
  duplex auto
```

```
speed auto
!
interface FastEthernet1/0
!
interface FastEthernet1/1
!
interface FastEthernet1/2
!
interface FastEthernet1/3
!
interface FastEthernet1/4
!
interface FastEthernet1/5
!
interface FastEthernet1/6
!
interface FastEthernet1/7
!
interface FastEthernet1/8
!
interface FastEthernet1/9
!
interface FastEthernet1/10
!
interface FastEthernet1/11
!
interface FastEthernet1/12
!
interface FastEthernet1/13
!
interface FastEthernet1/14
```

```
!  
interface FastEthernet1/15  
!  
interface Vlan1  
  no ip address  
!  
router ospf 1  
  mpls ldp sync  
  mpls ldp autoconfig  
  router-id 7.7.7.7  
  log-adjacency-changes  
!  
ip forward-protocol nd  
!  
!  
no ip http server  
no ip http secure-server  
!  
no cdp log mismatch duplex  
!  
!  
mpls ldp router-id Loopback0  
!  
control-plane  
!  
!  
!  
!  
!  
!  
!
```

```
!  
!  
!  
line con 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line aux 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line vty 0 4  
  login  
!  
!  
end
```

Configurações de R8

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!
```

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
  
!  
version 12.4  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname R8  
!  
boot-start-marker  
boot-end-marker  
!  
!  
no aaa new-model  
memory-size iomem 5  
no ip icmp rate-limit unreachable  
ip cef  
!  
!  
!  
!  
no ip domain lookup
```

```
ip auth-proxy max-nodata-conns 3
```

```
ip admission max-nodata-conns 3
```

```
!
```

```
mpls label range 800 899
```

```
mpls label protocol ldp
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
ip tcp synwait-time 5
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
interface Loopback0
```

```
ip address 8.8.8.8 255.255.255.255
```

```
ip ospf 1 area 0
!
interface FastEthernet0/0
ip address 172.16.89.8 255.255.255.0
ip ospf 1 area 0
duplex auto
speed auto
!
interface FastEthernet0/1
ip address 172.16.78.8 255.255.255.0
ip ospf 1 area 0
duplex auto
speed auto
!
interface FastEthernet1/0
!
interface FastEthernet1/1
no switchport
ip address 172.16.58.8 255.255.255.0
ip ospf 1 area 0
!
interface FastEthernet1/2
!
interface FastEthernet1/3
!
interface FastEthernet1/4
!
interface FastEthernet1/5
!
interface FastEthernet1/6
!
```



```
interface FastEthernet1/7
!
interface FastEthernet1/8
!
interface FastEthernet1/9
!
interface FastEthernet1/10
!
interface FastEthernet1/11
!
interface FastEthernet1/12
!
interface FastEthernet1/13
!
interface FastEthernet1/14
!
interface FastEthernet1/15
!
interface Vlan1
  no ip address
!
router ospf 1
  mpls ldp sync
  mpls ldp autoconfig area 0
  router-id 8.8.8.8
  log-adjacency-changes
!
ip forward-protocol nd
!
!
no ip http server
```

```
no ip http secure-server
!
no cdp log mismatch duplex
!
!
mpls ldp router-id Loopback0
!
control-plane
!
!
!
!
!
!
!
!
!
!
line con 0
exec-timeout 0 0
privilege level 15
logging synchronous
line aux 0
exec-timeout 0 0
privilege level 15
logging synchronous
line vty 0 4
login
!
!
end
```

Configurações de R1

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
  
!  
  
version 12.4  
  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
  
!  
  
hostname R1  
  
!  
  
boot-start-marker  
boot-end-marker  
  
!  
!  
  
no aaa new-model  
  
memory-size iomem 5
```

no ip icmp rate-limit unreachable

ip cef

!

!

!

!

no ip domain lookup

ip auth-proxy max-nodata-conns 3

ip admission max-nodata-conns 3

!

!

!

!

!

!

!

!

!

!

!

!

!

!

!

!

!

!

!

ip tcp synwait-time 5

!

!

```
!  
!  
!  
interface Loopback0  
  ip address 1.1.1.1 255.255.255.255  
!  
interface Loopback1  
  ip address 10.1.1.1 255.255.255.255  
!  
interface FastEthernet0/0  
  ip address 192.168.15.1 255.255.255.0  
  duplex auto  
  speed auto  
!  
interface FastEthernet0/1  
  no ip address  
  shutdown  
  duplex auto  
  speed auto  
!  
router bgp 15  
  no bgp default ipv4-unicast  
  bgp log-neighbor-changes  
  neighbor 192.168.15.5 remote-as 59  
!  
address-family ipv4  
  redistribute connected  
  neighbor 192.168.15.5 activate  
  no auto-summary  
  no synchronization  
exit-address-family
```

```
!  
ip forward-protocol nd  
!  
!  
no ip http server  
no ip http secure-server  
!  
no cdp log mismatch duplex  
!  
!  
!  
control-plane  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
line con 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line aux 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line vty 0 4
```

```
login
```

```
!
```

```
!
```

```
end
```

Configurações de R2

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!
```

```
version 12.4
```

```
service timestamps debug datetime msec
```

```
service timestamps log datetime msec
```

```
no service password-encryption
```

```
!
```

```
hostname R2
```

```
!
```

```
boot-start-marker
```

```
boot-end-marker
```

```
!
```



```
ip tcp synwait-time 5
!
!
!
!
!
interface Loopback0
ip address 2.2.2.2 255.255.255.255
ip ospf 25 area 0
!
interface Loopback1
ip address 10.1.1.1 255.255.255.255
ip ospf 25 area 0
!
interface FastEthernet0/0
no ip address
shutdown
duplex auto
speed auto
!
interface FastEthernet0/1
ip address 192.168.25.2 255.255.255.0
ip ospf 25 area 0
duplex auto
speed auto
!
router ospf 25
router-id 2.2.2.2
log-adjacency-changes
!
ip forward-protocol nd
```

```
!  
!  
no ip http server  
no ip http secure-server  
!  
no cdp log mismatch duplex  
!  
!  
!  
control-plane  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
line con 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line aux 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line vty 0 4  
  login  
!
```

```
!  
end
```

Configurações de R3

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
  
!  
version 12.4  
service timestamps debug datetime msec  
service timestamps log datetime msec  
no service password-encryption  
!  
hostname R3  
!  
boot-start-marker  
boot-end-marker  
!  
!
```



```
!  
!  
!  
!  
!  
interface Loopback0  
  ip address 3.3.3.3 255.255.255.255  
!  
interface Loopback1  
  ip address 10.10.10.10 255.255.255.255  
!  
interface FastEthernet0/0  
  no ip address  
  shutdown  
  duplex auto  
  speed auto  
!  
interface FastEthernet0/1  
  no ip address  
  shutdown  
  duplex auto  
  speed auto  
!  
interface FastEthernet1/0  
  no switchport  
  ip address 192.168.39.3 255.255.255.0  
!  
interface FastEthernet1/1  
!  
interface FastEthernet1/2  
!
```

```
interface FastEthernet1/3
!
interface FastEthernet1/4
!
interface FastEthernet1/5
!
interface FastEthernet1/6
!
interface FastEthernet1/7
!
interface FastEthernet1/8
!
interface FastEthernet1/9
!
interface FastEthernet1/10
!
interface FastEthernet1/11
!
interface FastEthernet1/12
!
interface FastEthernet1/13
!
interface FastEthernet1/14
!
interface FastEthernet1/15
!
interface Vlan1
  no ip address
!
router bgp 39
  no bgp default ipv4-unicast
```

```
bgp log-neighbor-changes
neighbor 192.168.39.9 remote-as 59
!
address-family ipv4
  redistribute connected
  neighbor 192.168.39.9 activate
  no auto-summary
  no synchronization
exit-address-family
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
!
!
!
control-plane
!
!
!
!
!
!
!
!
!
!
```


service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname R4

!

boot-start-marker

boot-end-marker

!

!

no aaa new-model

memory-size iomem 5

no ip icmp rate-limit unreachable

ip cef

!

!

!

!

no ip domain lookup

ip auth-proxy max-nodata-conns 3

ip admission max-nodata-conns 3

!

!

!

!

!

!

!

!

!

!

```
!  
!  
!  
!  
!  
!  
!  
!  
!  
!  
ip tcp synwait-time 5  
!  
!  
!  
!  
!  
interface Loopback0  
ip address 4.4.4.4 255.255.255.255  
ip ospf 49 area 0  
!  
interface Loopback1  
ip address 10.10.10.10 255.255.255.255  
ip ospf 49 area 0  
!  
interface FastEthernet0/0  
no ip address  
shutdown  
duplex auto  
speed auto  
!  
interface FastEthernet0/1  
no ip address
```

```
shutdown
duplex auto
speed auto
!
interface FastEthernet1/0
!
interface FastEthernet1/1
no switchport
ip address 192.168.49.4 255.255.255.0
ip ospf 49 area 0
!
interface FastEthernet1/2
!
interface FastEthernet1/3
!
interface FastEthernet1/4
!
interface FastEthernet1/5
!
interface FastEthernet1/6
!
interface FastEthernet1/7
!
interface FastEthernet1/8
!
interface FastEthernet1/9
!
interface FastEthernet1/10
!
interface FastEthernet1/11
!
```

```
interface FastEthernet1/12
```

```
!
```

```
interface FastEthernet1/13
```

```
!
```

```
interface FastEthernet1/14
```

```
!
```

```
interface FastEthernet1/15
```

```
!
```

```
interface Vlan1
```

```
no ip address
```

```
!
```

```
router ospf 49
```

```
router-id 4.4.4.4
```

```
ispf
```

```
log-adjacency-changes
```

```
!
```

```
ip forward-protocol nd
```

```
!
```

```
!
```

```
no ip http server
```

```
no ip http secure-server
```

```
!
```

```
no cdp log mismatch duplex
```

```
!
```

```
!
```

```
!
```

```
control-plane
```

```
!
```

```
!
```

```
!
```

```
!
```

```
!  
!  
!  
!  
!  
!  
line con 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line aux 0  
  exec-timeout 0 0  
  privilege level 15  
  logging synchronous  
line vty 0 4  
  login  
!  
!  
end
```