DCSG-P4: Disaggregated Cell Site Gateway (DCSG) em pipeline programável de pacotes interoperável

Kleber M. S. Rezende[1], Alan T. da Silva[1], Rodrigo Pierini[1], Christian E. Rothenberg[1]

[1]Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP), Campinas/SP, Brasil

{k232154,a265560}@dac.unicamp.br, {rpierini,chesteve}@unicamp.br

Resumo. *À medida que a arquitetura de redes 5G amadurece, novos serviços tornam-se disponíveis para as operadoras de redes móveis e provedores de serviço. A implementação de tais serviços, aliada à busca pela independência de fornecedor e potencializada pelas tecnologias SDN, é um dos fatores que impulsionam a inovação. Dentro deste contexto, o cell site gateway (CSG) se destaca como um elemento de grande densidade e, por isso, soluções que reduzam os custos destes equipamentos sem comprometer qualidade são bem vindas. O Telecom Infra Project (TIP) estabeleceu algumas especificações que orientam o desenvolvimento destes elementos de rede, considerando a separação dos planos de controle e de dados, denominados de Disaggregated CSG (DCSG). Este trabalho apresenta uma proposta para a implementação de um DCSG virtualizado, com características comumente utilizadas por Provedores de Serviço, baseadas nas especificações do TIP. As principais contribuições deste trabalho incluem a implementação do pipeline projetado, a adaptação para hardware Tofino de baixo custo e a validação experimental em testbed no qual foram conduzidos testes funcionais e de interoperabilidade. Os resultados obtidos demostraram que o encaminhamento de pacotes realizado pela nossa solução é adequado para a interoperar com equipamentos IP/MPLS tradicionais de função fixa.*

Abstract. *As 5G network architecture matures, new services become available to mobile network operators and service providers. The adoption of such services, coupled with the pursuit for supplier independence, enhanced by SDN technologies, drives increased pace for innovation. Within this context, the cell site gateway (CSG) represents high density elements and, therefore, solutions that reduce the equipment costs without loss of quality are being pursued. The Telecom Infra Project (TIP) brought some specifications that guide the creation of these network elements, considering the disaggregation of control and data planes, known as Disaggregated CSG (DCSG). In this work, we propose the implementation of a virtualized DCSG, incorporating characteristics normally used by Service Providers, based on the TIP specifications. Our main contributions are: implementation of the designed pipeline; prototype implementation in low-cost Tofino hardware; testbed experiments for functional and interoperability tests. The obtained results indicate that the packet forwarding, performed by our solution, is suitable for interoperability with legacy fixed-function equipment.*

1. Introdução

Operadoras de redes móveis e provedores de serviços de Internet, tradicionalmente, dependem de fornecedores específicos para projetar e construir sua estrutura de comunicação. Tal dependência torna as atualizações mais lentas, aumenta os custos e dificulta a inovação. Com o esperado aumento da densidade das redes móveis celulares de quinta geração (5G) em relação às redes de quarta geração (4G), mais equipamentos de um mesmo fornecedor serão necessários para atender às demandas de densidade destas redes. Isto pode contribuir para um risco extra de aprisionamento a um determinado fornecedor [Chaudhary 2020].

A adoção de soluções abertas e desagregadas, como *Software-Defined Networking* (SDN) [Kreutz et al. 2014] [Foundation 2024] e *Network Function Virtualization* (NFV) [Rosa et al. 2014], permitiu que os Provedores de Serviços em Nuvem pudessem realizar inovações mais rapidamente. Neste contexto, o *Programming Protocol-Independent Packet Processors* (P4) [Org 2024] apresenta-se como um dos principais marcos no mundo SDN. P4 refere-se a uma linguagem específica de domínio para dispositivos de rede que permite definir como o plano de dados de roteadores e *switches* processam pacotes. Estes dispositivos, quando habilitados para P4, são independentes de protocolo. Além disso, o P4 fornece um modelo abstrato adequado para programar o plano de dados da rede que delineia os cabeçalhos e especifica os comportamentos do *parser*/*deparser* e processamento dos pacotes.

Na arquitetura de redes 5G encontram-se distribuídos diversos dispositivos de encaminhamento de pacotes, dentre eles o *Cell Site Gateway* (CSG), também conhecido como *Cell Site Router* (CSR). Este elemento de rede é um componente de Camada 2 (L2) e Camada 3 (L3) da rede *backhaul* de telecomunicações que realiza o transporte de dados da rede de acesso celular para a rede *core* do lado do provedor de serviço e vice-versa, conforme ilustrado na Figura 1.

Normalmente, as estações base móveis são conectadas a um CSG usando interfaces *Gigabit Ethernet* RJ45 ou *Small Form-factor Pluggable* (SFP) tradicionais. No entanto, com a recente implantação de redes 5G, as estações base utilizarão inter-

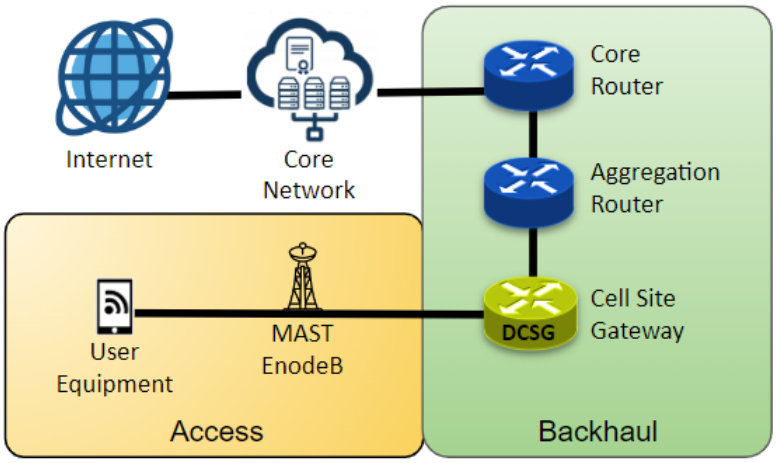


Figura 1. DCSG localizado na borda da Rede Backhaul. Adaptado de [Marcom 2021]

faces SFP+ de 10 *Gigabit Ethernet* para acomodar os crescentes requisitos de alta capacidade. Isso significa que a maioria dos CSGs atualmente implantados tornaram-se inadequados para transportar o tráfego da estação base 5G. A observação destes fatos instigou o surgimento de iniciativas como o *Telecom Infra Project* (TIP) e seus diversos projetos como o *Disaggregated Cell Site Gateway* (DCSG) sob gestão do grupo *Open Optical and Packet Transport* (OOPT) [Atrio et al. 2019]. Com a arquitetura de plataforma de hardware aberta, as operadoras agora têm a opção de escolher um dos vários sistemas operacionais de rede oferecidos pelo fornecedor de software de sua preferência.

O protocolo *Multi-Protocol Label Switching* (MPLS) [Rosen et al. 2001] fornece tratamento eficiente para transmissão de pacotes IP na rede ao facilitar o roteamento desses pacotes em rotas pré-determinadas, chamadas de *Label-Switched Path* (LSP). As redes tradicionais utilizam o MPLS para troca de rótulos sempre que um fluxo de tráfego precisa ser habilitado na topologia da rede, e cada nó mantém um estado de perfluxo exclusivo. Embora seja um protocolo com mais de duas décadas de uso, de acordo com o levantamento feito por [Depasquale et al. 2023], o serviço MPLS ainda é dominante quando é preciso executar a escolha da tecnologia *fronthaul* e *midhaul* para sites de macrocélulas. O estudo ainda conclui que o MPLS foi identificado como uma etapa intermediária e de interesse atual para os Provedores de Serviços de Comunicações.

Ainda segundo o TIP [Atrio et al. 2019], o principal caso de uso previsto para o DCSG é atuar em *backhaul* 2G/3G/4G/5G para a rede IP/MPLS de transporte móvel. No entanto, a natureza de rede aberta do DCSG permite que diversas aplicações de *backhaul* diferentes sejam implementadas na mesma plataforma *whitebox*. Aplicações adicionais, além da agregação de *cell site* para *backhaul* móvel, incluem:

- Unidade interna para *backhaul* de micro-ondas
- Roteador de borda do provedor
- Agregação de *fronthaul* com suporte TSN[1]
- DCI[2] e *fronthaul* regional com suporte OpenZR+
- Redes 5G privadas

Este trabalho visa a implementação de um DCSG para atuar como roteador de borda de um provedor de serviços, usando a linguagem P4. A intenção final é apresentar uma solução de *software* que utiliza tecnologias *open source* e que possa ser executada em uma plataforma de *hardware* de baixo custo, como o Tofino ASIC, em conjunto com equipamentos legados.

A organização deste trabalho segue a seguinte ordem. Seção 2 descreve os trabalhos relacionados. A Seção 3 aborda a definição do problema e a solução proposta. Os resultados obtidos são apresentados na Seção 4, enquanto a Seção 5 expõe as conclusões e discussões sobre trabalhos futuros.

[1]Time-Sensitive Networking - https://1.ieee802.org/tsn/

[2]Data Center Interconnect - https://www.ciena.com/insights/what-is/What-is-DCI.html

2. Trabalhos Relacionados

Ao passo que o paradigma SDN se estabelece como sucessor das redes tradicionais, existem desafios pertinentes de interoperabilidade entre protocolos de comunicação destas redes IP/MPLS legadas com dispositivos programáveis do plano de dados, tais como os *switches* P4. Alguns trabalhos com essa perspectiva foram analisados.

Uma arquitetura baseada em SDN para gerenciamento de uma rede *Traffic Engineering Diffserv Aware* (DS-TE) MPLS é apresentada em [Bahnasse et al. 2018]. Um ambiente híbrido simulado foi desenvolvido para testar a arquitetura proposta: o plano de dados emulado no software *Graphical Network Simulator 3* (GNS3) com roteadores CiscoXR e 7200 IOS e o OpenDay-Light como o plano de controle. O tráfego de dados foi composto por VoIP, video, HTTP e ICMP. Apesar da abordagem inserida no contexto SDN, é ausente no trabalho a questão da programabilidade de redes com uma linguagem específica como, por exemplo, P4.

Em [Liu et al. 2022] é apresentado o modelo *Domain Programming Router* (DPRouter), baseado na arquitetura *Server-Switch* [Lu et al. 2011] que utiliza um processador de dados dpDPU baseado em *Reconfigurable Match Tables* (RMT) com suporte ao gerenciamento unificado da tabela de fluxo do plano de dados, implementado em FPGA programável. Embora o artigo, mencione o uso de programabilidade P4, apenas o roteamento e encaminhamento baseado em nome *Named-Data Network* (NDN) e algumas funcionalidades relacionadas com segurança são apresentados como ações realizadas pelo plano de dado. Nada é informado acerca do suporte de *features* relacionadas com protocolos conhecidos. No entanto, os autores mencionam que o conjunto de instruções personalizadas de domínio RMT pode ser estendido num trabalho futuro.

O BNG construído por [Kundel et al. 2019] refere-se ao projeto e à implementação de um plano de dados BNG de código aberto que atende às demandas de *Gateways* de Rede de Banda Larga em ambientes de nível de operadora. A implementação é baseada na arquitetura *Central Office Re-architected as a Datacenter* (CORD), criada pela *Open Networking Foundation* (ONF), que suporta a execução da funcionalidade de escritório central, usando VNFs em *hardware commodity*, cujo objetivo é superar a dependência de fornecedores e *hardware* proprietário em redes de operadoras. O pipeline construído aborda fluxo *upstream* e *downstream* de forma diferente e utiliza de um controlador para auxiliar na tarefa de autenticação e autorização do assinante e configuração da sessão PPPoE. Os testes e avaliação do pipeline foram realizados nos *targets* BMv2, P4-NetFPGAs e Netronome P4-SmartNICs. Além disso, algumas estimativas são dadas para a implementação no Tofino. A arquitetura ainda dispõe de mecanismo de segurança para evitar a falsificação de endereço de origem IP.

O paradigma da desacoplação das funções de identificação e de localização de um host na Internet foi abordado por [Steinert et al. 2023], que criaram o pipeline P4-LISP. Trata-se de um roteador LISP (*Locator/Identifier Separation Protocol*) de alto desempenho, implementado em P4, que suporta todos os recursos relevantes, como *Ingress Tunnel Router* (ITR), *Egress Tunnel Router* (ETR), dentre outros. O código foi compilado no Tofino ASIC e o seu desempenho foi exaustivamente testado.

Embora tenha suporte para o IPv4, o foco principal do trabalho está na definição dos túneis que permitem a comunicação entre hosts de diferentes domínios LISP.

[Abhishek Singh et al. 2021] criaram um cenário onde foi configurado o *Segment Routing – Traffic Engineering* (SR-TE) usando roteadores CISCO IOSXRv no Cisco Modeling Labs. O SR *Path Computation Element* (SR-PCE) construído no IOS-XRv também se enquadra na topologia. Sempre que a situação da rede muda, as políticas de caminho dinâmico, como métrica IGP, métrica TE, métrica HopCount e métrica de latência, são introduzidas na topologia da rede e o caminho é recalculado e implementado automaticamente. Sob essas condições de política, um desempenho de escalabilidade melhor e eficaz é alcançado. Neste artigo, o SR-TE é implementado com roteamento por segmentos sobre o plano de dados MPLS de políticas dinâmicas.

Em [Ollora Zaballa et al. 2021], é abordada a implementação e integração do protocolo *In-Band Network Telemetry* em switches programáveis P4 sobre uma rede SDN híbrida. Neste cenário, são discutidas as restrições que precisam ser administradas de forma que os *switches* programáveis P4 possam interagir com os dispositivos MPLS legados.

Tabela 1. Comparação de trabalhos relacionados.

Referência do trabalho	**Implementação**	**Programabilidade do Data Plane**	**Target**	**Features configuráveis**
[Bahnasse et al. 2018]	Arquitetura baseada em SDN para o gerenciamento de uma rede DS-TE	Não	Cisco IOS-XR e 7200 IOS executados no GNS3	Não se aplica
[Liu et al. 2022]	Modelo genérico de router	Sim (P4)	FPGA, Server-Switch	Não informado
[Kundel et al. 2019]	BNG	Sim (P4)	BMv2, FPGA, P4-SmartNIC, Tofino ASIC	PPPoE, VLAN, MPLS, IPv4
[Steinert et al. 2023]	Router LISP	Sim (P4)	Tofino ASIC	NAT, IPv4
[Abhishek Singh et al. 2021]	Políticas de caminho dinâmico	Não	CISCO IOSXRv no Cisco Modeling Labs	Não se aplica
[Ollora Zaballa et al. 2021]	Modelo genérico de router	Sim (P4)	P4-SmartNIC, Tofino ASIC	MPLS
Nossa Solução para DCSG	DCSG	Sim (P4)	BMv2, Tofino ASIC	VLAN, QinQ, MPLS, IPv4

3. DCSG-P4: Projeto do DCSG na linguagem P4

3.1. Definição do Problema

Os CSGs, atualmente utilizados nas redes 4G, possuem arquiteturas monolíticas que dificultam a diversificação de fornecedores por parte das operadoras e provedores. Esta é uma característica que torna desafiadora a implementação de redes de comunicação no padrão 5G e *5G-and-Beyond*, a curto e médio prazo, e no padrão 6G, a longo prazo, devido à necessidade latente de interoperabilidade entre os equipamentos tradicionais e os novos dispositivos de rede programável.

A desagregação dos planos de controle e de dados, principal característica trazida pelo paradigma SDN, é usada como alternativa para minimizar este problema

e, com ela, iniciativas de construção de CSG desagregados começam a ficar em evidência, como o DCSG proposto pelo TIP.

3.2. Desenvolvimento em P4 da lógica do pipeline DCSG

De forma que os critérios e características discutidos nas seções anteriores fossem devidamente correspondidos na implementação do projeto, foi desenvolvido um programa na linguagem P4 denominado *dcsg.p4*. Uma visão geral do *pipeline* do programa é apresentada na figura 2 e permite a análise e encaminhamento de pacotes Ethernet com ou sem marcação de VLAN/QinQ[3], além de pacotes MPLS e IPv4.

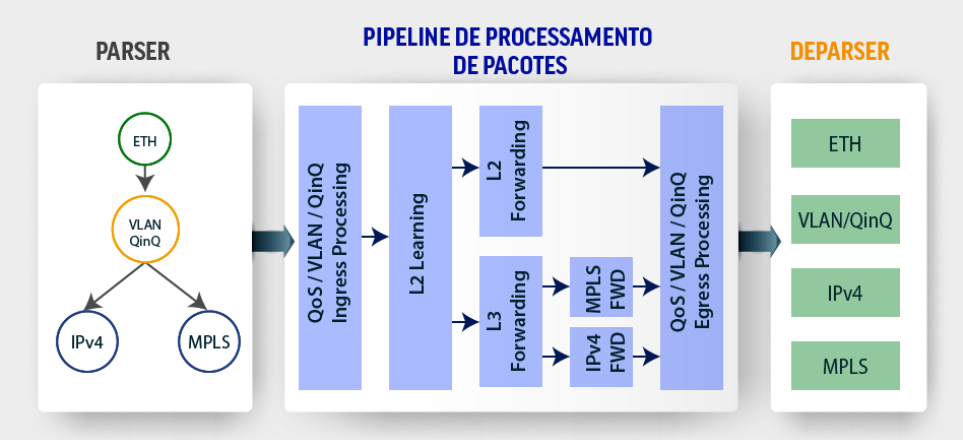


Figura 2. Pipeline do DCSG implementado

3.2.1. Detalhamento do Parser

O bloco Parser é responsável pela análise dos cabeçalhos encontrados nos pacotes que chegam ao switch programável. Tal análise consiste na verificação do conteúdo de campos presentes nestes cabeçalhos, para tomada de decisões dentro do próprio bloco e nos blocos subsequentes. Uma visão mais detalhada do parser pode ser vista na figura 3. A primeira coisa que deve ser verificada é se o pacote veio do controlador (via CPU_PORT). Se for este caso, deve-se analisar o conteúdo constante no cabeçalho especialmente criado para este tipo de comunicação. Como será mostrado mais adiante, a principal informação obtida nesta situação é a porta de saída indicada pelo controlador.

Na sequência, o Parser verifica a existência de marcações de VLAN no pacote. Se houver, os valores de *vlan_id* encontrados são armazenados em campos específicos nos metadados locais. Caso contrário, um *vlan_id* padrão será definido. Se o pacote chegar com marca(s) de VLAN, o valor do *vlan_id* mais externo, contido na pilha de cabeçalhos, também será armazenado nos metadados locais. Finalmente, se o campo *Ether_type* sinalizar a presença de outra marca de VLAN no mesmo pacote (QinQ), o Bloco *Parse Inner VLAN Tag* fará o armazenamento do *vlan_id* mais interno, informado no cabeçalho, no campo *inner_vlan_id* dos metadados locais.

[3]https://www.ieee802.org/1/pages/802.1ad.html

Posteriormente, verifica-se qual estrutura está encapsulada no *frame* Ethernet. Para este projeto, espera-se pacotes MPLS, IPv4, ARP/RARP. Os protocolos *Link Layer Discover Protocol* (LLDP) e *Broadcast Domain Discovery Protocol* (BDDP) foram considerados apenas para que controladores, que venham a interagir com nosso pipeline, possam ter meios de descobrir a topologia da rede que está sendo controlada, porém, a discussão e análise destes mecanismos estão fora do escopo deste projeto. Pacotes que não contenham os cabeçalhos previstos neste parser, são marcados para serem descartados na etapa de processamento que será detalhada mais adiante. O bloco *Deparser* é responsável pela inserção de cabeçalhos nos pacotes que serão enviados por uma ou mais interfaces do equipamento.

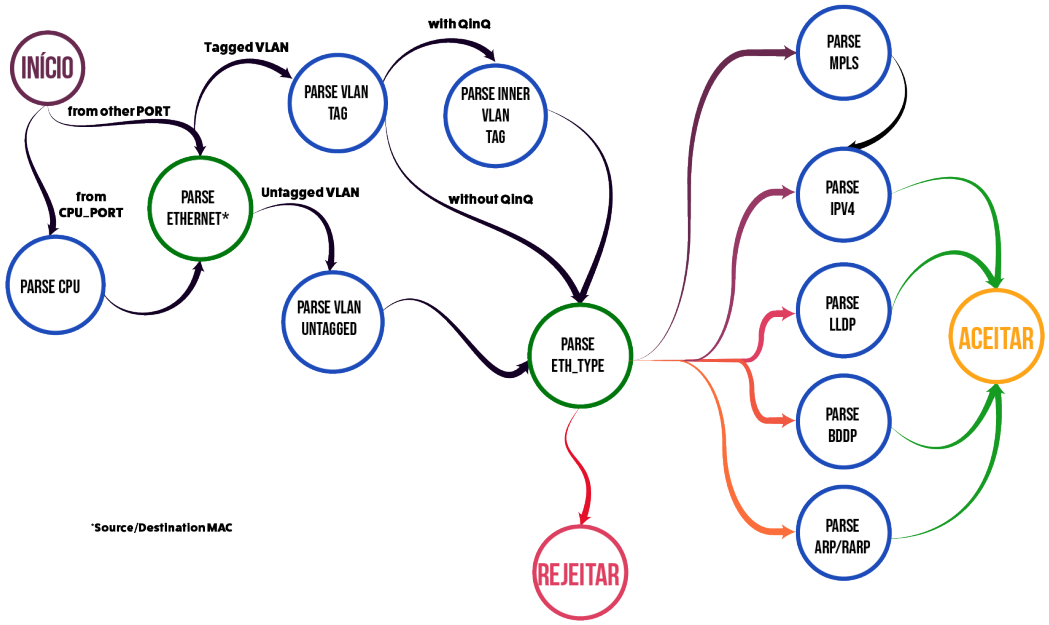


Figura 3. Detalhamento do Parser do DCSG implementado

3.2.2. Detalhamento do Pipeline de Processamento de Pacotes

O Pipeline de Processamento de Pacotes é composto por blocos de controle combinados com tabelas *match-action*, por meio das quais definem-se as ações que podem ser executadas e as chaves que são consideradas para selecionar a ação desejada. A figura 4 mostra que, logo após a etapa de Parser, é feita uma verificação para determinar se o pacote foi marcado para descarte. Caso não tenha havido tal marcação, alguns campos dos metadados intrínsecos do *target* são copiados para os metadados locais. O objetivo desta ação é tornar mais fácil o processo de portabilidade do código entre *targets* diferentes.

Na eventualidade do pacote ser recebido a partir do controlador, utiliza-se a porta de saída determinada pelo plano de controle. Em seguida, realiza-se o mapeamento de campos de QoS conforme definido na tabela *qos_mapping*, atualiza-se os metadados intrínsecos com as informações relevantes e encaminha-se o pacote para pipeline de saída (*egress pipeline*). Caso contrário, o pacote será submetido a etapa de filtragem, podendo ser descartado ou não. Se aceito, o processo de aprendizagem

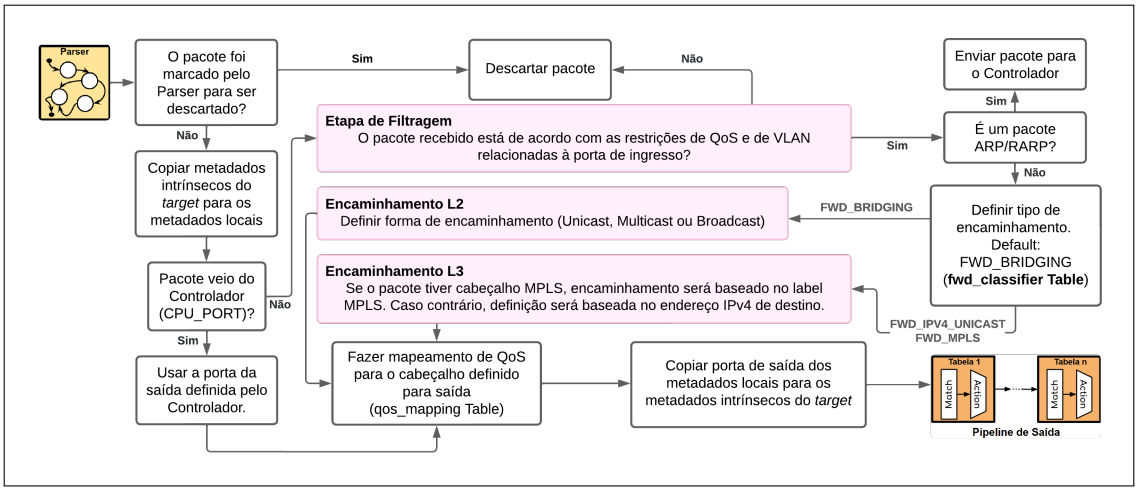


Figura 4. Visão macro do Pipeline de Entrada do DCSG-P4

layer 2 será executado, enviando pacotes ARP/RARP para o controlador. Se o pacote não for deste tipo, inicia-se o processo de determinação do tipo encaminhamento que será executado. As etapas de filtragem, de encaminhamento *layer* 2 e encaminhamento *layer* 3 serão detalhadas a seguir.

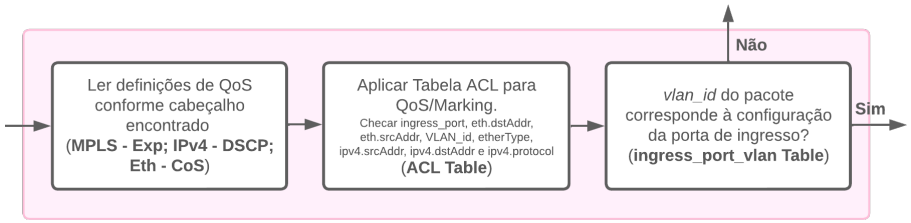
Etapa de Filtragem

Inicialmente, as informações de QoS contidas no pacote recebido (MPLS - *Experimental Bits* - Exp; IPv4 - *Differentiated Services Field Codepoints* - DCSP; e/ou Ethernet - *Class of Service* - CoS) são extraídas conforme ilustrado na figura 5(a). Estas informações são copiadas para campos específicos dos metadados locais, de acordo com a presença do respectivo cabeçalho no pacote. Listas de controle de acesso podem ser especificadas por meio da tabela ACL, considerando os campos mostrados na figura.

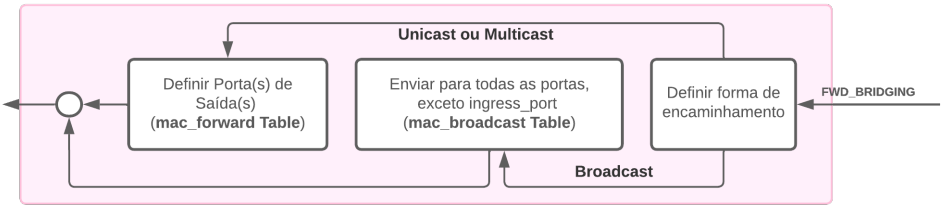
Por meio da tabela *ingress_port_vlan* pode ser realizado o processo de filtragem baseado em VLANs. Basicamente, ela leva em consideração a porta de entrada (*ig_port*) e a presença de uma marca de VLAN (*hdr.vlan_tag.isValid*) no pacote. Se chegar um pacote marcado (*tagged*), o *vlan_id* encontrado será usado para decidir sobre seu descarte (*action deny*) ou não (*action permit*). No caso de chegar um pacote *untagged* numa porta de acesso, a chave *hdr.vlan_tag.vlan_id* será desconsiderada e a *action permit_with_internal_vlan* pode ser chamada para vincular um *vlan_id* ao pacote. A ação padrão (quando não há correspondência de chaves) é negar (descartar) um pacote. Processo similar é usado no pipeline de saída.

Encaminhamento L2

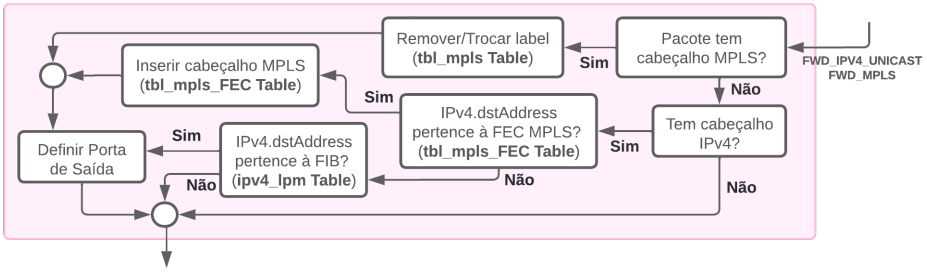
É possível que o pipeline implementado realize 3 tipos de encaminhamento, conforme descrito na figura 4: *Bridging* L2 (*multicast/broadcast* ou *unicast*), IPv4 e MPLS. O tipo de encaminhamento é definido por meio de uma *action*, chamada *set_forwarding_type*, disponível na tabela *fwd_classifier*, que utiliza um parâmetro com um dos seguintes valores: Bridging (0x0); MPLS (0x1) ou IPv4 (0x2). O tipo de encaminhamento padrão é *Bridging*. Assim sendo, recomenda-se o uso de um controlador L2 *Learning* a fim de evitar a inundação (*flooding*) da rede. O



(a) Etapa de Filtragem do Pipeline de Entrada



(b) Mecanismo de Encaminhamento L2



(c) Mecanismo de Encaminhamento L3

Figura 5. Processos de filtragem e encaminhamentos L2/L3 usados pelo DCSG-P4

mecanismo de encaminhamento L2, mostrado na figura 5(b), consiste na análise de duas tabelas. Primeiramente, analisa-se a tabela *mac_forward* em busca da porta de saída vinculada à chave (*key*) *hdr.ethernet.dstAddr* e, caso nenhuma entrada (*entry*) seja compatível (*miss*) com o endereço MAC de destino encontrado na cabeçalho Ethernet do pacote, realiza-se um *brodcast* de acordo com a porta de ingresso (tabela *mac_broadcast*).

Encaminhamento L3

Conforme mostrado na figura 5(c), caso o pacote possua um cabeçalho MPLS, a definição da porta de saída será realizada com base neste protocolo. O *switch* realizará a troca de *label* ou manterá o rótulo atualmente utilizado, caso esteja desempenhando o papel de *Label Switch Router* (*switch* P), ou fará a remoção da *label*, caso esteja na borda da malha MPLS (*Label Edge Router* - *switch* PE). Caso possua apenas cabeçalho IPv4, primeiramente será verificado se o endereço IP de destino pertence a uma *Forwarding Equivalency Class* (FEC) do MPLS. Neste caso, um cabeçalho MPLS será acrescentado antes do envio do pacote para a porta de saída. Caso contrário, o roteamento IP tradicional será realizado.

Em comparação com o pipeline descrito em [Kundel et al. 2019], além de algumas *features* distintas, tivemos o cuidado de criar um mecanismo para a definição

da forma de encaminhamento, já que, como descrito anteriormente, o DCSG consiste num elemento de encaminhamento L2 e L3. Outra característica de nosso pipeline, não encontrada em outros trabalhos relacionados, diz respeito ao mapeamento de campos QoS usados pelos diferentes protocolos considerados em nosso projeto (ETH - CoS; MPLS - ExpBit; IPv4 - DSCP).

4. Resultados Obtidos

4.1. A Topologia de Testes

O código *dcsg.p4* foi implementado de tal forma que os conceitos apresentados neste trabalho sejam efetivamente validados. Portanto, uma arquitetura com dois *switches* Tofino foi desenvolvida e implementada na infraestrutura do laboratório, conforme demonstrado na figura 6. Ambos *switches* TOFINO são do modelo Edgecore Wedge 100BF-32X, denominados T1 e T2, conectados ao servidor "Jambu". Este servidor é equipado com processador AMD EPYC 7313 de 32 núcleos em 3 Ghz, 32 GB de memória RAM e sistema operacional Ubuntu 20.04.1 LTS x86_64, com kernel versão 5.4.0-170-generic. O emulador utilizado nos testes foi o EVE-NG v5.0.1-19-Community e o roteador legado foi emulado neste ambiente, executando o Software Cisco IOS XR Versão 6.0.1.

O gerador de tráfego PIPO-TG [Costa et al. 2024] foi executado no Tofino T1 para fornecer o fluxo de pacotes que atravessa toda a malha composta. O PIPO-TG, ferramenta desenvolvida pelo mesmo grupo de pesquisa, cria pacotes a partir do gerador interno do Tofino, possibilitando a transferência de dados em altas taxas.

A parte referente à rede de Acesso, composta por *switches* de *software* que executam o pipeline DCSG apresentado na seção 3 e usam a arquitetura V1Model (BMv2), foi configurada para realizar apenas o encaminhamento IPv4. A malha MPLS fornece um cenário mais heterogêneo, composto por *switches* de *software* e de *hardware*, além de um equipamento legado CISCO IOS XR. Os *switches* PE-S1 e PE-S7 estão configurados como *Provider Edge* e, como tal, fazem a

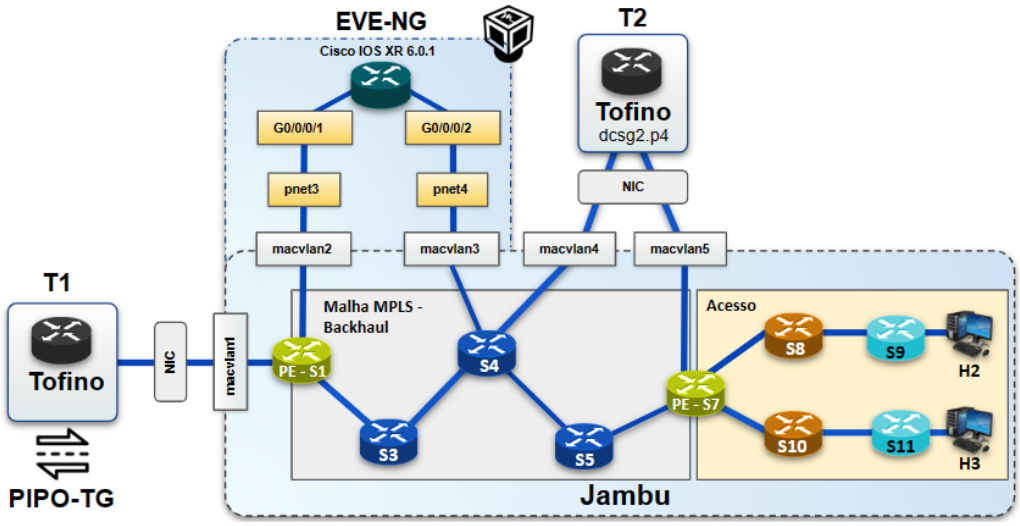


Figura 6. Ambiente de testes criado e utilizado neste trabalho

inserção e a remoção de cabeçalhos MPLS, enquanto os demais *switches* dentro da malha (*switches* P) fazem a troca de rótulos MPLS (*swap label*) quando necessário.

Os *switches* de *software* BMv2 (emulados no servidor Jambu) e o Tofino ASIC executam instâncias compiladas do código *dcsg.p4*, enquanto o equipamento legado Cisco teve a configuração MPLS realizada via CLI (*Command Line*) do IOS XR.

O resultado da compilação do programa no Tofino ASIC, em comparação com um *switch* tradicional de referência, apresentou taxas de utilização dos diferentes recursos de *hardware* bem próximas aos mínimos previstos, indicados na coluna baseline (%). A tabela 2 apresenta os resultados para a compilação do pipeline para o Tofino. Esta utilização está relacionada, apenas, à execução da lógica funcional.

Tabela 2. Resultado da compilação do pipeline para *hardware* Tofino

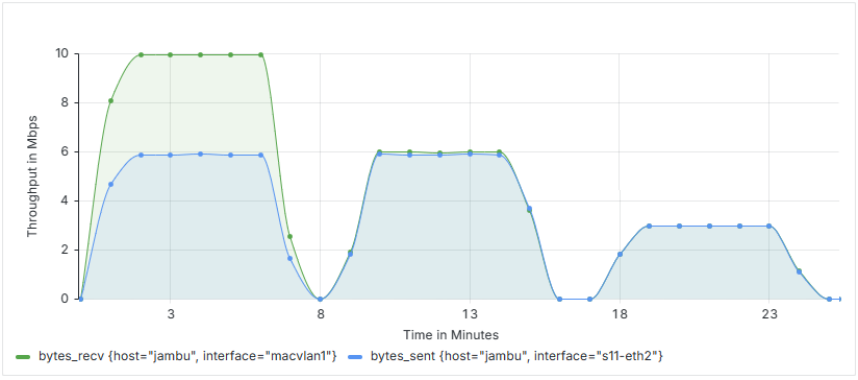
Recurso	**baseline** (%)	**dcsg.p4** (%)	**uso adicional (%)**
Exact Match Input Xbar	16,9	18,1	1,2
Hash Bit	20,1	22,5	2,4
SRAM	23,6	25,3	1,7
TCAM	21,9	22,9	1,0
VLIW Instruction	12,8	15,4	2,6
Exact Match Result Bus	18,8	20,4	1,6

4.2. Testes executados

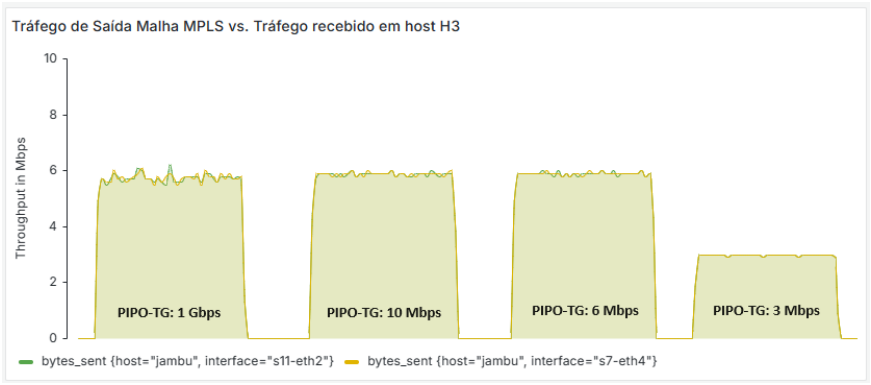
A migração de uma arquitetura legada para uma rede puramente SDN pode ser inviável no curto prazo do ponto de vista de CAPEX e de OPEX [Ollora Zaballa et al. 2021]. Em virtude disso, a interoperabilidade de equipamentos legados, que usam protocolos como o MPLS, com dispositivos de redes programáveis em linguagens como o P4 é indispensável para adequação da infraestrutura de redes de comunicações aos moldes dos parâmetros da geração 5G e além. Assim, os testes conduzidos neste trabalho visam corroborar com esta premissa. Em nosso pipeline, o MPLS foi configurado por meio do uso das tabelas *tbl_mpls_FEC* e *tbl_mpls*, conforme mostrado anteriormente na figura 5(c). Neste quesito de interoperabilidade, os pacotes MPLS enviados dos *switches* BMv2 para o roteador Cisco, assim como aqueles que transitaram no sentido inverso, foram processados de forma transparente em ambos elementos de rede.

Conforme [Kundel et al. 2019], a implementação de funções de rede na arquitetura BMv2 terá um desempenho, na melhor das hipóteses, semelhante ao kernel Linux em termos de taxa de transferência, perda de pacotes e latência. Assim sendo, para geração de tráfego, utilizamos quatro taxas de transmissão em nossos testes: 1 Gbps, 10 Mbps, 6 Mbps e 3 Mbps. Estes valores foram escolhidos a fim de encontrar a limitação de máximo *throughput* da máquina hospedeira do ambiente emulado.

Iniciamos os testes com o tráfego gerado à taxa de 1 Gbps e percebemos que a taxa de entrega ficou limitada aos 6 Mbps, o mesmo aconteceu com o fluxo de pacotes sendo gerado à taxa de 10 Mbps. Para 6 Mbps nosso pipeline conseguiu realizar a entrega na mesma vazão, fato também observado quando reduzimos a taxa de geração de dados para 3 Mbps.



(a) Tráfego de Entrada vs. Tráfego de Saída



(b) Tráfego de saída de malha MPLS vs. tráfego recebido em host H3

Figura 7. Resultados obtidos

Assim, no caso do *testbed* configurado para este trabalho, os valores de *throughput* medidos nos nós emulados ficaram limitados a 6 Mbps, conforme apresentado em 7(a). Para efeitos demonstrativos, este gráfico considerou as taxas de 10 Mbps, 6 Mbps e 3 Mbps; e os valores plotados representam a média das medições num intervalo de 1 minuto. As medições foram realizadas durante um intervalo de cerca 5 minutos para cada uma das taxas mencionadas e foram executadas nas interfaces da entrada do servidor Jambu (macvlan1) e na saída do último *switch* emulado S11 (s11-eth2). A linha verde representa o tráfego de entrada, enquanto a linha azul sinaliza o *throughput* recebido no destino.

A análise das medições puras (sem o cálculo da média) nas interfaces s7-eth4 e s11-eth2, mostrou uma diferença na oscilação do valor máximo do *throughput*, conforme ilustrado na figura 7(b). Neste caso, o intervalo de tempo entre as medições plotadas no gráfico é de 10 s. Para a geração de tráfego à taxa de 1 Gbps, o *throughput*, observado na rede de Acesso da topologia, variou entre 5,5 Mbps e 6,2 Mbps, enquanto que o tráfego gerado à 3 Mbps se comportou de maneira mais estável, variando entre 2,9 Mbps e 3,0 Mbps.

O *testbed* apresentado pode ser reproduzido com relativa facilidade, já que as instâncias emuladas dos *switches* de *software* (BMv2), assim como o equipamento Cisco emulado no EVE-NG, podem ser plenamente reproduzidos em outros ambientes. A única limitação visualizada para reprodutibilidade do cenário aqui

apresentado está relacionada com a disponibilidade de um *hardware* Tofino. Contudo, esta limitação pode ser contornada se for considerado o uso de uma máquina virtual que execute uma instância do Tofino Model.

5. Considerações Finais

Este trabalho descreve a implementação de algumas funcionalidades especificadas pelo TIP para um DCSG utilizado como Roteador de Borda por um provedor de serviços. Essa implementação foi realizada por meio de técnicas de programação de planos de dados com base na linguagem P4.

Os resultados obtidos demonstram o sucesso operacional das funcionalidades implementadas, que incluem o encaminhamento L2 (Ethernet) com ou sem marcação de VLAN/QinQ, assim como o encaminhamento L3 por meio de IPv4 ou MPLS. Os testes funcionais, com tráfego encaminhado tanto pelos *switches* de *software* quanto pelo Tofino foram bem-sucedidos. Além disso, observou-se com êxito a interoperabilidade ao integrar um dispositivo legado (Cisco IOS XR) no ambiente de testes construído para este propósito.

Observamos também que nossa solução conseguiu lidar com volumes significativos de tráfego, sendo que o *throughput* foi limitado pelo servidor onde os *switches* emulados estavam hospedados.

Como perspectiva de trabalho futuro, sugere-se a extensão da solução apresentada para incluir suporte ao TSN (*Time-Sensitive Networking*) por meio do IEEE 1588 *Precision Time Protocol* (PTP), conforme previsto nas especificações do TIP já mencionadas anteriormente. Isso permitiria a realização de testes em cenários *cell site*. Consideramos relevante também a avaliação da escalabilidade das funcionalidades desenvolvidas, por meio da saturação das tabelas *match-action*.

6. Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, da Padtec S/A e do Instituto Federal do Sul de Minas (IFSULDEMINAS).

Referências

Abhishek Singh, J., Sachin Kumar, M. R., and Shushrutha, K. S. (2021). Implementation of segment routing-traffic engineering over mpls. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–5.

Atrio, J. A. G., Morillo, M. J. L., Agabio, P., Garcia, L. M., Moretón, D. M., Sutton, A., López, V., Diallo, M. B., and Palmeira, S. F. (2019). Disaggregated cell site gateway: Technical specification v1.1. Disponível em: `https://cdn.mediavalet.com/usva/telecominfraproject/fSlhAvf4P0iflU8O-5Z8Cw/kMI43F8h10SP9DANVm8x8A/Original/DCSG_Technical_Specification_-_Telecom_Infra_Project.pdf`. Acesso em: 06/10/2023.

Bahnasse, A., Louhab, F. E., Ait Oulahyane, H., Talea, M., and Bakali, A. (2018). Novel sdn architecture for smart mpls traffic engineering-diffserv aware management. *Future Generation Computer Systems*, 87:115–126.

Chaudhary, A. (2020). Niral open product framework. *Telecom Business Review*, 13(1):53.

Costa, F. G., Vogt, F., Cesen, F. E. R., de Castro, A. G., Luizelli, M. C., and Rothenberg, C. E. (2024). Pipo-tg: Parameterizable high-performance traffic generation. In *37th IEEE/IFIP Network Operations and Management Symposium (NOMS), Seoul, South Korea, Maio 2024*.

Depasquale, E.-V., Tinka, M., Zammit, S., and Davoli, F. (2023). A survey of trends and motivations regarding communication service providers' metro area network implementations. *arXiv preprint arXiv:2309.11969*.

Foundation, O. N. (2024). Sdn definition. Disponível em: `https://www.opennetworking.org/sdn-resources/sdn-definition`. Acesso em: 06/10/2023.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Kundel, R., Nobach, L., Blendin, J., Kolbe, H.-J., Schyguda, G., Gurevich, V., Koldehofe, B., and Steinmetz, R. (2019). P4-bng: Central office network functions on programmable packet pipelines. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–9.

Liu, Z., Lv, G., Wang, J., and Yang, X. (2022). Domain-specific programming router model. In *International Conference on Emerging Networking Architecture and Technologies*, pages 26–37. Springer.

Lu, G., Guo, C., Li, Y., Zhou, Z., Yuan, T., Wu, H., Xiong, Y., Gao, R., and Zhang, Y. (2011). {ServerSwitch}: A programmable and high performance platform for data center networks. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*.

Marcom, E. N. (2021). Edgecore dcsg - csr200 (as5915-18x) introduction. Disponível em: `https://www.youtube.com/watch?v=nxz3l5DGwP4`. Acesso em: 06/10/2023.

Ollora Zaballa, E., Franco, D., Thomsen, S. E., Higuero, M., Wessing, H., and Berger, M. S. (2021). Towards monitoring hybrid next-generation software-defined and service provider mpls networks. *Computer Networks*, 191:107960.

Org, P. (2024). P4 open source programming language. `https://p4.org/`. Acesso em: 09/01/2024.

Rosa, R., Siqueira, M., Barea, E., Marcondes, C., and Rothenberg, C. (2014). Network function virtualization: Perspectivas, realidades e desafios. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Rosen, E., Viswanathan, A., and Callon, R. (2001). Multiprotocol label switching architecture. Technical report.

Steinert, B., Häberle, M., Nick, J.-O., Farinacci, D., and Menth, M. (2023). P4-lisp: A p4-based high-performance router for the locator/identifier separation protocol. In *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, pages 89–97. IEEE.