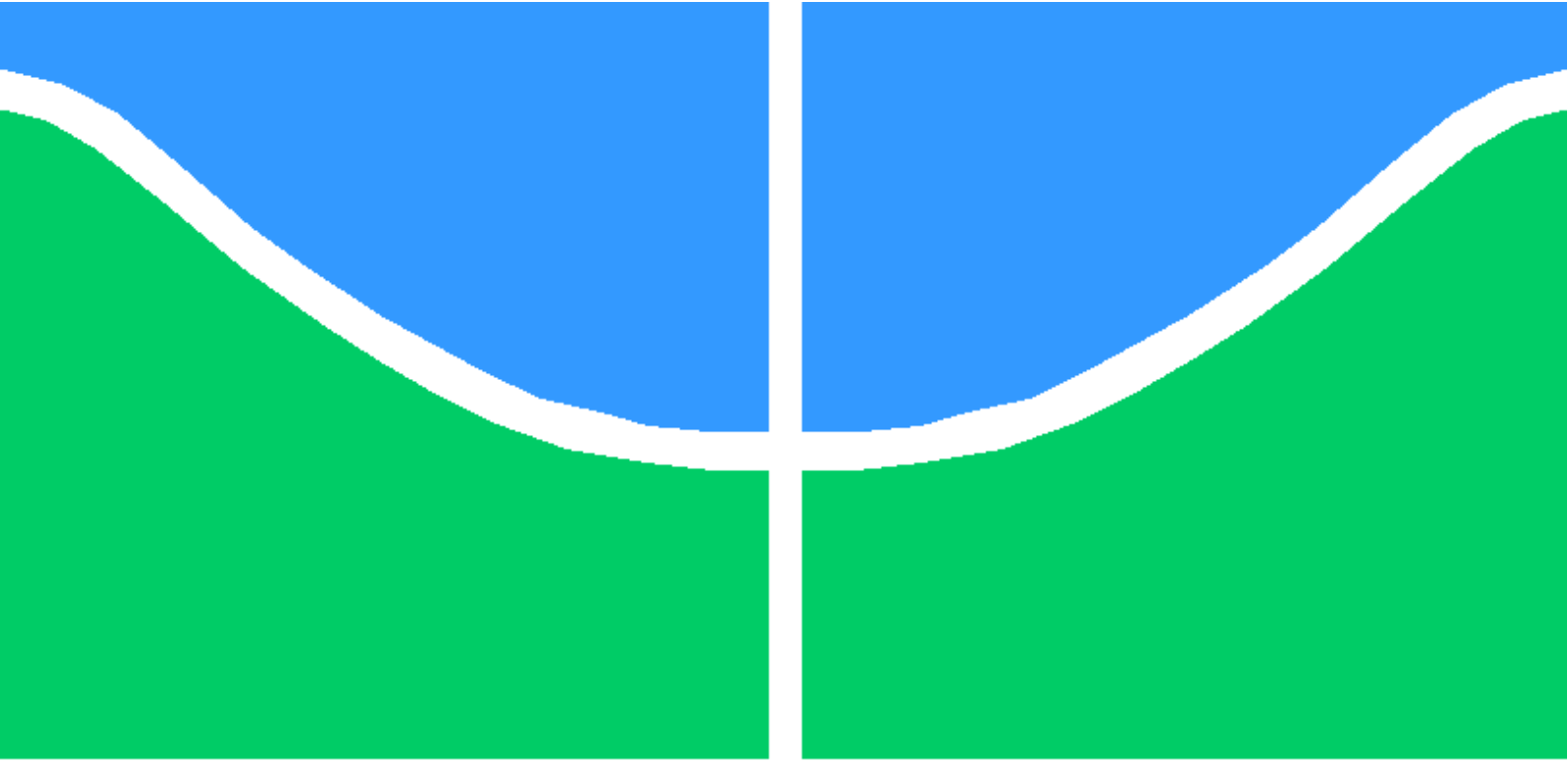

DISSERTAÇÃO DE MESTRADO

ARQUITETURA E PROTÓTIPO DE UMA REDE SDN-OPENFLOW PARA PROVEDOR DE SERVIÇO

Fernando López Rodríguez

Brasília, maio de 2014

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ARQUITETURA E PROTÓTIPO DE UMA REDE SDN-OPENFLOW PARA PROVEDOR DE SERVIÇO

FERNANDO LÓPEZ RODRÍGUEZ

ORIENTADOR: DIVANILSON RODRIGO DE SOUSA CAMPELO

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGEE.DM - 562/14

BRASÍLIA/DF: MAIO DE 2014

FICHA CATALOGRÁFICA

LÓPEZ RODRÍGUEZ, FERNANDO	
Arquitetura e Protótipo de uma Rede SDN-OpenFlow para Provedor de Serviço [Distrito Federal] 2014.	
59p., 210 x 297 mm (ENE/FT/UnB, Mestre, Dissertação de Mestrado – Universidade de	
Brasília. Faculdade de Tecnologia.	
Departamento de Engenharia Elétrica	
1. Provedor de Serviço	2.OpenFlow
3. Robustez	4. Velocidade de encaminhamento
5. Sobrecarga do controlador	6. Dependência ao controlador
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

LOPEZ., F (2014). Arquitetura e Protótipo de uma Rede SDN-OpenFlow para Provedor de Serviço. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGEE.DM-562/14, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 59p.

CESSÃO DE DIREITOS

AUTOR: Fernando López Rodríguez.

TÍTULO: Arquitetura e Protótipo de uma Rede SDN-OpenFlow para Provedor de Serviço.

GRAU: Mestre ANO: 2014

_____

Fernando López Rodríguez
CPF:701.290.151-99
RG:D20705-01

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

ARQUITETURA E PROTÓTIPO DE UMA REDE SDN-OPENFLOW PARA PROVEDOR DE SERVIÇO

Fernando López Rodríguez

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Divanilson Rodrigo de Sousa Campelo, CIn/UFPE e PPGEE/UnB _____
Orientador

Prof. Gustavo Sousa Pavani, UFABC _____
Examinador externo

Prof. William Ferreira Giozza, ENE/UnB _____
Examinador interno

Dedicatória

A minhas filhas Sofía Ema e María Eugenia.

Fernando López Rodríguez

Agradecimentos

À minha esposa e filhas, por sua compreensão e apoio durante as longas jornadas de estudo.
À minha família toda que são grandes motivadores para minha carreira.
Ao Professor Doutor Divanilson Rodrigo de Sousa Campelo que me orientou e motivou durante todo o processo.
Ao Professor Doutor Ugo Silva Dias, por seus importantes conselhos e sugestões.
Aos funcionários do Departamento de Engenharia Elétrica, pelo apoio nas necessidades administrativas durante o Mestrado.
Ao Brasil, que me deu a possibilidade de me especializar em esta importante universidade, sendo aluno estrangeiro.

Fernando López Rodríguez

RESUMO

As redes de grande porte, como Provedores de Serviços, são arquiteturas robustas, capazes de dar suporte a grandes volumes de tráfego com características muito diferentes. Seus equipamentos dão suporte a cargas elevadas de processamento e são, ao mesmo tempo, responsáveis por construir a lógica de roteamento e por encaminhar o tráfego. Por terem o controle implementado de forma distribuída e por serem construídas com equipamentos de um limitado número de fabricantes, estas redes apresentam limitações de controle e engenharia de tráfego, dificultando, assim, a diferenciação entre os serviços que os diversos provedores fornecem. Adicionalmente, a inteligência da rede está oculta nos equipamentos, tornando às inovacões muito lentas e amarradas aos interesses dos fabricantes. Como alternativa a este cenário, este trabalho propõe uma arquitetura de rede SDN-OpenFlow que tenta solucionar os problemas previamente mencionados, bem como os inconvenientes que a característica centralizadora de OpenFlow possui. É apresentada uma arquitetura de rede OpenFlow robusta, capaz de dar suporte a tempos de resposta elevados e a quedas do controlador, sem adição de tempos de espera no estabelecimento de novos fluxos e com significativa redução na carga submetida ao controlador. Como prova de conceito, é implementado um protótipo utilizando o Open vSwitch como software para a virtualização dos clientes OpenFlow, o Mininet para a criação da topologia e o Ryu como controlador, todos com suporte a OpenFlow 1.3.

ABSTRACT

Large scale networks, such as Service Providers, are robust architectures, capable of supporting large volumes of traffic with very different characteristics. Their network equipment have significant processing load, being responsible for building both a routing logic and routing traffic at the same time. By having the network control implemented in a distributed manner and being built with a limited number of vendors, these networks have limitations of control and traffic engineering, hindering the differentiation between Services Providers. Additionally, the network intelligence is hidden in the network equipment, making the innovations very slow and conditioned to the vendors interests. As an alternative option, this work proposes an SDN-OpenFlow network architecture that tries to improve the previously mentioned problems, and at the same time solves the arising difficulties related to the SDN network centralizing feature. With the proposed architecture, a robust OpenFlow network is created to support high controller response times and controller shut down, without additional delays in the creation of flows and with significant reduction of controller's load. A prototype has been constructed using Open vSwitch as a virtualization software for OpenFlow clients, Mininet for the topology construction and Ryu as the controller, all with OpenFlow 1.3 support.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

ACL	Access Control List
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
DSCP	Differentiated Services Code Point
EXP	Experimental Bits
FEC	Forwarding Equivalent Class
FIB	Forwarding Information Base
HSI	High Speed Internet
ISIS	Intermediate System to Intermediate System
ISIS-TE	ISIS - Traffic Engineering
IP	Internet Protocol
IPv6	Internet Protocol Version 6
LAN	Local Area Network
LER	Label Edge Router
LDP	Label Distribution Protocol
LSR	Label Switch Routers
MPLS	Multiprotocol Label Switching
MPLS-TE	MPLS - Traffic Engineering
MPLS-VPN	MPLS - Virtual Private Network
OSPF	Open Shortest Path First
OSPF-TE	OSPF - Traffic Engineering
POP	Points of Presence
QoS	Quality of Service
RIB	Routing Information Base
RSVP	Resource Reservation Protocol
SDN	Software Defined Network
SP	Service Provider
STP	Spanning Tree Protocol

TCP	Transport Control Protocol
TDM	Time Division Multiplexing
TLS	Transport Layer Security
ToS	Type of Service
TTL	Time to Live
VC	Virtual Circuits
VLAN	Virtual Local Area Network
VoIP	Voice over IP
VPN	Virtual Private Network
WDM	Wavelength Division Multiplexing

Capítulo 1

Introdução

1.1 Definição do problema

Provedores de serviços (*service providers*, SP) são organizações que comercializam acesso à Internet tanto para empresas e usuários finais, como para outros provedores. Além dos serviços de acesso, cada vez mais os provedores oferecem uma diversidade de serviços associados, como centro de dados, serviços de VoIP (*Voice over IP*), televisão digital e interconexões privadas, entre outros. Por estas razões, os provedores devem transportar uma grande variedade de serviços com requisitos de QoS (*Quality of Service*) diferentes, os quais, adicionados ao grande volume de tráfego e à dimensão das redes dos provedores, tornam o gerenciamento destas redes uma tarefa complexa.

Em termos arquitetônicos, as redes de SP possuem mecanismos de controle automáticos e totalmente distribuídos, tais como protocolos de roteamento (por ex., *Border Gateway Protocol* – BGP, *Open Shortest Path First* – OSPF, *Intermediate System to Intermediate System* – ISIS) e de sinalização (por ex., *Label Distribution Protocol* – LDP, *Resource Reservation Protocol* – RSVP). Estes mecanismos de controle geram arquiteturas robustas, capazes de recalcular automaticamente sua topologia com a queda de qualquer um dos seus equipamentos. Porém, os mecanismos levam a equipamentos complexos, responsáveis por construir tabelas de roteamento e pelo encaminhamento de tráfego ao mesmo tempo, fazendo com que tipicamente tenham cargas de processamento elevadas nestas redes.

Como características adicionais, as redes de SP são caras em infraestrutura e operação, tendo que manter grupos de administradores diferentes para cada tipo de redes como IP (*Internet Protocol*) e WDM (*Wavelength Division Multiplexing*). Os equipamentos utilizados são de um grupo reduzido de fabricantes e possuem um número de funcionalidades limitadas reduzindo sensivelmente as possibilidades de diferenciação e engenharia de tráfego. A "inteligência" da rede está oculta nos equipamentos, tornando as inovações na rede extremamente lentas e amarradas aos interesses dos fabricantes.

O caminho alternativo às redes com o controle distribuído (redes atuais) são as redes definidas por software (*software defined networks*, SDN), que têm como proposta separar o controle das redes para um dispositivo denominado controlador. Dentro destas encontra-se o protocolo OpenFlow

[McKeown et al., 2008] [OpenFlow v.1.3 , 2012] [Limoncelli, 2012], que proporciona uma arquitetura centralizada com uma interface aberta e padronizada, através da qual equipamentos clientes podem interagir com um controlador dotado de uma visão global da rede e encarregado de construir as tabelas de fluxo para o roteamento de pacotes em todos os clientes. Dessa forma, é possível separar o plano de dados (clientes OpenFlow) e de controle (controlador). O protocolo OpenFlow proporciona inúmeras possibilidades para engenharia de tráfego [Egilmez et al., 2011] [Das et al., 2011] [Agarwal et al., 2013] [Chial Sanchez, 2013], unificação do plano de controle para diferentes tipos de redes, como IP e WDM [Das, 2012] [Shirazipour et al., 2012] [Das et al., 2010] [Giorgetti et al., 2012], computação em nuvem [Bakshi, 2013] [Matias et al., 2011], gerenciamento de mobilidade [Bifulco et al., 2012] [Namal et al., 2013], entre outras. Tais possibilidades têm atraído o interesse de diferentes fabricantes de redes, os quais já estão incorporando OpenFlow em seus produtos [SDN Product Directory,].

No entanto, apesar das vantagens indicadas no parágrafo anterior, Openflow possui desvantagens [Benton et al., 2013]. Particularmente para um SP, a arquitetura OpenFlow padrão parece não ser apropriada em razão de:

- sua característica centralizadora, que gera dependência excessiva no controlador, o que reduz em grande medida a robustez da rede;
- o requisito de que cada pacote de um novo fluxo seja processado e encaminhado pelo controlador, gerando assim possível sobrecarga neste;
- degradação de desempenho, pois cada fluxo tem que aguardar a resposta do controlador para seu roteamento.

1.2 Objetivos do trabalho

Este trabalho propõe uma arquitetura SDN-OpenFlow robusta, capaz de dar suporte a tempos de resposta elevados e a quedas do controlador, sem adição de tempos de espera no estabelecimento de novos fluxos e com significativa redução na carga submetida ao controlador para o gerenciamento de tráfego em SP, com e sem QoS. Ao mesmo tempo é capaz de aprimorar as limitações que as redes atuais com controle distribuído têm como limitações na engenharia de tráfego, inovações amaradas aos interesses dos fabricantes, equipamentos de rede com grande carga em seu processamento. Adicionalmente, é apresentado um protótipo que implementa a lógica proposta como prova de conceito.

1.3 Apresentação do manuscrito

O restante deste trabalho é organizado conforme descrito a seguir:

- O Capítulo 2 descreve o caso de uso escolhido e apresenta uma visão geral de SDN e OpenFlow.

- O Capítulo 3 aborda a arquitetura proposta no trabalho, com destaque para a lógica geral de funcionamento, os mecanismos proativos que permitem alcançar todas as redes destino para encaminhamento interno e MPLS (*Multiprotocol Label Switching*), e a lógica proposta para o gerenciamento do tráfego QoS, que gera uma rede robusta e sem adição de tempos de espera na criação de novos fluxos. Tambem é sugerida uma arquitetura alternativa com equipamentos híbridos, capazes de implementar OpenFlow e o controle distribuído clássico simultaneamente.

- O Capítulo 4 apresenta resultados do cálculo analítico do atraso das implementações que utilizam OpenFlow padrão.

- O Capítulo 5 tem como objetivos descrever os *softwares* utilizados, mostrar o protótipo desenvolvido, os diferentes testes realizados e os principais resultados do trabalho mediante uma tabela comparativa entre a arquitetura proposta com equipamentos puramente OpenFlow, a arquitetura proposta com equipamentos híbridos, uma rede configurada com OpenFlow padrão e uma rede com controle distribuído clássico.

- O Capítulo 6 apresenta as principais conclusões do trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Caso de Uso e Conceitos Prévios

2.1 Visão geral de uma rede de Provedor de Serviço

Os provedores de serviço são os blocos fundamentais na arquitetura da Internet. A interconexão entre SPs em conjunto com o protocolo de roteamento global BGP (*Border Gateway Protocol*) e protocolos de roteamento interno permitem a troca de informações entre quaisquer pontos do mundo através da Internet. Os SPs são redes de grande dimensão que oferecem um número amplo de opções de conectividade aos seus clientes, usualmente com a oferta de serviços de VoIP, vídeo e telefonia celular, entre outros. A Fig. 2.1 ilustra as possíveis caracteristicas de um SP, listadas a seguir:

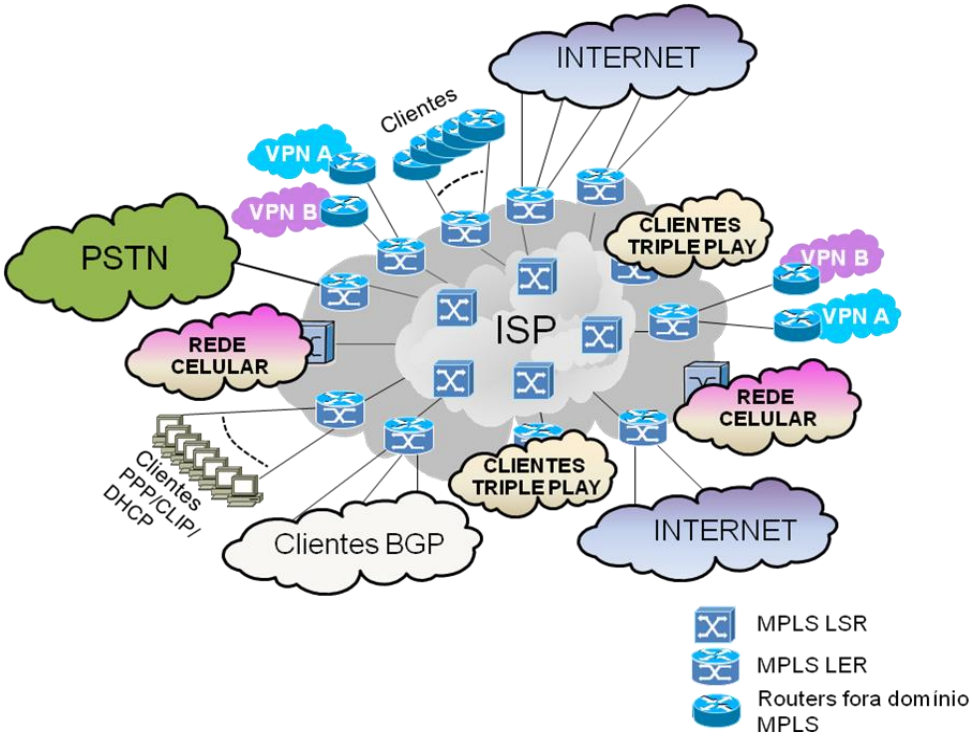

Figura 2.1: Visão geral de uma rede de Provedor de Serviço

- consta de um número significativo de equipamentos interconectados mediante diferentes tipos de conexões, como *Ethernet*, SDH/SONET, ATM (*Asynchronous Transfer Mode*), *Frame Relay*, WDM, etc;
- utiliza protocolo de roteamento de estado de enlace como ISIS (*Intermediate System to Intermediate System*) ou OSPF (*Open Shortest Path First*);
- tem implementado MPLS (*Multiprotocol Label Switching*) em todo o backbone da sua rede, com equipamentos LER (*Label Edge Router*) e equipes puramente LSR (*Label Switch Router*);
- utiliza a funcionalidade MPLS-TE (*Multiprotocol Label Switching - Traffic Engineering*), para realizar engenharia de tráfego;
- presta serviço de VPN mediante MPLS-VPN (*Multiprotocol Label Switching - Virtual Private Network*);
- dependendo do tipo de cliente (residencial ou empresarial) presta serviço de conectividade à Internet por meio de diversas tecnologias como *Frame Relay*, ATM, xDSL (*Digital Subscriber Line*), TDM (*Time Division Multiplexing*), *dial-up*, rede celular ou acesso óptico;
- tem equipamentos na localidade do cliente, que poderá ser administrado por ele ou não;
- possui clientes de considerável porte, interconectados mediante sessões BGP;
- possui diversas conexões a diferentes Tier1 (redes de trânsito que interconectam os ISPs), administradas mediante sessões BGP, proporcionando-lhes conectividade global;
- oferece conectividade IP à sua rede celular compartilhando seu backbone IP;
- interconecta a rede de telefonia básica (PSTN) ao backbone IP;
- dá suporte ao serviço *triple play* (vídeo, VoIP e HSI) aos clientes que o solicitarem.

Pode-se notar que as características anteriormente descritas são comuns a um grande número de SPs, dão uma ideia da diversidade de serviços e tipos de tráfego que os SPs possuem.

2.2 SDN

As redes definidas por software (SDN) são uma abordagem para o gerenciamento de redes de computadores que evoluiu a partir de trabalho realizado na Universidade de Berkeley e Stanford no ano 2008. As SDN permitem que os administradores de rede gerenciem serviços de redes através da abstração das funcionalidades de nível inferior. Isto é conseguido através da dissociação do plano de controle (onde são construídas as decisões de roteamento) e o plano de dados (o nível mais baixo composto pelos dispositivos físicos responsáveis de routear o tráfego). Nas SDN, o plano de controle é trasladado para um dispositivo central denominado controlador, que possui a visão global da rede e é responsável por configurar o plano de dados dos dispositivos de redes

mediante um protocolo padrão aberto. Adicionalmente, o controlador permite a criação de uma camada de alto nível de abstração (camada SDN), que permite a programação dos serviços a serem configurados na rede (ver Fig. 2.2).

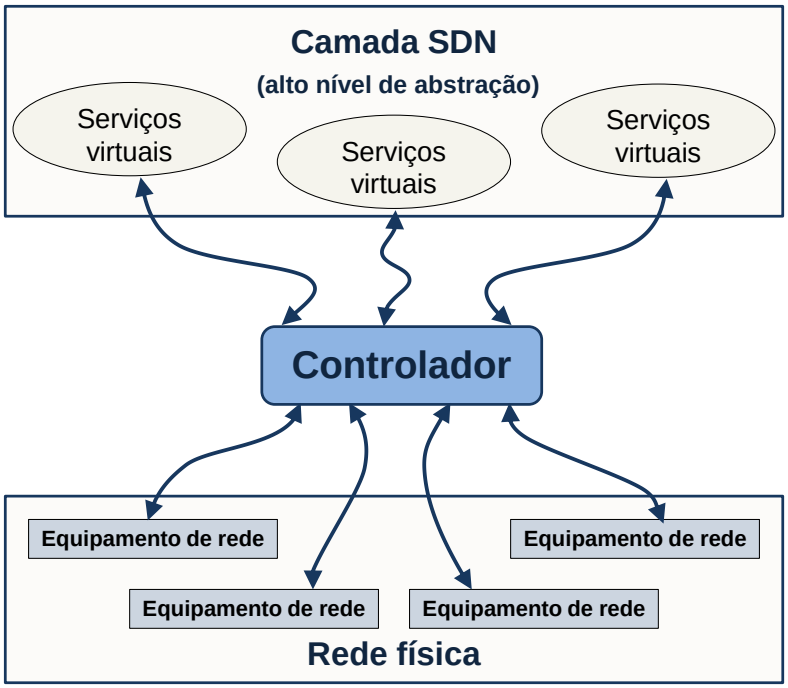

Figura 2.2: Arquitetura SDN

Para uma melhor comprenção do conceito de SDN, é mostrada a arquitetura interna atual de um roteador e quais as mudanças que SDN propõe.

A arquitetura atual de um roteador (ver Fig. 2.3) pode ser dividida em duas partes:

- o plano de dados: está geralmente implementado em hardware em que se encontra a tabela de fluxos denominada FIB (*Forwarding Information Base*) dos dispositivos de rede. Esta tabela é constituída pela informação necessária para identificar univocamente os fluxos e as ações a seguir para cada um deles (por exemplo, sair por determinada porta);
- o plano de controle: é implementado em software e é o lugar onde se encontram a lógica de roteamento, isto é, o protocolo de roteamento, e a tabela construída por ele, denominada RIB (*Routing Information Base*).

Cada primeiro pacote de um novo fluxo que chega a um roteador é inicialmente enviado ao plano de controle, e este decide o roteamento e a ação a ser tomada. Esta informação é colocada como uma nova entrada na tabela de fluxos (FIB) dentro do plano de dados. Com isto, no caso de chegar outro pacote desse mesmo fluxo, é o plano de dados quem encaminha o pacote diretamente, sem a necessidade de gerar uma nova consulta ao plano de controle, agilizando em grande medida o roteamento.

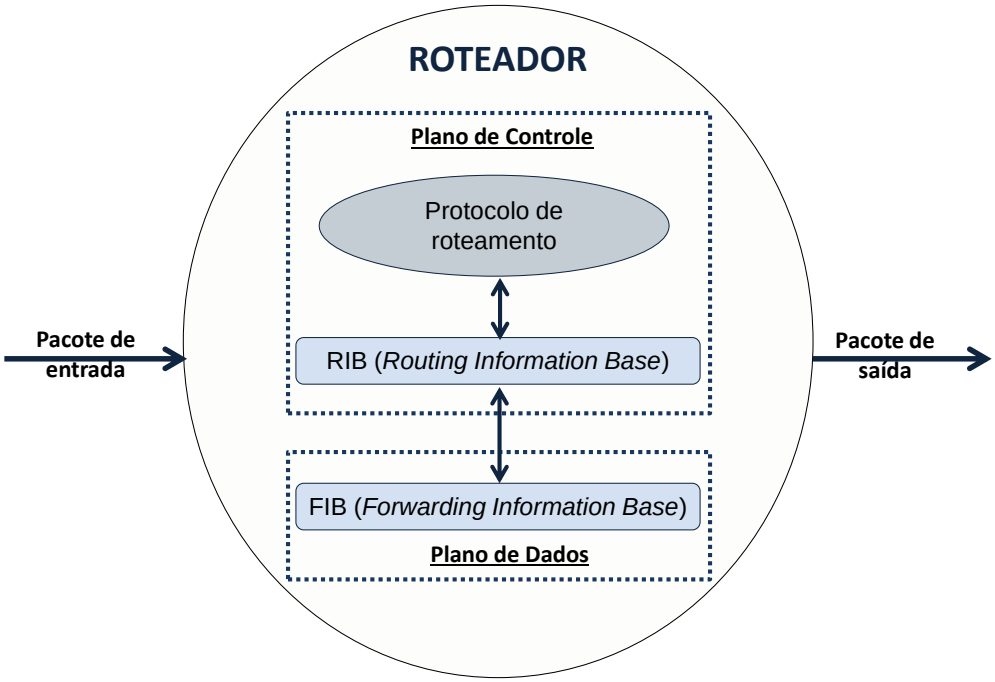

Figura 2.3: Arquitetura de um roteador

A idéia de uma SDN [SDN Definition,] é que os dispositivos de rede só tenham o plano de dados previamente descrito, diminuindo sensivelmente a complexidade destes dispositivos, e que o plano de controle esteja em um novo elemento de rede, denominado controlador. O controlador possui a visão e o controle global da rede, gerando, assim, inúmeras novas possibilidades de controle.

Para uma melhor compreensão de uma SDN, considere a situação descrita na Fig 2.4. Quando o primeiro pacote de um novo fluxo chegar a um dispositivo de rede, este ainda não tem pré-configurado um fluxo em seu plano de dados que lhe indique como encaminhar o pacote. Então, o dispositivo de rede envia uma consulta ao controlador, que calcula o melhor caminho para o fluxo específico. Com esta informação e mediante um protocolo de comunicação apropriado, o controlador configurará o fluxo específico no plano de dados de todos os equipamentos intervenientes no trajeto, possibilitando, assim, o roteamento do pacote. É importante notar que os sucessivos pacotes do mesmo fluxo, agora já têm pré-configurado o fluxo específico no plano de dados e, portanto, não terão que realizar uma nova consulta ao controlador.

Para que o funcionamento de uma SDN seja possível, é necessário que exista um protocolo de comunicação entre os diferentes elementos de rede e o controlador, de forma que este último seja capaz de manipular as tabelas de fluxos dos elementos de redes. O protocolo mais famoso e com estágio de desenvolvimento mais avançado com essa finalidade é o protocolo aberto OpenFlow [McKeown et al., 2008] [OpenFlow v.1.3 , 2012], escolhido no presente trabalho. OpenFlow proporciona uma interface aberta padrão para que a comunicação seja possível.

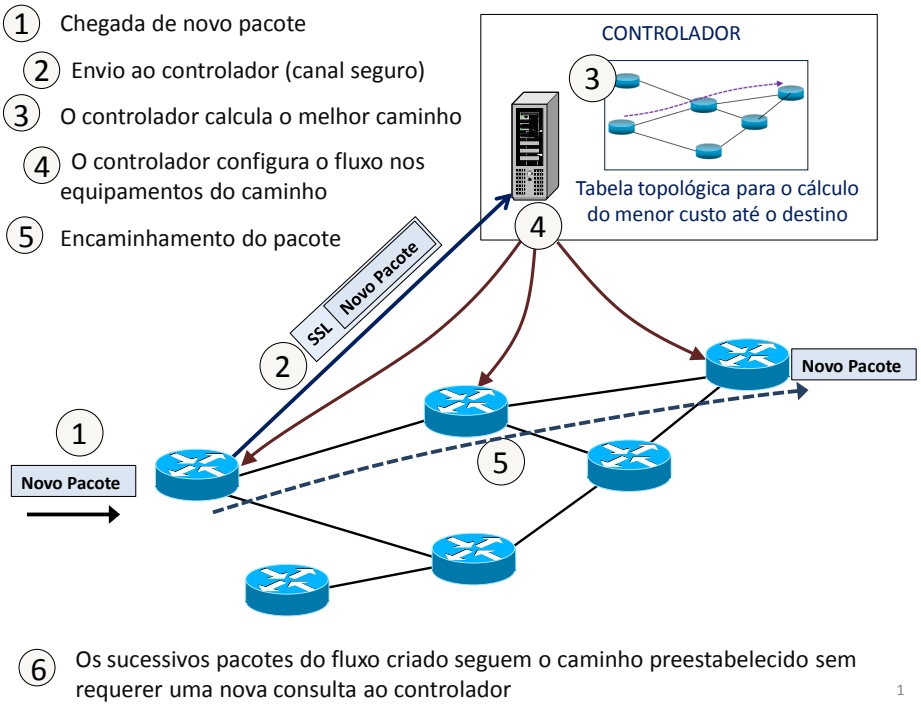

Figura 2.4: Modo de funcionamento de uma SDN

2.3 Protocolo OpenFlow

2.3.1 Características básicas

Um elemento de rede que implementa OpenFlow, conforme ilustrado na Fig. 2.5, tem que satisfazer três características fundamentais:

- ter uma tabela de fluxos e, para cada fluxo, uma ação a tomar;
- opcionalmente um canal TCP ou um canal seguro, que será utilizado para toda a comunicação entre os elementos de rede e o controlador OpenFlow;
- utilizar o protocolo OpenFlow para toda a comunicação previamente mencionada, o que proporcionará uma interface padrão que permitirá a comunicação entre os elementos de rede (de qualquer fabricante que o implemente) e o controlador.

Para cada fluxo entrante, os *switches* OpenFlow devem executar no mínimo três ações básicas:

- encapsular e reenviar o pacote ao controlador por meio de um canal seguro. Isto será tipicamente realizado para todos os pacotes que ainda não têm uma regra específica preenchida na tabela de fluxos;
- encaminhar o pacote mediante uma regra pré-estabelecida na tabela de fluxos. Este é o caso geral, uma vez que o controlador configura o fluxo no equipamento de rede;

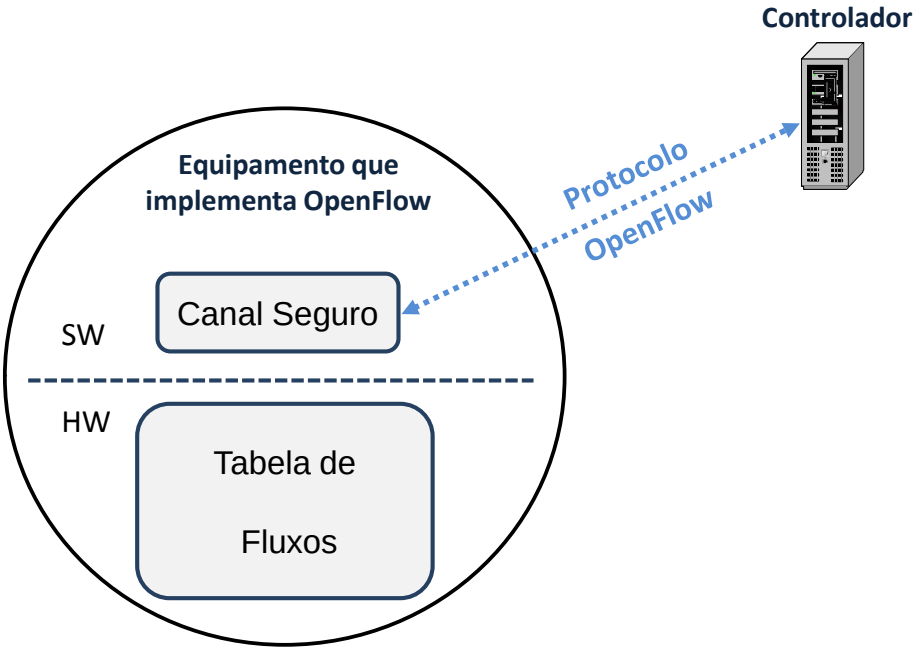

Figura 2.5: Arquitetura de equipamento que implementa OpenFlow

- descartar pacotes. Funcionalidade que pode ser usada para impedir o roteamento de determinados fluxos na rede.

2.3.2 Definições

A seguir são definidas as diferentes nomenclaturas utilizadas pelo protocolo OpenFlow, com o auxílio de alguns exemplos que ilustram melhor o conceito:

- Porta (*Port*): Ponto no qual um pacote ingressa ou sai da linha de processamento de OpenFlow. Pode ser uma porta física, uma porta lógica definida pelo *switch* (por exemplo, agrupamento de várias portas físicas como uma só porta lógica) ou portas reservadas, todas definidas no protocolo OpenFlow.
- Linha de processamento (*Pipeline Processing*): É composta por um grupo de tabelas de fluxos interligadas.
- Tabela de fluxos (*Flow Table*): Etapas da linha de processamento compostas por linhas de coincidência (entradas de fluxo ou regras).
- Entradas de fluxo (*Flow Entry*): São elementos na tabela de fluxos usados para processar pacotes buscando coincidências. Estas contêm um grupo de campos de coincidência para os pacotes processados, uma prioridade de busca, um grupo de contadores para registrar todas as ocasiões que as coincidências ocorrerem, além de ter um grupo de instruções a aplicar.

- Campos de coincidência (*Match Fields*): Campos que são examinados em procura de coincidência. Estes incluem os cabeçalhos dos pacotes, as portas de ingresso, valores de metadata, etc. Os campos de coincidência podem ser específicos na procura de um pacote particular ou podem procurar grupo de pacotes utilizando máscaras.

- Metadata: É o valor de um registro, o qual é usado para transportar informação de uma tabela para outra, dentro da linha de processamento.

- Instruções (*Instructions*): Integram as entradas de fluxo e são utilizadas para descrever o processamento que ocorre quando os pacotes coincidem com regras da tabela de fluxo. Uma instrução pode modificar a linha de processamento (direcionar o pacote para outra tabela de fluxo com número de sequência superior) ou conter um grupo de ações a serem executadas.

- *Ações*: Estas podem ser adicionadas ao *action set* e executadas ao final da linha de processamento, ou podem ser aplicadas imediatamente ao pacote (*apply-action*).

- *Apply-Action*: Ação que é executada imediatamente, no momento que é estabelecida a coincidência com uma regra de qualquer tabela dentro da linha de processamento.

- *Action Set*: É um grupo de ações que são cumulativas durante o processamento do pacote através das tabelas e só são executadas quando o pacote chega ao final da linha de processamento.

- Grupo: É uma lista de *action buckets* e uma maneira de escolher uma ou várias *action buckets* para cada pacote.

- *Action Bucket*: É um grupo de ações e parâmetros associados e definidos para grupos.

- Controlador: Uma entidade que interage com os *switches* OpenFlow, usando o protocolo OpenFlow. É o responsável da construção e gestão das tabelas de fluxo de cada equipamento da rede.

- *Meter*: Elemento do equipamento de rede que pode medir e controlar a taxa de transmissão para fluxos de pacotes.

2.3.3 Linha de processamento

Uma tabela de fluxos é formada por entradas de fluxo (regras), as quais são constituidas por: campos de coincidência, prioridade, contadores, instruções, tempos de espera e *cookie*, conforme ilustrado na Fig 2.6, em que as duas primeiras (campos de coincidência e prioridade) determinam univocamente uma entrada na tabela de fluxos. OpenFlow também estabelece uma linha de coincidência denominada *miss-table*, a qual pode ser colocada para processar os fluxos que não têm coincidência na tabela.

A Fig. 2.7 ilustra o processamento OpenFlow que ocorre na chegada de um novo pacote. Quando um novo pacote chega à rede, o equipamento OpenFlow realiza uma inspeção na Tabela 0 (primeira tabela do processamento), procurando coincidências dos campos do pacote com as

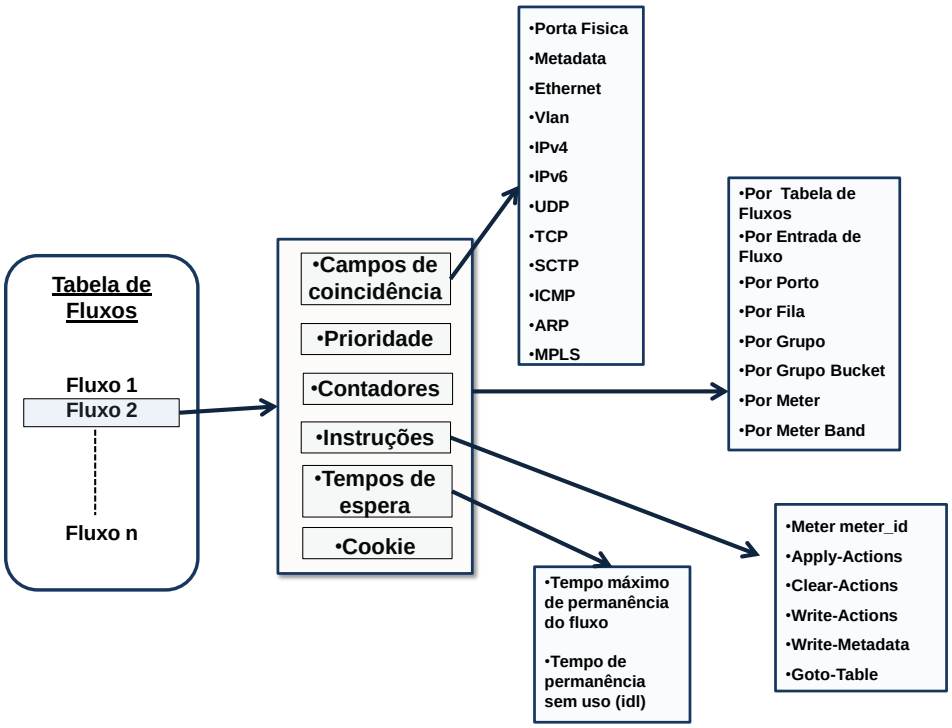

Figura 2.6: Elementos componentes das entradas de fluxo

regras da tabela de fluxos. No caso de encontrar coincidência, a regra terá que processar as instruções correspondentes, as quais terão ações associadas. Dentre as instruções, encontra-se o direcionamento para outra tabela de numeração superior, o que possibilita a realização de ligações sucessivas entre tabelas, formando a linha de processamento OpenFlow. O processamento de um pacote finaliza quando coincide (há *matching*) com uma regra que não tem direcionamento para outra tabela, executando as *action-set*.

2.3.3.1 Instruções

Como mencionado anteriormente, cada linha de coincidência (regra) têm instruções ligadas, conforme ilustrado na Fig. 2.6, pudendo ser dos seguintes tipos:

- *Meter*: Encaminha o pacote para um medidor de tráfego, proporcionando um método para construir limitadores de tráfego.
- *Apply-Actions*: Ações de execução imediata, sem modificar o *action set*. Estas ações são utilizadas para modificar o pacote entre tabelas. Se a *apply-action* indica uma ação *Output*, uma cópia do pacote é encaminhada imediatamente e o pacote original continua o processamento normal. Esta última é uma característica muito utilizada na arquitetura proposta no trabalho.
- *Write-Actions*: Ações que vão se adicionando no *action set* e só serão executadas ao final da última tabela. Se duas regras em diferentes tabelas da linha de processamento do pacote

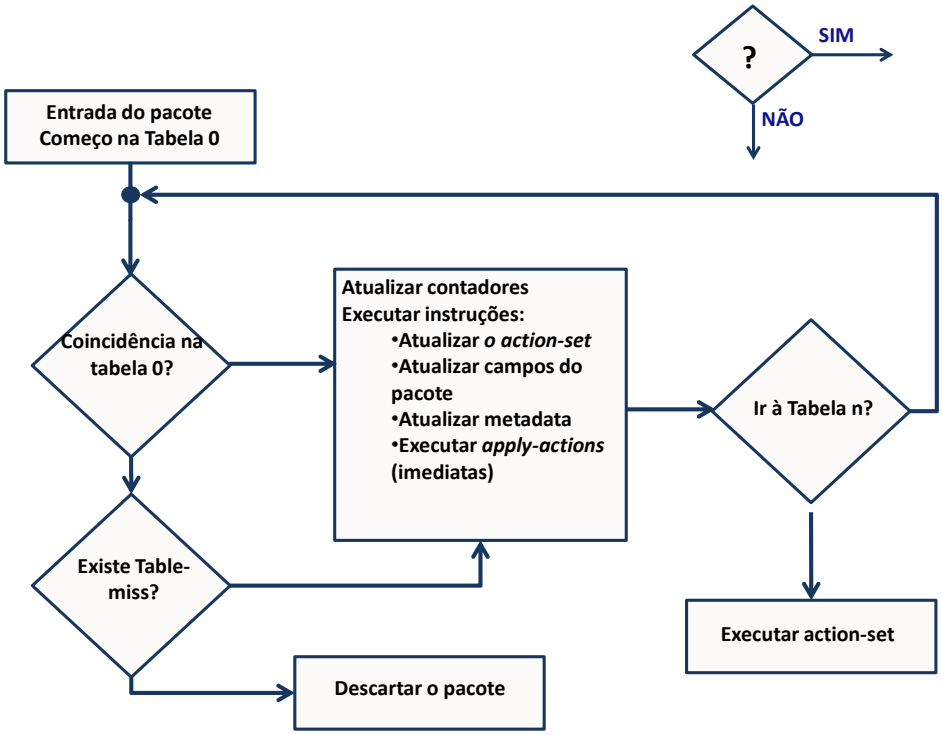

Figura 2.7: Processamento OpenFlow

estabelecem parâmetros diferentes para uma mesma ação, a última sobrescreve a primeira.

- *Clear-Actions*: Apaga as *action set* previamente preenchidas.
- *Write-Metadata*: Permite o envio de informação adicional entre tabelas.
- *Goto-Table*: Indica qual é a tabela que continuará o processamento do pacote.

2.3.3.2 Ações

A seguir são listadas e descritas as ações de maioir relevância para este trabalho:

- *Output*: Envía o pacote para uma porta de saída, a qual pode ser de um dos tipos a seguir:
  - Física: São portas definidas no equipamento que têm uma correspondência com as portas físicas do equipamento;
  - Lógica: São portas definidas no equipamento que não têm correspondência direta com portas físicas do *switch*. Estas portas são de mais alto nível de abstração e devem ser definidas utilizando métodos fora de OpenFlow (exemplo: *link aggregation group, tunnels, loopback interfaces*);
  - Reservada: São portas definidas por OpenFlow para facilitar roteamento de pacotes. A seguir são listadas as mais importantes para este trabalho:
    - *All*: Representa todas as portas do equipamento;

    * *Controller*: Representa o canal de controle com o controlador. Quando é indicada a porta *Controller*, o pacote ou o cabeçalho dele é encapsulado dentro de uma mensagem *packet-in* e é enviado ao controlador;
    * *In-Port*: Representa a porta de entrada do pacote;
    * *Normal*: Esta saída só pode ser utilizada por equipamentos híbridos, isto é, equipamentos que tenham implementado o controle distribuído atual (não OpenFlow) e o controle OpenFlow simultaneamente. Esta saída indica que o pacote tem que sair da linha de processamento OpenFlow para continuar seu processamento no plano de controle distribuído tradicional. Esta funcionalidade é detalhada e utilizada na arquitetura híbrida proposta na seção 3.2;

- *Set-Queue*: Ligado a uma ação de saída, determina qual fila de saída processa o pacote, permitindo aplicar políticas de priorização de tráfego;
- *Drop*: Especifica que o pacote tem que ser descartado;
- *Group*: Indica que seja processado por uma ação grupal, permitindo o pacote utilizar várias saídas simultaneamente, o balanceamento de tráfego, a implementação de caminhos alternativos realizando uma comutação rápida no caso de falha no caminho principal, etc. É importante ressaltar que diferentes entradas de fluxo (regras) na mesma ou em diferentes tabelas podem ter a mesma ação de grupo;
- *Push-Tag/Pop-Tag*: Permite ações com os *tag*, seja para a identificação de *VLAN* ou para o rótulo *MPLS*;
- *Set-Field*: Permite modificar ou estabelecer uma grande variedade de campos do cabeçalho do pacote;
- *Change-TTL*: Permite estabelecer ou copiar o *TTL* do *IP TTL* e do *MPLS TTL*.

2.3.4 As mensagens OpenFlow

Para a comunicação entre os diferentes elementos de rede e o controlador, é utilizado o canal OpenFlow. Este atua como interface, possibilitando a configuração e o gerenciamento dos diferentes equipamentos de rede. A implementação da comunicação entre o plano de dados e o canal OpenFlow é específica de cada equipamento, mas o canal OpenFlow tem que ser padrão. O protocolo OpenFlow dá suporte a três tipos de mensagens: *controller-to-switch*, *asynchronous* e *symmetric*, cada um deles com uma grande variedade de subtipos, conforme ilustrado na Fig 2.8.

2.3.4.1 Controller-to-Switch

Estas mensagens são iniciadas pelo controlador, e devem ou não ser respondidas pelos equipamentos clientes. Dentre estas mensagens, destacam-se os subtipos descritos a seguir:

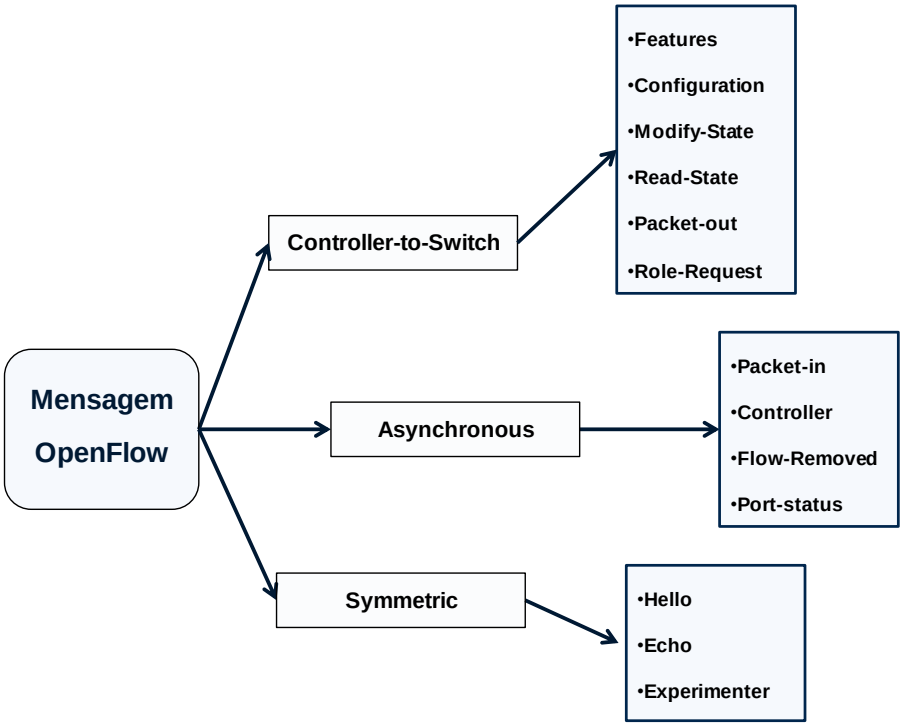

Figura 2.8: Tipos das mensagens OpenFlow

- *Features*: O controlador pode solicitar as capacidades do equipamento, enviando uma solicitação *Feature*. O *switch* deve responder com um *Features Replay*, especificando assim suas capacidades. Isto é usualmente realizado após o estabelecimento do canal OpenFlow;
- *Configuration*: O controlador pode estabelecer e solicitar parâmetros de configuração aos equipamentos;
- *Modify-State*: Estas mensagens são enviadas pelo controlador para gerenciar o estado dos *switches*. O principal propósito é adicionar, apagar e modificar entradas de fluxo nas tabelas, e também para estabelecer propriedades nas portas dos equipamentos;
- *Read-State*: É utilizado pelo controlador para coletar diversos tipos de informação dos clientes como configuração atual, estatísticas e capacidades do equipamento;
- *Packet-out*: Utilizado pelo controlador para especificar qual é a porta de saída que o equipamento cliente tem que utilizar para encaminhar o pacote, o qual foi inicialmente recebido por uma mensagem *Packet-in*. O *Packet-out* pode conter o pacote completo ou uma referência a um pacote guardado no cliente. Adicionalmente, esta mensagem tem que conter uma lista de ações a serem aplicadas, mas no caso de não haver ações, a mensagem indica que o pacote tem que ser apagado. Esta última característica é muito utilizada neste trabalho;
- *Role-Request*: Mensagem utilizada pelo controlador para estabelecer seu papel no canal OpenFlow ou para perguntar o papel. Isto é utilizado quando um cliente é conectado a vários controladores.

2.3.4.2 Asynchronous

Estas mensagens são enviadas pelos equipamentos cliente para o controlador, sem nenhuma solicitação prévia. Estas mensagens são utilizadas para comunicar a chegada de um pacote, a mudança do estado de uma porta ou algum erro. Os principais tipos destas mensagem são listados a seguir:

- *Packet-in*: Estas mensagens permitem transferir o controle do roteamento de um pacote de entrada ao controlador. No caso de um pacote entrante ter coincidência com uma regra ligada a uma ação *output* à porta reservada *Controller*, esta encaminha o pacote ao controlador mediante uma mensagem *Packet-in*. Esta mensagem pode conter o pacote total ou só parte de seu cabeçalho, mas, neste último caso, o equipamento de rede tem que guardar o pacote completo;
- *Flow-Removed*: O equipamento de rede informa ao controlador que um fluxo foi removido de uma das tabelas de fluxo. Estas mensagens são enviadas por uma entrada de fluxo, caso tenha sido estabelecida a *flag OFPFF_ SEND_ FLOW_ REM*, e são geradas como resultado de uma solicitação do controlador de remoção de fluxo ou devido à expiração do *timeout* ligado à entrada de fluxo;
- *Port-status*: O cliente informa ao controlador a mudança no estado de uma de suas portas. Estas mensagens notificam as mudanças de estado das portas, seja por uma ação de configuração no equipamento ou por um evento de mudança de estado;
- *Error*: Com estas mensagens, os clientes podem notificar ao controlador vários tipos de erros.

2.3.4.3 Symmetric

Estas mensagens são enviadas pelos equipamentos de rede ou pelo controlador, sem nenhuma solicitação prévia.

- *Hello*: São mensagens trocadas entre o controlador e os equipamentos de rede, na inicialização da conexão.
- *Echo*: São mensagens enviadas pelo controlador e pelos clientes. Inicialmente é enviada uma solicitação *Echo request*, que é respondida mediante um *Echo reply*. Estas mensagens são principalmente utilizadas para verificar o estado do canal OpenFlow.
- *Experimenter*: Permite implementar funcionalidades adicionais, dentro do espaço das mensagens OpenFlow.

2.3.5 Troca de mensagens

Nas subseções a seguir, é descrita uma comunicação básica entre um equipamento cliente e o controlador.

2.3.5.1 Estabelecimento da conexão

As conexões no canal OpenFlow, ilustradas na Fig. 2.9, são iniciadas pelo cliente, que tem previamente configurados o endereço IP e a porta do controlador. Inicialmente, é estabelecida uma conexão TLS (*Transport Layer Security*) ou uma conexão TCP (para o caso de canal sem segurança), e em seguida o controlador e o cliente OpenFlow passam a trocar as mensagens *Hello* que permitem acordar os parâmetros básicos (como a versão de OpenFlow a ser utilizada). Quando a conexão se encontra estabelecida, ambos os dispositivos trocarão periodicamente mensagens *Echo* (*request/reply*), verificando, assim, o estado do canal de controle OpenFlow.

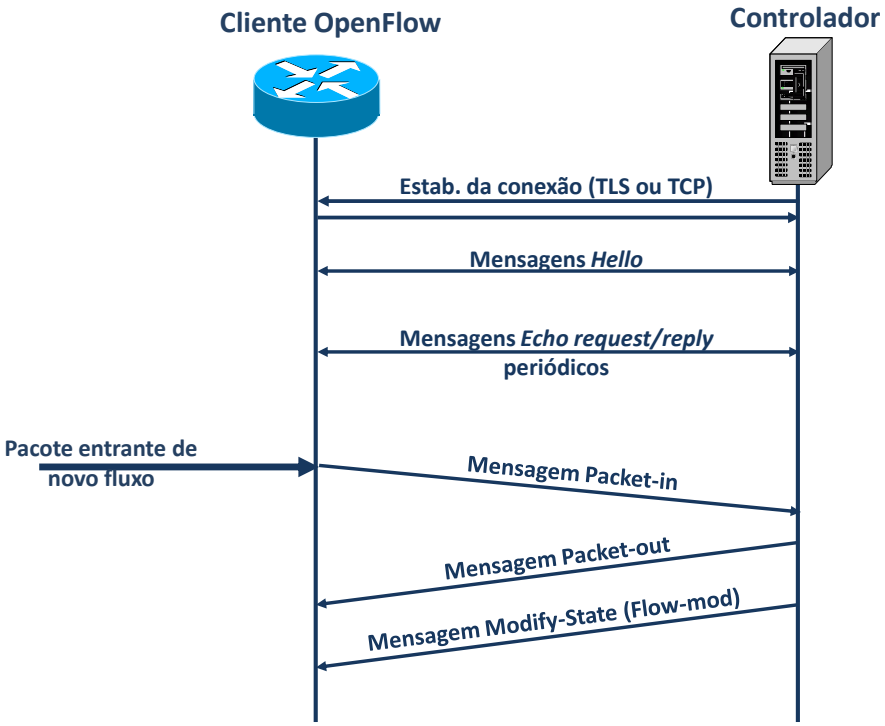

Figura 2.9: Sequência das mensagens OpenFlow

2.3.5.2 Chegada de um pacote

Quando um pacote pertencente a um novo fluxo chega ao cliente OpenFlow, este é encaminhado ao controlador mediante uma mensagem *Packet-in* (Fig. 2.9). O controlador calcula o melhor caminho para o pacote entrante e, normalmente, executa as duas ações seguintes:

1. responde o *Packet-in* mediante um *Packet-out*, indicando como o equipamento deve encaminhar o pacote;
2. gera uma mensagem do tipo *Modify-State* (*Flow-Mod*) para que seja criado um fluxo específico na tabela do equipamento cliente e de todos os outros clientes intervenientes no caminho. Desta forma, os pacotes sucessivos do mesmo fluxo não têm que realizar a mesma consulta ao controlador.

Capítulo 3

Arquitetura Proposta Com e Sem Equipamentos Híbridos

3.1 Arquitetura Proposta

3.1.1 Considerações gerais

Neste trabalho, é proposta uma arquitetura de rede de SP baseada em OpenFlow com plano de dados MPLS [Das, 2012] [López and Campelo, 2013]. A tecnologia MPLS já contempla a abstração do plano de dados mediante o conceito de fluxos, adequada, portanto, à utilização com OpenFlow. O próprio protocolo OpenFlow vem sendo desenvolvido nesta direção [OpenFlow v.1.3 , 2012], com a incorporação das funcionalidades necessárias que permitem a utilização do plano de dados MPLS. Dentre as funcionalidades adicionadas, existe a possibilidade de que as regras dentro da tabela de fluxos comparem, modifiquem, adicionem e apaguem rótulos MPLS.

O fato de OpenFlow permitir definição e controle de fluxos individuais incrementa a precisão e as possibilidades de gerenciamento. Contudo, esta manipulação individual também leva a um maior processamento, atraso e sobrecarga na rede. É importante destacar que cada fluxo adicional requer a configuração da regra correspondente nas tabelas de fluxos de todos os equipamentos intervenientes no encaminhamento fim a fim, com várias comunicações com o controlador. Por esta razão, é importante distinguir e definir as características dos diferentes tipos de tráfego na rede, analisando qual tráfego requer um tratamento fluxo a fluxo e qual um tratamento geral, com o objetivo de diminuir a carga no controlador.

3.1.2 Lógica geral de funcionamento

Na arquitetura proposta, há uma tabela inicial indexada como 0 em cada cliente OpenFlow, a qual encaminha o processamento a uma lógica de gerenciamento especializada, conforme ilustrado na Fig. 3.1. Na tabela 0, são aplicadas regras gerais para classificar os pacotes de entrada com o menor número de regras de coincidência possíveis, as quais podem verificar, por exemplo, presença

do rótulo MPLS, portas específicas de VoIP, vídeo ou outras aplicações, grupo de direções IPs internas, *IP precedence* do cabeçalho IP ou EXP (*Experimental bits*) do cabeçalho MPLS, entre outras. Estas regras poderão, opcionalmente em caso de tráfego QoS, marcar o *IP precedence* do pacote para facilitar o processamento nas tabelas e equipamentos seguintes. As lógicas de gerenciamento especializadas escolhidas são descritas e denominadas como é indicado a seguir:

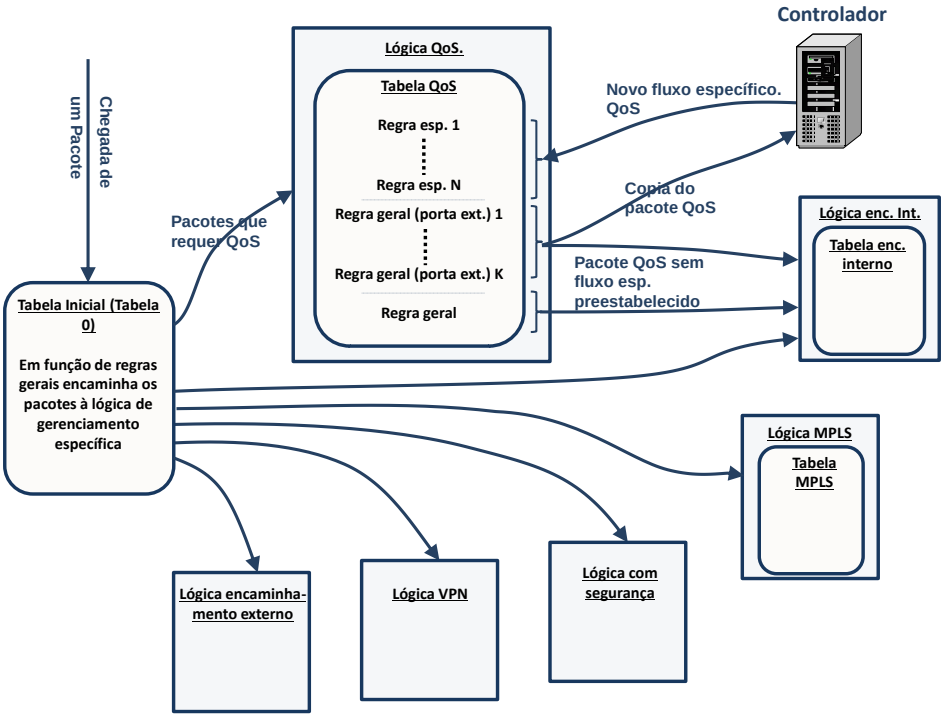

Figura 3.1: Arquitetura geral e lógica de gerenciamento QoS em equipamento de borda

- Encaminhamento interno: permite alcançar todas as redes internas pertencentes ao SP, como as redes de serviços internos e redes de clientes corporativos ou residenciais.
- Encaminhamento MPLS: é o encaminhamento baseado no rótulo MPLS. Encaminhamento similar ao obtido nas redes atuais com protocolos como RSVP ou LDP.
- Encaminhamento de QoS: permite tratar fluxos que requeiram QoS fim a fim em razão dos seus requisitos de serviço. Exemplos são fluxos de comunicação de VoIP, vídeo, TV digital e aplicações de controle em tempo real. Esta lógica é uma das principais contribuições deste trabalho de mestrado.
- Outros lógicas não desenvolvidos neste trabalho como:
  - Encaminhamento externo que permite alcançar todos os destinos fora do sistema autônomo;
  - Encaminhamento VPN (*Virtual Private Network*) que permitem interconectar diferentes pontos clientes mantendo políticas de isolamento;

– Encaminhamento com regras de segurança: utilizado para o tráfego que necessita ser autorizado antes de ser encaminhado.

Cada uma destas lógicas de gerenciamento é implementada por uma tabela ou grupo de tabelas OpenFlow interligadas, as quais se encontram dentro da linha de processamento de OpenFlow em cada um dos equipamentos, como mostrado na Fig. 3.1. Estas tabelas variam em função do tipo de cliente OpenFlow e do lugar em que ele se situa na rede. Um exemplo é o caso dos OpenFlow LSR (*OpenFlow Label Switch Routers*), que não requerem conhecimento de BGP ou VPN para o roteamento, pois encaminham os pacotes considerando somente o rótulo MPLS.

3.1.3 Lógicas de gerenciamento para encaminhamento interno e MPLS

Estas lógicas são formadas por tabelas com informação de encaminhamento interno para todas as redes destino, similar ao que acontece nas redes atuais (isto é, não fluxo a fluxo). Estas tabelas são construídas inicialmente (criadas proativamente antes da chegada dos fluxos) pelo controlador em cada um dos clientes OpenFlow e atualizadas dinamicamente pelo controlador quando uma mudança topológica acontece. Para isso, o controlador deve possuir a informação de todas as redes diretamente conectadas a cada equipamento e as interconexões entre equipamentos (topologia) e deve ser informado pelos clientes OpenFlow de qualquer mudança.

Com o conhecimento topológico é possível para o controlador executar um protocolo de roteamento interno conveniente para descobrir como alcançar todas as redes a partir de cada um dos nós, e assim construir as tabelas para a lógica de gerenciamento para o encaminhamento interno em cada um dos clientes OpenFlow. Em seguida, o controlador estabelece ligações dos rótulos MPLS com as redes, formando o que é denominado FEC (*Forwarding Equivalent Class*) na terminologia MPLS. Com a configuração adequada de engenharia de tráfego no controlador é possível construir e preencher as tabelas que compõem a lógica de gerenciamento para o encaminhamento MPLS.

Estas duas tabelas deverão incluir os campos de coincidência (*matching*), as ações de modificação, adição ou remoção de rótulos MPLS, assim como a porta de saída correspondente, entre outras. Nos casos em que o protocolo de roteamento encontrar mais de um caminho ótimo, ambos poderão ser considerados e adicionados nas tabelas correspondentes. Para isto, é possível configurar no OpenFlow as *Group Tables* [OpenFlow v.1.3 , 2012], que permitem tratar o tráfego com um algoritmo de balanceamento adequado e assim melhorar a distribuição da carga na rede.

É importante ressaltar que o algoritmo de roteamento só é conhecido e executado no controlador, que é o responsável por obter o encaminhamento para todos os destinos para cada um dos nós. Os dispositivos de rede não precisam trocar informação de roteamento e mudança topológica, não sendo necessário aguardar tempos de espera que assegurem a propagação correta da informação; é suficiente informar ao controlador as mudanças para se obter a nova convergência. Por fim, não é necessário utilizar algoritmos complexos de propagação e roteamento como OSPF-TE (*OSPF Traffic Engineering*) ou ISIS-TE (*ISIS Traffic Engineering*), nem de distribuição de rótulos como LDP ou RSVP.

3.1.4 Lógica de gerenciamento para QoS

Quando um pacote que requer QoS entra na rede, esta lógica cria um caminho fim a fim (caminho criado reativamente ao fluxo QoS entrante) que permita satisfazer os requisitos de QoS do fluxo específico. Para identificar quando um pacote entra na rede, devem ser identificados a fronteira da rede e, com isso, os equipamentos de borda (os que têm interfaces externas e internas) e os equipamentos internos (os que têm todas as interfaces internas). Para compreender como esta lógica trabalha, descrevem-se inicialmente os tipos de regras que a tabela QoS possui:

- Regras gerais: permitem encontrar coincidências gerais com qualquer tipo de tráfego de QoS considerado. Adicionalmente, nos equipamentos de borda, as regras QoS permitem diferenciar se o pacote está entrando na rede ou saindo dela (examinando a interface de entrada do pacote). Estas regras são criadas inicialmente para encaminhar os primeiros pacotes de cada fluxo QoS e são utilizadas durante o período de tempo no qual ainda não existe uma regra específica para o fluxo individual.
- Regras específicas: são regras para cada fluxo QoS individual (fluxos que podem opcionalemnte se definir univocamente por porta e IP origem mais porta e IP destino). Inicialmente, a tabela QoS não dispõe de regras específicas; estas serão preenchidas pelo controlador durante a operação.

Quando um novo pacote QoS entra na rede por um equipamento de borda, este é encaminhado à sua tabela QoS e é inicialmente processado por uma regra QoS geral com portas de entrada externas. Esta regra encaminha o pacote por duas linhas de processamento simultaneamente, conforme ilustrado na Fig. 3.1 e detalhado na 3.2, são descritas a seguir:

1. Primeiramente, a regra geral com portas de entrada externas na tabela QoS envia uma requisição de novo fluxo ao controlador (*packet-in*) contendo uma cópia total ou parcial do pacote. Para isso, o cliente OpenFlow tem a opção da **instrução** de execução imediata do tipo ***Apply-Actions*** e dentro desta uma **ação** do tipo ***Output*** à porta reservada ***Controller*** [OpenFlow v.1.3 , 2012]. Com esta informação, o controlador calculará o caminho ótimo que permita satisfazer os requerimentos QoS do fluxo. Para isso, podem ser utilizados algoritmos de Dijkstra com restrições similares aos utilizados por ISIS-TE ou OSPF-TE, ou protocolos de encaminhamento novos específicos para cada tipo de qualidade de serviço, como, por exemplo, tráfego em tempo real sem perdas ou tráfego em tempo real com possibilidade de perda [Egilmez et al., 2011]. Finalmente, o controlador envia uma mensagem *packet-out* sem ação, indicando que o pacote que gerou a consulta tem que ser apagado (pois o pacote já foi encaminhado pela segunda linha de procesamento, descrita no ponto 2 abaixo) e configura uma regra específica para este novo fluxo na tabela de QoS de cada um dos equipamentos intervenientes no caminho ótimo obtido (fluxo construído no sentido oposto ao encaminhamento). É importante mencionar que todos os fluxos de QoS específicos são criados como prioritários (para preceder as regras gerais) e com um *timeout* apropriado (para que sejam apagados automaticamente depois de um período de inatividade). Adicionalmente,

OpenFlow tem a opção de uma fila de atendimento preferencial e os fluxos QoS específicos podem ser ligados a esta.

2. Simultaneamente ao ponto anterior, o pacote continua o processamento estabelecido pela regra geral de QoS com portas de entrada externas. Esta regra tem definida uma **instrução** do tipo ***Goto-Table*** [OpenFlow v.1.3 , 2012], que indica que o pacote tem que continuar seu processamento pela tabela de encaminhamento interno (Fig. 3.1 e 3.2). Desta forma, o pacote será processado imediatamente como um fluxo sem qualidade de serviço e não terá que aguardar a resposta do controlador.

Para os equipamentos internos ou no caso de um equipamento de borda de saída, a regra geral QoS utilizada é mais simples, dado que só tem que enviar ao pacote para a tabela de encaminhamento interno. Isto acontece porque a solicitação de criação da regra específica já foi enviada ao controlador pelo primeiro equipamento de borda e o controlador criará o fluxo QoS específico em todos os equipamentos do caminho (incluídos os equipamentos de borda de saída). É dessa forma que estes equipamentos enviam o pacote somente à tabela de encaminhamento interno, até que eles tenham uma regra QoS específica preenchida.

3.1.5 Considerações importantes da arquitetura.

Ressalta-se que na arquitetura proposta, o controlador não adiciona tempos de espera para os primeiros pacotes de cada fluxo. Esta afirmação é baseada no fato de que os pacotes do mesmo fluxo sempre são encaminhados inicialmente pelo encaminhamento interno e são paralelamente processados pelo controlador. Posteriormente, quando a tabela de QoS tiver o fluxo específico preenchido, os pacotes subsequentes do fluxo coincidirão com esta nova regra e encaminharão o fluxo pela via ótima para este tipo de tráfego.

Adicionalmente, destaca-se como uma das características mais importantes desta lógica, a redução sensível da dependência do controlador. Na arquitetura proposta, quando o controlador cair ou apresentar tempos de resposta altos, os novos fluxos QoS sempre continuam sendo encaminhados pela lógica de gerenciamento interna. Este é um aspecto de robustez indispensável para um SP.

Além disso, para qualquer implementação de OpenFlow [Luo et al., 2012] é importante haver mecanismos nos clientes que evitem que pacotes sucessivos do mesmo fluxo gerem as mesmas consultas ao controlador enquanto se aguarda a resposta do controlador, evitando-se sobrecargas desnecessárias. Na arquitetura OpenFlow padrão, isto é complexo de implementar, dado que ela requer que todos os pacotes seguintes no período de tempo que se aguarda por resposta sejam armazenados na memória do equipamento; quando o *packet-out* do primeiro pacote chegar, tem que ser executada a ação que o *packet-out* indique em todos os pacotes que aguardam em memória. Na arquitetura proposta, é mais fácil de implementar, dado que não há a necessidade de manter nenhuma informação dos sucessivos pacotes, pois, quando o *packet-out* chegar eles já terão sido encaminhados pela regra de encaminhamento interno. Na arquitetura proposta, deve-se guardar o cabeçalho do primeiro pacote de cada um dos fluxos que aguarda resposta do controlador e

comparar o cabeçalho do pacote entrante, com os cabeçalhos dos pacotes guardados. Desta forma, é possível conhecer se este é o primeiro pacote de um fluxo QoS ou se é um pacote de um fluxo que aguarda resposta (ver Fig 3.2).

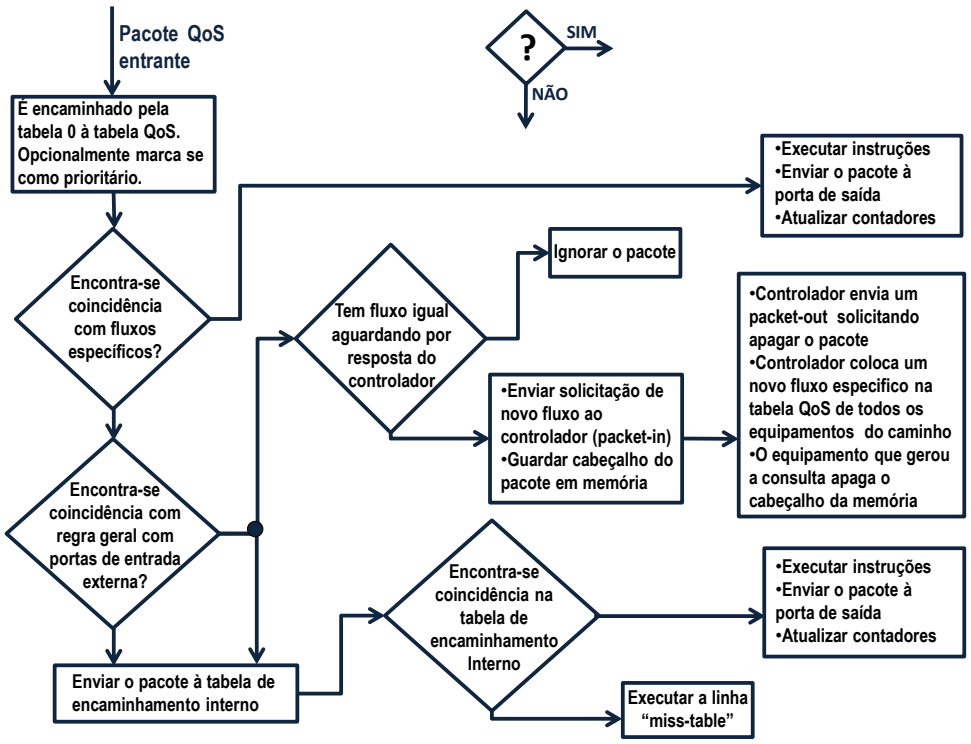

Figura 3.2: Arquitetura de controle proposta para equipamento de borda

Como os primeiros pacotes de um fluxo QoS são encaminhados pela lógica de encaminhamento interna e os sucessivos pacotes do fluxo pelo caminho QoS específico, é muito provável que a trajetória do fluxo mude e, com isso, os equipamentos internos utilizados no trajeto. Este comportamento ressalta a importância de os equipamentos internos não realizarem a consulta de novo fluxo QoS ao controlador, pois eles podem não estar no caminho QoS ótimo. Desta forma, previne-se que o controlador crie um fluxo QoS adicional desnecessário que utilize em seu cominho o nó interno que realiza a consulta.

Menciona-se também que, segundo o resultado obtido em [Wamser et al., 2011], na caracterização do tráfego para dispositivos móveis, a probabilidade de se encontrar um pacote pertencente a um novo fluxo em um cliente OpenFlow pode alcançar 4%. Se projetarmos este resultado ao tráfego em geral, isto implica que para o caso de OpenFlow padrão, 4% do tráfego total pode ser encaminhado ao controlador. No caso de um SP, este número elevado de consultas representa um desafio muito grande para um controlador e, adicionalmente requer o aumento dos recursos da rede para dar suporte ao incremento de largura de banda.

Como alternativa ao problema mencionado no parágrafo anterior, a arquitetura proposta permite diminuir consideravelmente a carga do controlador, pelas considerações descritas a seguir:

- o tráfego sem QoS não é encaminhado ao controlador. Ele é encaminhado pela tabela de

encaminhamento interno proativamente configurada;

- só o primeiro pacote de cada fluxo QoS é enviado ao controlador, reduzindo ainda mais a carga;
- quando chegar um novo pacote QoS, só os equipamentos de borda de entrada na rede geram uma consulta ao controlador.

Finalmente, menciona-se que para os caminhos específicos QoS não é implementado o balanceamento de carga, obtendo-se, assim, um caminho único similar aos VCs (*virtual circuits*) de outras tecnologias, reduzindo o *jitter* do tráfego QoS da rede.

3.2 Arquitetura com equipamentos híbridos

3.2.1 Conceito de equipamento híbrido

Um elemento de rede OpenFlow Híbrido é um equipamento de rede que incorpora as funcionalidades de OpenFlow e ao mesmo tempo mantém as capacidades do plano de controle distribuído padrão, conforme ilustrado na Fig. 3.3. Estes equipamentos possuem duas linhas de processamento para o envio de pacotes: a padrão que utiliza uma tabela de encaminhamento construída com a informação obtida do plano de controle distribuído clássico, isto é, informação obtida por protocolos como OSPF, ISIS, LDP, RSVP e BGP implementados em cada equipamento de rede, e a linha de processamento construída e controlada pelo controlador utilizando OpenFlow, composta por um grupo de tabelas OpenFlow interligadas mediante a **instrução *Goto-Table***.

Atualmente, existem várias opções de implementação de *switches* híbridos [Ivan Pepelnjak, 2012] [Open Networking Foundation, 2013], algumas delas já se encontram implementadas em equipamentos de rede. Estas opções tentam proporcionar alternativas de processamento que permitam distinguir quando um pacote de entrada tem que ser processado pelo controle distribuído ou pelo controle OpenFlow, dentre as quais:

- Que cada porta do equipamento de rede seja ligada a um único tipo de controle. Isto é, que seja ligada ao controle OpenFlow ou ao controle distribuído atual.
- Que cada VLAN seja ligada a um tipo único de controle, melhorando a utilização das portas disponíveis no equipamento.
- Que exista uma ACL inicial que permita distinguir qual pacote tem que ser processado por um tipo de controle e qual pacote tem que ser processado pelo outro tipo de controle.
- Que o controle OpenFlow seja o primeiro caminho de procura, mas se não existir coincidência com uma linha de OpenFlow, continuar o processamento pelo controle distribuído atual.
- Utilizar a funcionalidade ainda não utilizada, mas já considerada na especificação de OpenFlow 1.3 (porta reservada ***Normal***) que permite que qualquer regra dentro das tabelas que

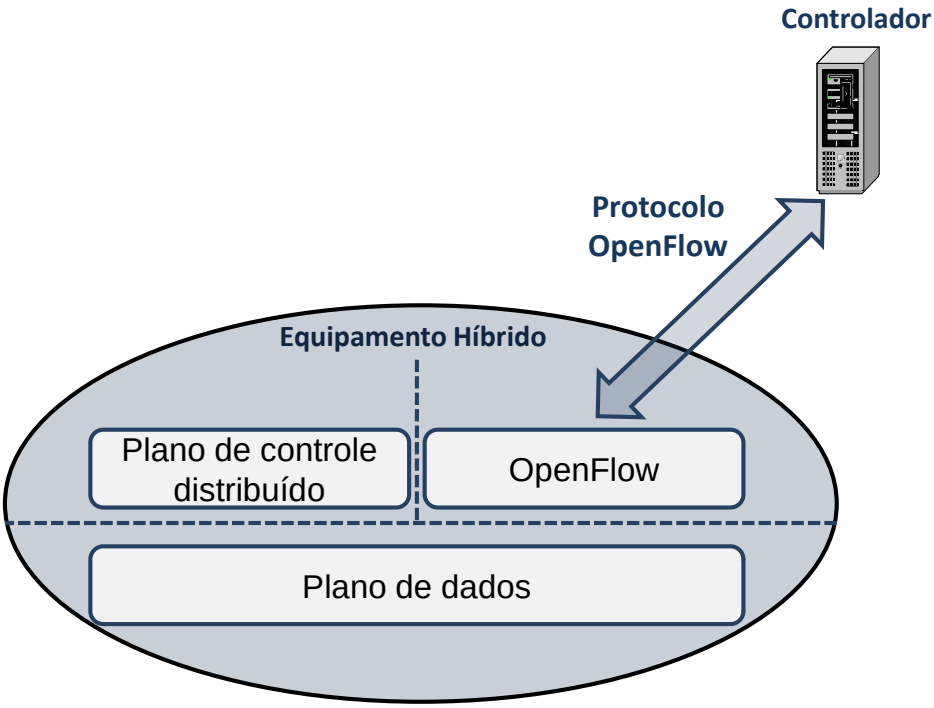

Figura 3.3: Equipamento OpenFlow híbrido

compõem a linha de processamento OpenFlow encaminhe o pacote ao plano de controle distribuído padrão. Para esta finalidade, OpenFlow permite definir em suas regras de coincidência nas instruções ***Apply-Actions*** ou ***Write-Actions*** a **ação *output*** à porta reservada ***Normal*** [OpenFlow v.1.3 , 2012]. Desta forma o pacote coincidente nesta regra é encaminhado internamente ao controle distribuído clássico.

Pela potencialidade que a última das alternativas proporciona e pelo fato de que na arquitetura proposta os equipamentos de rede não são somente elementos de camada de enlace de dados, esta última foi a alternativa escolhida para esta proposta de arquitetura com equipamentos híbridos. Nesta proposta, todos os pacotes são inicialmente processados pelo controle OpenFlow e este encaminhará os pacotes ao controle distribuído clássico utilizando a funcionalidade previamente descrita. A arquitetura é desenvolvida na secção 3.2.2.

Finalmente, ressalta-se que segundo indicado ao longo da dissertação, os equipamentos que implementam a especificação 1.3 de OpenFlow podem, em qualquer uma das suas regras presentes em suas tabelas, enviar o pacote ao controlador para outra tabela, encaminhar diretamente o pacote por uma porta de saída, redirecionar o pacote ao processamento distribuído clássico, encaminhar o pacote por uma ação de grupo ou realizar várias destas alternativas simultaneamente combinando as **instruções *Apply-Actions***, ***Write-Actions*** e ***Goto-Table***.

3.2.2 Proposta com equipamentos híbridos

A arquitetura hibrida proposta combina equipamentos híbridos, com a proposta de arquitetura da seção 3.1. Desta forma, é possível desenvolver uma arquitetura alternativa que programe mediante OpenFlow a Tabela 0 e a lógica de gerenciamento de QoS e, por outra parte, realize a lógica de gerenciamento interno e MPLS com o controle distribuído clássico (ver Fig. 3.4). As lógicas restantes de gerenciamento poderão ser implementadas por OpenFlow ou pelo controle distribuído, caso seja conveniente.

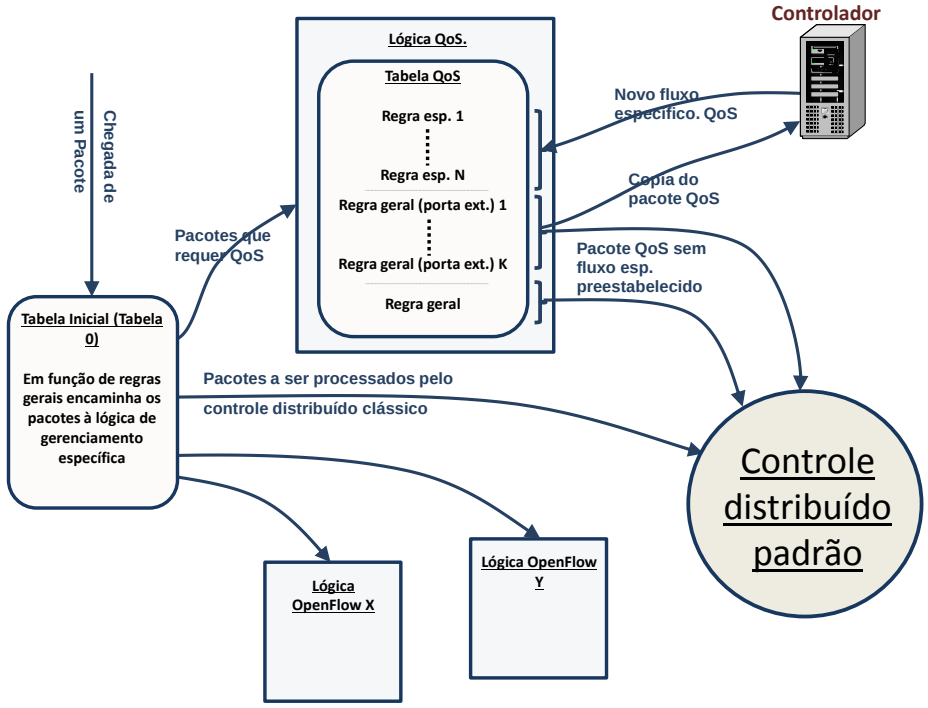

Figura 3.4: Arquitetura de controle com equipamentos híbridos

Novamente, nesta arquitetura o controle implementado nos equipamentos de borda (os que tem portas externas e internas à rede OpenFlow) deve ser diferenciado do controle implementado nos equipamentos internos (os que tem todas as portas conectadas a equipamentos pertencentes à rede OpenFlow). Esta distinção se deve ao fato de que somente os roteadores de borda solicitarão novo fluxo QoS ao controlador.

Com a arquitetura previamente mencionada, o tráfego com QoS tem o comportamento descrito a seguir e ilustrado na Fig. 3.5. Quando um pacote QoS pertencente a um novo fluxo chega a um equipamento de borda por uma porta externa (pacote entrante à rede), este é inicialmente processado pela Tabela 0 e é encaminhado à tabela QoS. Como ainda não existe uma regra específica configurada para o novo fluxo, o pacote é encaminhado por uma regra QoS geral (com porta de entrada externa) que encaminhará o pacote ao controlador e simultaneamente para o controle distribuído padrão (analogamente ao indicado na seção 3.1). Desta forma, enquanto não existir a regra específica na tabela QoS, os sucessivos pacotes sucessivos do novo fluxo serão encaminhados pelo controle distribuído padrão e quando a regra específica estiver disponível, estes pacotes serão

encaminhados por esta regra.

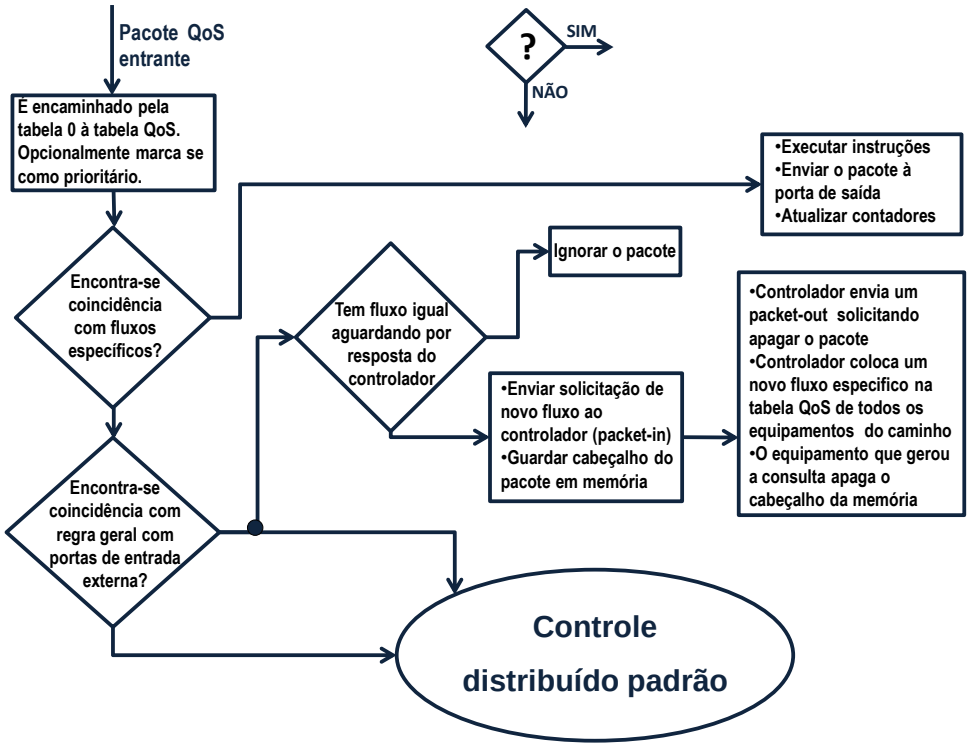

Figura 3.5: Arquitetura de controle proposta para equipamentos hibridos de borda

3.2.3 Considerações adicionais

Esta arquitetura é considerada na comparação de arquiteturas apresentada na seção 5.3 e poderia ter pouca resistência a ser utilizada em um SP. Esta arquitetura incrementa as possibilidades de engenharia de tráfego por meio da incorporação de OpenFlow e, ao mesmo tempo, mantém completamente a robustez das redes atuais, pois, no caso de queda do controlador, o serviço pode continuar funcionando indefinidamente com o controle distribuído clássico. Esta topologia é robusta ainda no caso de mudança topológica durante a queda do controlador.

Finalmente, menciona-se que esta arquitetura pode ser de muita utilidade em um processo de migração que tenha como objetivo transformar uma rede com controle distribuído padrão para uma rede com o controle centralizado OpenFlow, como a desenvolvida na seção 3.1. Ressalta-se que as diferentes lógicas podem ser migradas independentemente e por etapas, proporcionando um processo de migração confiável e controlado.

Capítulo 4

Cálculo de atrasos com OpenFlow padrão

4.1 Introdução

Neste capítulo, é realizada uma modelagem analítica do cálculo do atraso que a arquitetura OpenFlow padrão adiciona na criação de novos fluxos em um provedor de serviços. Adicionalmente, são mostrados os resultados do atraso obtido e como ele depende dos diferentes parâmetros da rede.

É importante lembrar que, com OpenFlow padrão na arquitetura proposta, os primeiros pacotes de cada fluxo são encaminhados imediatamente, sem ter que aguardar resposta do controlador. Neste capítulo, é quantificada uma das melhorias proporcionadas pela arquitetura proposta.

4.2 Considerações prévias

As mensagens entre cliente OpenFlow e o servidor OpenFlow são atrasadas por diferentes componentes que, juntos, contribuem no atraso total fim-a-fim. Estes podem ser divididos em quatro grupos, de acordo com suas características, ilustradas na Fig. 4.1:

- Atraso por propagação: tempo decorrido em que a informação tarda em se propagar por um meio físico (por exemplo, o tempo que a luz tarda em se propagar dentro de um cabo de fibra).
- Atraso por processamento: está presente em todo nó do trajeto da mensagem e se deve à verificação de erros do pacote recebido, às operações efetuadas nos pacotes e ao processo de determinação da melhor saída.
- Atraso por transmissão: ligado ao tempo que tarda um nó em colocar os bits de um pacote no meio físico. Este dependerá do tamanho do pacote e da largura de banda do link.

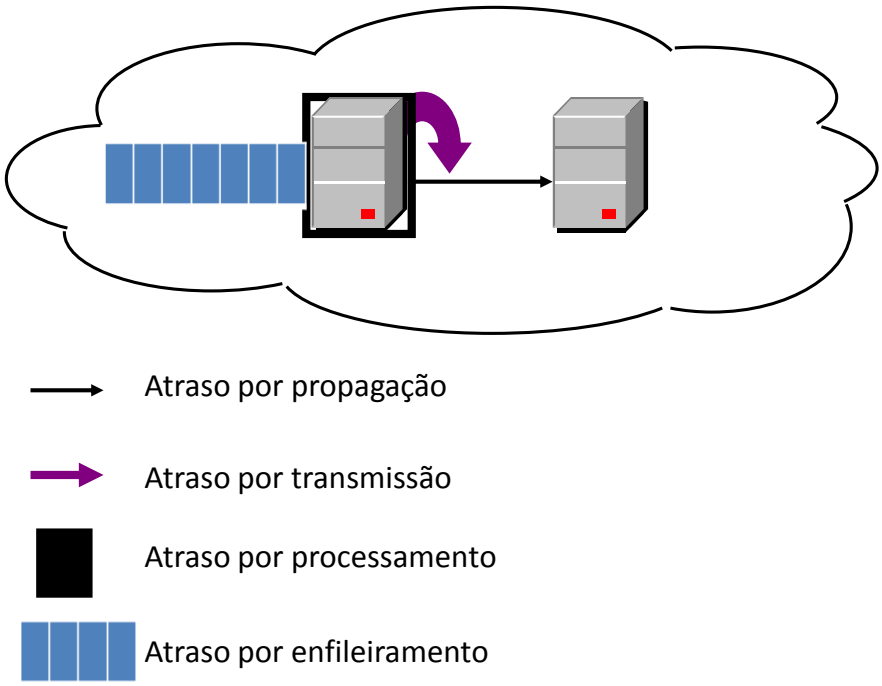

Figura 4.1: Tipos de atraso

- Atraso por enfileiramento: trata-se do tempo que o pacote que chega a um nó tem que aguardar para começar a ser transmitido pelo nó. Depende da carga presente no nó.

4.2.1 Atraso por propagação

Para o cálculo teórico, considera-se a velocidade de propagação (V_p) do meio físico que transporta a informação. Para a fibra ótica, tem-se:

$$V_p = \frac{c}{n} \tag{4.1}$$

em que n é o indice de refração no meio e c é a velocidade da luz no vácuo. Se o comprimento da fibra é L, o atraso por propagação é:

$$T_p = \frac{L}{V_p} \tag{4.2}$$

e o atraso total por propagação num sistema será obtido do somatório dos atrasos em cada um dos segmentos que compõem o trajeto.

4.2.2 Atraso por processamento

Este será modelado dentro do tempo de serviço no atraso por enfileiramento.

4.2.3 Atraso por transmissão

Para seu cálculo é considerada a velocidade de transmissão do link v (em bit/s) e o comprimento do bloco a ser transmitido, l (em bits). O atraso de transmissão T_t será, portanto:

$$T_t = \frac{l}{v} \tag{4.3}$$

para cada um dos nós presentes no trajeto fim a fim. O atraso total por transmissão será o somatório dos atrasos individuais na saída de cada nó.

4.2.4 Atraso por enfileiramento

Para o cálculo do atraso em uma rede de filas, foram feitas diversas hipóteses de simplificação que permitem decompor o sistema total em um somatório de filas individuais. A base teórica para que seja possível esta simplificação encontra-se no teorema de Jackson [William Stallings, 2000], o qual está baseado em três suposições:

- A rede de filas consiste em m nós, em que cada nó possui um tempo de serviço com distribuição exponencial e independente dos outros nós.
- O tráfego de chegada de fora do sistema por qualquer um dos nós segue uma distribuição de Poisson.
- Quando um item é servido por um nó, ele vai imediatamente para outro nó com uma probabilidade fixa ou vai para fora do sistema também com probabilidade fixa. Isto é, se a saída de um nó é encaminhada para dois nós, um com probabilidade p e outro com probabilidade $1 - p$, esta probabilidade de distribuição de tráfego permanecerá fixa no tempo.

No âmbito das hipóteses mencionadas previamente, o teorema estabelece que nos casos de redes de filas, cada nó é um sistema independente de filas, com chegadas com distribuição de Poisson determinados pelos princípios de partição (*partitioning*), mixagem (*merging*), e enfileiramento em *tandem* (*tandem queuing*), conforme ilustrado na Fig. 4.2. As médias dos atrasos individuais podem ser somadas para obter o atraso meio total no sistema por enfileiramento.

No sistema de enfileiramento, foram considerados sistemas *M/M/1*, o que pode ser uma boa aproximação para o controlador OpenFlow como é mostrado em [Jarschel et al., 2011]. Num sistema *M/M/1*, as chegadas seguem a distribuição de Poisson, o tempo de serviço segue a distribuição exponencial e só há um servidor para atender os pedidos. Neste caso pode se aplicar uma fórmula bem conhecida para o cálculo dos atrasos:

$$T_r = \frac{Ts}{(1 - \rho)} \tag{4.4}$$

em que T_r é o atraso médio no sistema, T_s é o tempo médio de serviço e ρ é o fator de utilização do sistema. A Fig. 4.3 ilsutra os componentes dos tempos presentes no enfileiramento.

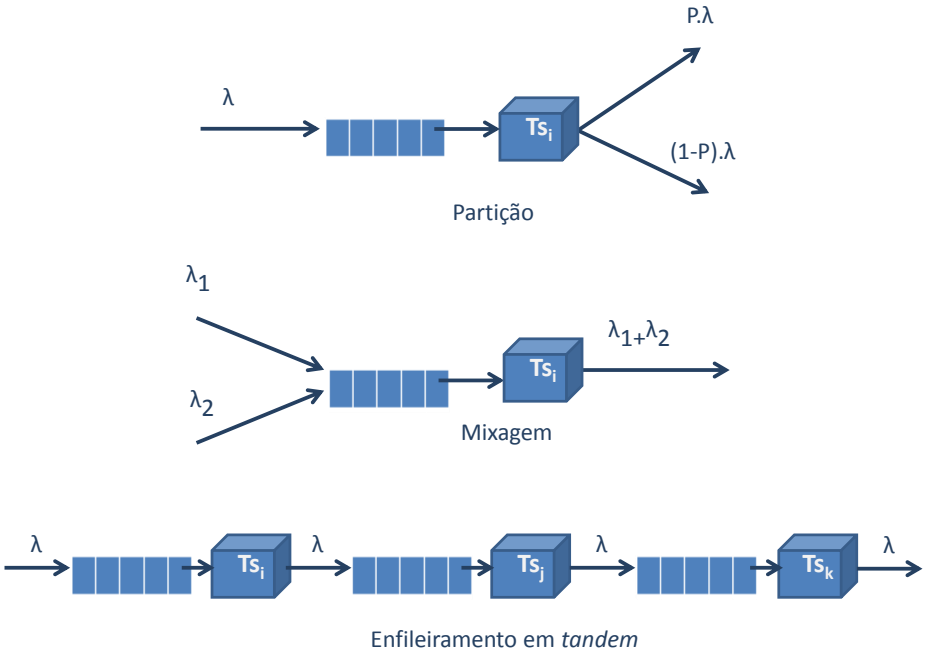

Figura 4.2: Princípios de partição, mixagem e enfileiramento em *tandem*

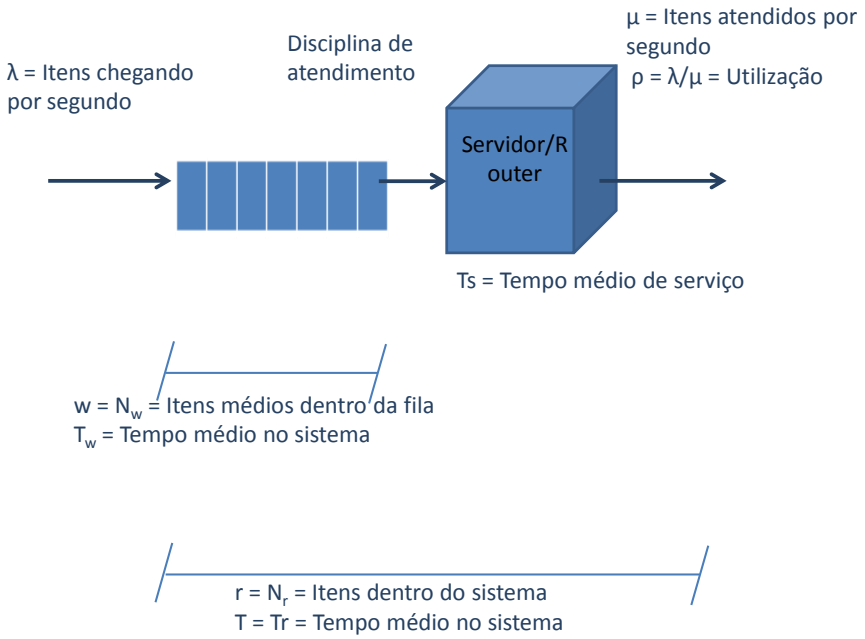

Figura 4.3: Tempos presentes no enfileiramento

4.3 Arquitetura e parâmetros escolhidos para a modelagem

Para os cálculos de atraso, modelou-se uma rede de um ISP com suas interconexões, seus roteadores de núcleo e de distribuição (clientes OpenFlow) e o controlador, conforme ilustrado na

Fig. 4.4. A arquitetura para a modelagem tem quatro POPs interconectados pelos roteadores de núcleo da rede. Um cliente no POP 1 é escolhido como o gerador da consulta OpenFlow de novo fluxo ao controlador.

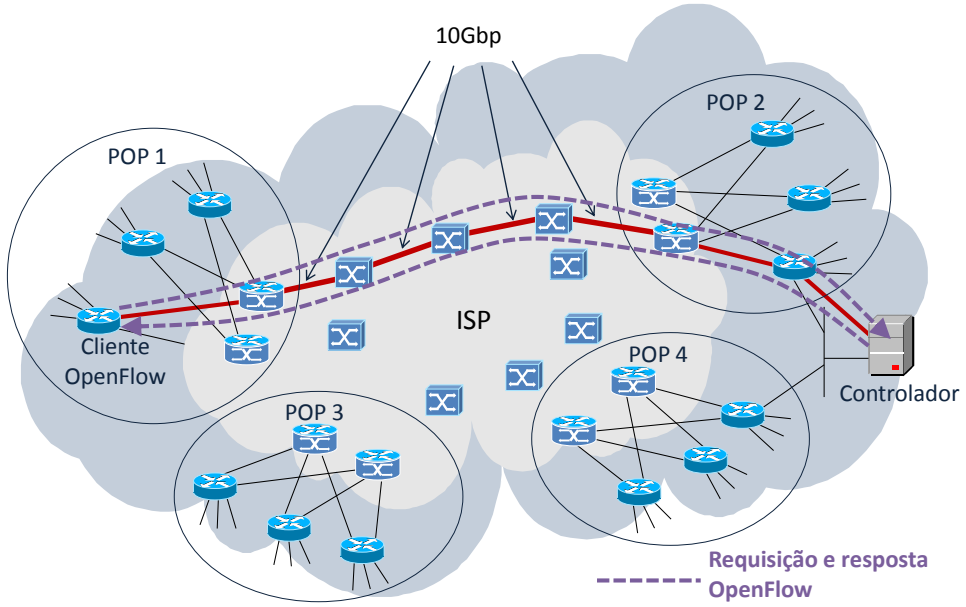

Figura 4.4: Arquitetura escolhida para o cálculo de atrasos

Para se obter a modelagem analítica, foi necessário realizar escolhas de parâmetros que se detalham a seguir:

- Os enlaces da rede entre equipamentos de núcleo têm largura de banda de 10 Gbps, e os enlaces nos POPs têm largura de banda de 1 Gbps;
- Os enlaces de núcleo têm comprimento $L_{Net}/5$ e os que ficam dentro dos POPs têm largura $L_{Net}/100$, em que L_{Net} foi considerado de 1000 Km;
- Na modelagem do enfileiramento dos roteadores, utilizou-se o modelo $M/M/1$ com uma percentagem de utilização de 50% ($\rho_{Rou} = 0,5$). Para o cálculo do tempo médio de serviço (T_s), foi utilizado o fato de que os roteadores dos SPs podem processar a largura de banda total em cada uma de suas interfaces, ainda quando ela transporta pacotes do tamanho mínimo l_{min} de 512 bits (é o pior casso, com o maior número de pacotes por segundo a ser processados). Por isto o T_{sRou} nos roteadores é calculado como:

$$T_{sRou} = \frac{l_{min}}{BW} \tag{4.5}$$

em que BW é a largura de banda do enlace de saída do roteador considerado;

- Para a modelagem do enfileiramento do controlador, foi utilizado o modelo $M/M/1$ com o tempo médio de serviço do controlador, T_{sCon}, igual a 0,24 ms, de acordo com o resultado obtido em [Jarschel et al., 2011]. Adicionalmente, foi considerada uma carga inicial $\rho_{Con} = 0,5$;
- Para a modelagem do enfileiramento do cliente OpenFlow, foi utilizado o modelo $M/M/1$ com o tempo médio de serviço do cliente, T_{sCli}, igual a 2 ms, e uma carga inicial $\rho_{Cli} = 0,5$;
- No cálculo dos atrasos por propagação, foi considerado um índice de refração de 1,5 em todos os casos;
- Para o cálculo do atraso por transmissão das mensagens foram considerados pacotes OpenFlow l_{Of} de 200 Bytes (1600 bits) de tamanho.

4.4 Cálculo de atrasos

4.4.1 Cálculo de atraso por propagação

Como foi estabelecido na seção 4.3, os enlaces de núcleo têm uma largura de $\frac{L_{Net}}{5}$, e de acordo com a equação 4.2, os atrasos por propagação dos enlaces do núcleo são:

$$T_{pN} = \frac{L_{Net}}{(5 * V_p)} \tag{4.6}$$

Analogamente, como foi estabelecido no seção 4.3, os enlaces dentro do POP têm uma largura de $\frac{L_{Net}}{100}$, e de acordo a equação 4.2, os atrasos por propagação dos enlaces do POP são:

$$T_{pP} = \frac{L_{Net}}{(100 * V_p)} \tag{4.7}$$

4.4.2 Cálculo de atraso por transmissão

Para o caso dos enlaces de 1 Gbps e, em acordo à equação 4.3, o atraso por transmissão é:

$$T_{tR1G} = \frac{l_{Of}}{BW_{R1G}} \tag{4.8}$$

em que l_{Of} é o tamanho dos pacotes OpenFlow considerados.

Analogamente, para o caso de enlaces de 10 Gbps, o atraso por transmissão, T_{tR10G}, é:

$$T_{tR10G} = \frac{l_{Of}}{BW_{R10G}} \tag{4.9}$$

4.4.3 Cálculo de atraso por enfileiramento

De acordo as equações 4.4 e 4.5, o atraso médio por enfileramento dos roteadores com enlace de 1 Gbps é:

$$T_{rR1G} = \frac{(l_{min}/BW_{R1G})}{(1-\rho_{R1G})} \tag{4.10}$$

Similarmente, o atraso médio por enfileiramento nos roteadores com enlaces 10 Gbps é:

$$T_{rR10G} = \frac{(l_{min}/BW_{R10G})}{(1-\rho_{R10G})} \tag{4.11}$$

Para o caso do controlador, utilizando a equação 4.4, tem-se um atraso médio de:

$$T_{rCon} = \frac{T_{sCon}}{(1-\rho_{Con})} \tag{4.12}$$

Finalmente, para o cliente OpenFlow, utilizando a equação 4.4, tem-se um atraso médio por enfileramento de:

$$T_{rCli} = \frac{T_{sCon}}{(1-\rho_{Cli})} \tag{4.13}$$

4.4.4 Cálculo do atraso total

O atraso total, T, é o atraso transcorrido entre o envio de uma consulta do cliente OpenFlow ao controlador, solicitando encaminhamento para um novo fluxo ($T_{Cli-Con}$) e a adição do novo fluxo na tabela do cliente OpenFlow mais distante ($T_{Con-Cli}$) (caminho ilustrado em vermelho na Fig.4.4).

$$T = T_{Cli-Con} + T_{Con-Cli} \tag{4.14}$$

Novamente, do caminho ilustrado em vermelho na Fig.4.4, pode se deduzir que:

$$T_{Cli-Con} = 3 * T_{pR1G} + 4 * T_{pR10G} + 3 * T_{rR1G} + 4 * T_{rR10G} + 3T_{tR1G} + 4 * T_{tR10G} + T_{rCon} \tag{4.15}$$

$$T_{Cli-Con} = 3 * T_{pR1G} + 4 * T_{pR10G} + 3 * T_{rR1G} + 4 * T_{rR10G} + 3T_{tR1G} + 4 * T_{tR10G} + T_{rCli} \tag{4.16}$$

e substituindo as equações 4.6 a 4.13 nas equações 4.15 e 4.16, obtém-se:

$$\begin{aligned} T_{Cli-Con} = \frac{3 * L_{Net}}{(100 * V_p)} + \frac{4 * L_{Net}}{(5 * V_p)} + \\ \frac{3 * (l_{min}/BW_{R1G})}{(1-\rho_{R1G})} + \frac{4 * (l_{min}/BW_{R10G})}{(1-\rho_{R10G})} + \\ \frac{3 * l_{Of}}{BW_{R1G}} + \frac{4 * l_{Of}}{BW_{R10G}} + \frac{T_{sCon}}{(1-\rho_{Con})} \end{aligned} \tag{4.17}$$

$$T_{Con-Cli} = \frac{3 * L_{Net}}{(100 * V_p)} + \frac{4 * L_{Net}}{(5 * V_p)} + \frac{3 * (l_{min}/BW_{R1G})}{(1 - \rho_{R1G})} + \frac{4 * (l_{min}/BW_{R10G})}{(1 - \rho_{R10G})} + \frac{3 * l_{Of}}{BW_{R1G}} + \frac{4 * l_{Of}}{BW_{R10G}} + \frac{T_{sCli}}{(1 - \rho_{Cli})} \tag{4.18}$$

Por fim, utilizando a equação 4.14, tem-se a formulação analítica para o atraso total:

$$T = \frac{8 * (l_{min}/BW_{R10G})}{(1 - \rho_{R10G})} + \frac{6 * l_{Of}}{BW_{R1G}} + \frac{6 * L_{Net}}{(100 * V_p)} + \frac{8 * L_{Net}}{(5 * V_p)} + \frac{6 * (l_{min}/BW_{R1G})}{(1 - \rho_{R1G})} + \frac{8 * l_{Of}}{BW_{R10G}} + \frac{T_{sCon}}{(1 - \rho_{Con})} + \frac{T_{sCli}}{(1 - \rho_{Cli})} \tag{4.19}$$

A tabela 4.1 mostra um resumo dos valores numéricos escolhidos para todos os parâmetros.

Substituindo os valores numéricos (tabela 4.1) na equação 4.19, obteve-se um atraso total de $T = 10,819$ ms .

4.5 Estudo das contribuições dos parâmetros variáveis na modelagem

Com a modelagem completa, procedeu-se a analisar o efeito de variações na carga dos roteadores com enlaces de 10 Gbps (ρ_{R10G}), a variação da carga dos roteadores com enlaces de 1 Gbps (ρ_{R1G}) e da carga do controlador (ρ_{Con}) no atraso total da requisição e resposta OpenFlow. Para isso, variaram-se as cargas de um valor inicial de 50% ($\rho = 0,5$) para 100% ($\rho = 1$).

A Fig.4.5 apresenta um gráfico comparativo destas variações de carga. Desta figura, pode-se observar um comportamento exponencial do atraso total com a variação dos parâmetros de carga da rede. Isso indica que, no caso de se utilizar OpenFlow padrão, não só são adicionados tempos no estabelecimento de novos fluxos, também estes tempos variam, tornando-se importantes nos casos de sobrecargas (ρ perto de 1), afetando assim a robustez e o desempenho da rede.

Ressalta-se que existem muitos elementos que podem afetar a comunicação entre o cliente OpenFlow e o controlador, tais como:

- cortes de fibra na rede;
- saturação em qualquer enlace do caminho fim-a-fim;
- não funcionamento de qualquer um dos equipamento de rede do caminho fim-a-fim

Tabela 4.1: Valores dos parâmetros utilizados

Parâmetro	**Valor**
L_{Net}	1.000 km
V_p	200.000 km/s
l_{min}	512 $Bits$
BW_{R10G}	10 $Gbps$
BW_{R1G}	1 $Gbps$
ρ_{R10G}	0.5
ρ_{R1G}	0.5
$rho_{OflowController}$	0.5
T_{sCon}	$0,24$ ms
$rho_{OflowClient}$	0.5
$T_{sOflowClient}$	2 ms
l_{Of}	1600 $Bits$

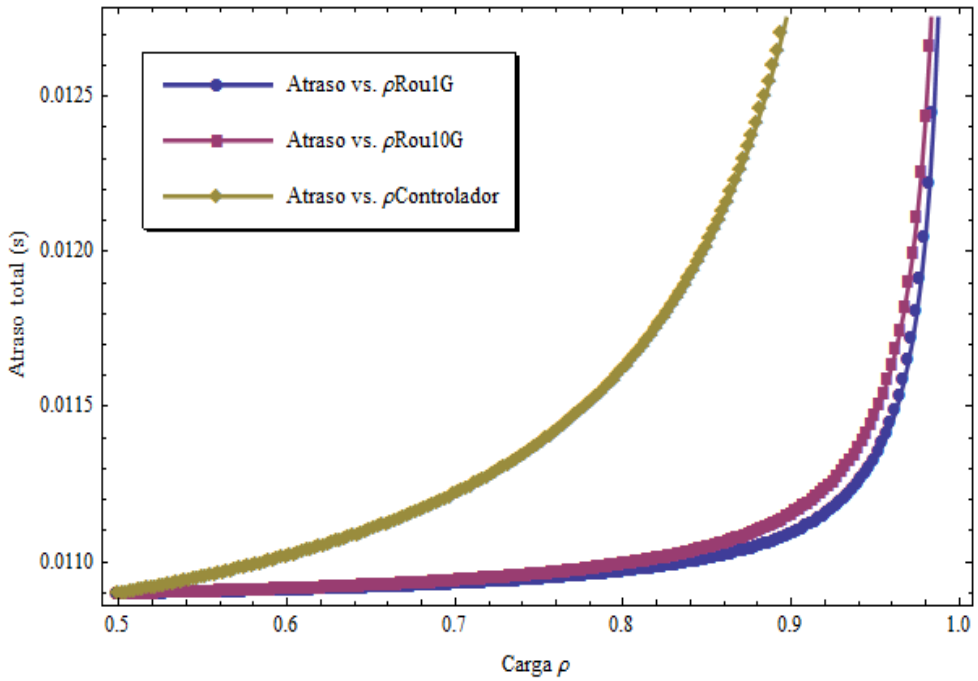

Figura 4.5: Atraso vs. carga

- sobrecarga do controlador;
- enlaces com perdas que gerem retransmissões dos pacotes TCP que intervém na comunicação OpenFlow;
- queda do contolador;

- enlaces com menor largura de banda que a utilizada em nossos cálculos.

Por fim, destaca-se que os cálculos apresentados neste capítulo permitem quantificar analiticamente os atrasos intervenientes na criação de novos fluxos para as implementações que utilizam OpenFlow padrão, bem como quantificar como as variações dos parâmetros da rede afetam estes tempos. Nas arquiteturas propostas no trabalho, os atrasos na criação de novos fluxos não existem, pois os pacotes sempre são encaminhados imediatamente, sem aguardar a resposta do controlador. Portanto, este capítulo destaca um dos principais benefícios das arquiteturas propostas.

Capítulo 5

Protótipo e Comparativo Das Arquiteturas

5.1 Softwares utilizados

Para a realização do protótipo desenvolvido neste capítulo, foram utilizados Mininet para a construção da topologia, Open vSwitch como software dos clientes OpenFlow, e Ryu para o controlador. Nas seguintes subseções, cada um deles é descrito.

5.1.1 Mininet

Mininet [Mininet v.2.1, 2012] [Lantz et al., 2010] é um emulador de rede capaz de criar uma topologia virtual com *hosts*, *switches*, controladores e enlaces virtuais. Mininet executa um software de rede padrão do Linux e seus switches executam OpenFlow.

Este emulador de rede é um projeto *open source* que permite criar uma rede de provas completa sobre um só computador, possibilitando assim a pesquisa, o desenvolvimento, a aprendizagem e a criação de protótipos e testes, sem nenhum custo. Adicionalmente Mininet otimiza o comportamento do sistema anfitrião, ainda quando é utilizado para grandes topologias. Estas topologias podem ser compostas por um grande número de *hosts*, *switches* OpenFlow, enlaces virtuais e controladores.

As redes Mininet executam um código real, incluindo aplicações de redes Unix/Linux padrão, assim como o *kernel* de Linux e a pilha de rede. É por isso que Mininet proporciona muito mais que um simples simulador, pois um projeto ou código desenvolvido e testado em Mininet, seja para um controlador, *switches* OpenFlow ou *hosts*, pode ser colocado num sistema real praticamente sem a necessidade de fazer mudanças. Mininet combina muitas das melhores características de emuladores, redes experimentais em *hardware* e simuladores.

Na comparação com abordagens baseadas em virtualização completa do sistema, Mininet tem as vantagens indicadas a seguir:

- inicia mais rápido (só em segundos);
- possibilita a criação de topologias com uma grande dimensão, pudendo alcançar centenas de *hosts* e *switches* interligados;
- fornece uma maior utilização da largura de banda;
- é mais fácil de ser instalado.

No caso em que é comparado com redes experimentais em *hardware*, Mininet tem as vantagens indicadas a seguir:

- é gratuito e está sempre disponível;
- possibilita que as topologias sejam reconfiguradas de uma forma muito rápida.

Finalmente se a comparação é realizada com simuladores, Mininet tem as vantagens indicadas a seguir:

- executa o código real, sem modificações, incluindo o código de aplicativo, o código do núcleo (*kernel*) do sistema operacional e o código do plano de controle (o código do controlador OpenFlow e o código de Open vSwitch);
- constrói topologias que se podem conectar facilmente às redes reais;
- tem possibilidade de crescimento muito grande.

A Fig. 5.1 faz um descrição em forma simplificada da criação por Mininet de uma topologia básica, formada por dois hosts, um *switch* OpenFlow que os interconecta e um controlador. Inicialmente, o lançador de Mininet cria dois processos *bash* que representam o *Host1* e o *Host 2*, cada um deles com seu próprio espaço de nomes. Seguidamente, são criadas duas duplas de *ethernet* virtuais, e são designados nomes do *name space*. Posteriormente, é criado um *switch* OpenFlow para interconectar os hosts e finalmente é criado um controlador OpenFlow com o canal de controle estabelecido.

É importante notar que o controlador que está incluído dentro do Mininet é o controlador NOX, só com suporte para OpenFlow 1.0. Por esta razão, para a criação de nosso protótipo, foi necessário instalar o controlador externo Ryu, com suporte para OpenFlow 1.3. Adicionalmente, o software utilizado por Mininet para a virtualizacão dos switches só implementa a versão 1.0 de OpenFlow, pelo qual também foi modificado para o protótipo, utilizando Open vSwitch com suporte para OpenFlow 1.3.

5.1.2 Open vSwitch

Open vSwitch [Pfaff et al., 2009] [Open vSwitch v.2.0, 2012] é um software de switch multicamada construído para ambientes virtuais e que utiliza a licença de código aberto Apache 2. O

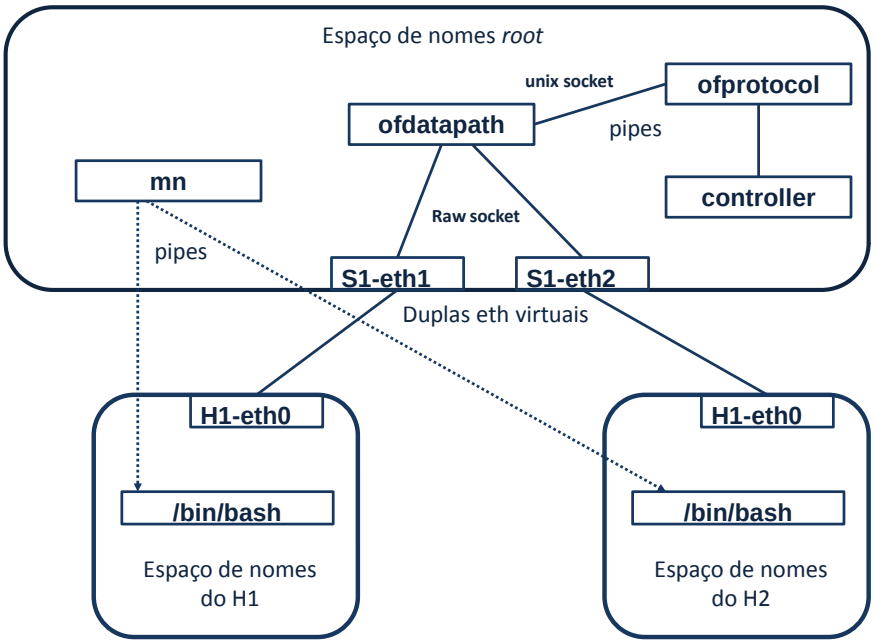

Figura 5.1: Arquitetura exemplo de Mininet

objetivo de Open vSwitch é implementar uma plataforma para switches de qualidade que seja capaz de dar suporte a interfaces de gerenciamento padrão e ser aberto para as funções de controle e roteamento programáveis.

Este software reside dentro do *hypervisor* ou domínio de gerenciamento e fornece conectividade entre as máquinas virtuais e as interfaces físicas. A arquitetura geral de Open vSwitch pode ser observada na Fig. 5.2.

Open vSwitch fornece uma interface local de gerenciamento através da qual a camada de virtualização pode manipular a configuração topológica. Isto inclui a criação de *switches* (vários *switches* virtuais podem ser criados em um único *host* físico), gerenciamento de conectividade das interfaces virtuais (para cada interface virtual é adicionada uma porta lógica em correspondência) e gerenciamento da conectividade com as interfaces físicas.

Adicionalmente o Open vSwitch fornece uma interface que permite a configuração dos *switches* virtuais, similar às interfaces de gerenciamento dos switches físicos. Também possibilita a manipulação remota do plano de dados de seus switches virtuais (tabelas de fluxos), especificando como processar os pacotes com base nos cabeçalhos de camadas 2, 3 e 4. A interface para a manipulação das tabelas é implementada por meio do protocolo OpenFlow.

O código é escrito em plataforma C independente e é fácil de ser transportado para outros ambientes. Dentro das características mais importantes que Open vSwitch dá suporte estão:

- visibilidade em comunicações entre máquinas virtuais;

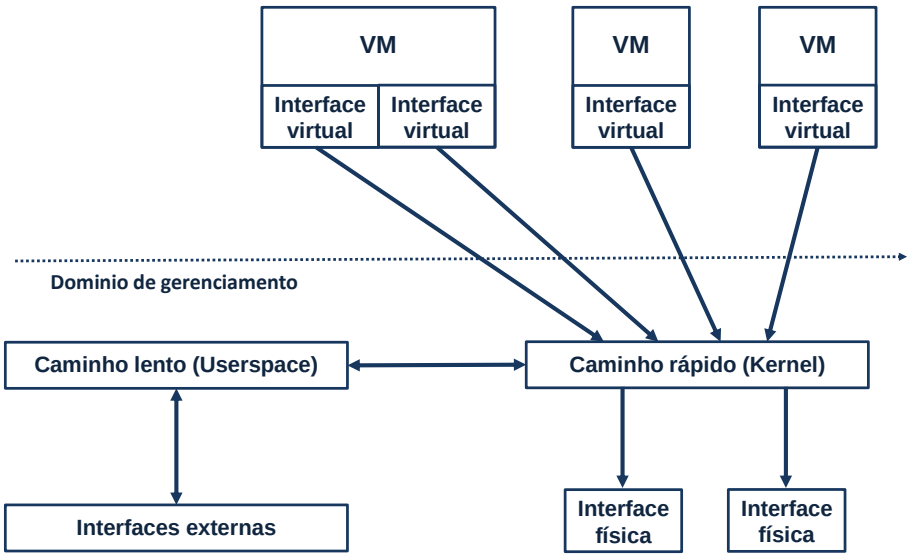

Figura 5.2: Arquitetura de Open vSwitch

- modelo padrão IEEE 802.1Q;
- STP(*Spanning Tree Protocol*);
- granularidade para o controle de QoS;
- controle de tráfego por cada interface de máquina virtual;
- suporte a OpenFlow em suas diferentes versões;
- IPv6 (Internet Protocol v6);
- linha de processamento com múltiplas tabelas e com mecanismos de cache de fluxos.

5.1.3 Controlador Ryu

O controlador Ryu [Ryu v.3.3, 2012] é um componente criado para o contexto das redes definidas por software. Todo seu código está disponível gratuitamente com a licença de Apache 2.0 e está completamente escrito na linguagem de programação Python, facilitando, assim, a criação de novas aplicações de controle (Fig 5.3).

Ryu fornece componentes de software com APIs muito bem definidas, o que facilita aos desenvolvedores a criação de novas formas de gerenciamento de redes e aplicações de controle. Ryu dá suporte a vários protocolos para o gerenciamento dos equipamentos de rede como Netconf, OF-config, OpenFlow. Particularmente, para este último dá suporte às versões 1.0, 1.2 e 1.3, característica essencial para sua utilização no protótipo desenvolvido neste trabalho.

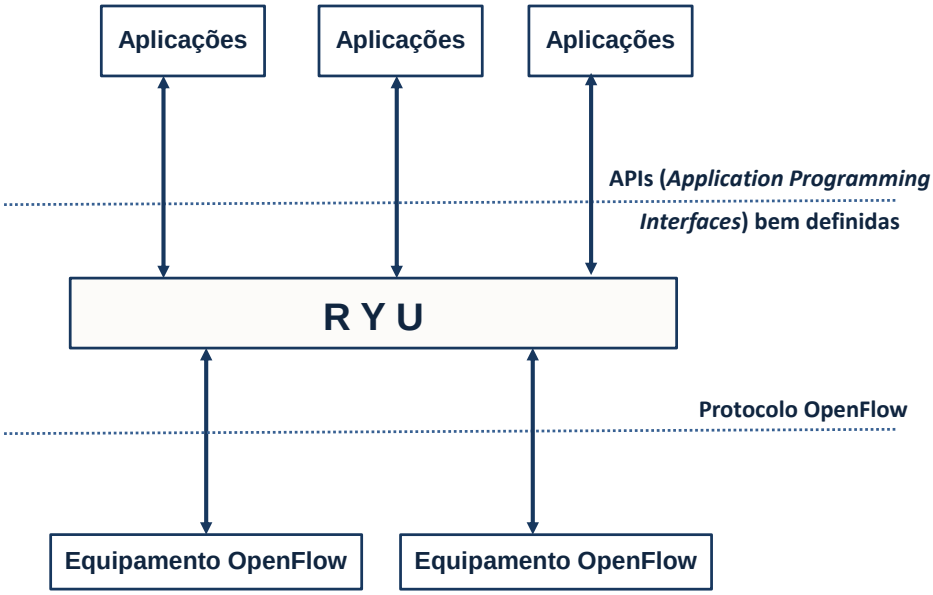

Figura 5.3: Controlador Ryu

5.2 Protótipo

5.2.1 Descrição geral

Para exemplificar a arquitetura proposta, foi criado um protótipo da topologia mostrada na Fig. 5.4. Ele é composto por três POPs (*Point Of Presence*), cada um com dois equipamentos de núcleo (SCxx), dois equipamentos de borda (SBxx) e três hosts. Adicionalmente, cada equipamento de núcleo encontra-se conectado a outros três equipamentos de núcleo e aos equipamentos de borda de seu POP.

Para a construção da topologia, foram utilizados os *softwares* descritos na seção 5.1, todos com suporte para OpenFlow 1.3:

- Mininet versão 2.1 [Mininet v.2.1, 2012] [Lantz et al., 2010], capaz de criar uma topologia virtual completa, com centenas de *hosts* e *switches* interligados em um só computador. Adicionalmente, Mininet combina muitas das melhores características de emuladores, redes experimentais em *hardware* e simuladores;
- o Open vSwitch versão 2.0 [Open vSwitch v.2.0, 2012] [Pfaff et al., 2009]: software de switch multicamada construído para ambientes virtuais e com licença de código aberto Apache 2.0;
- o controlador Ryu versão 3.3 [Ryu v.3.3, 2012]: Componente de software criado para o contexto das redes definidas por software, com a licença de código aberto Apache 2.0. Ele tem

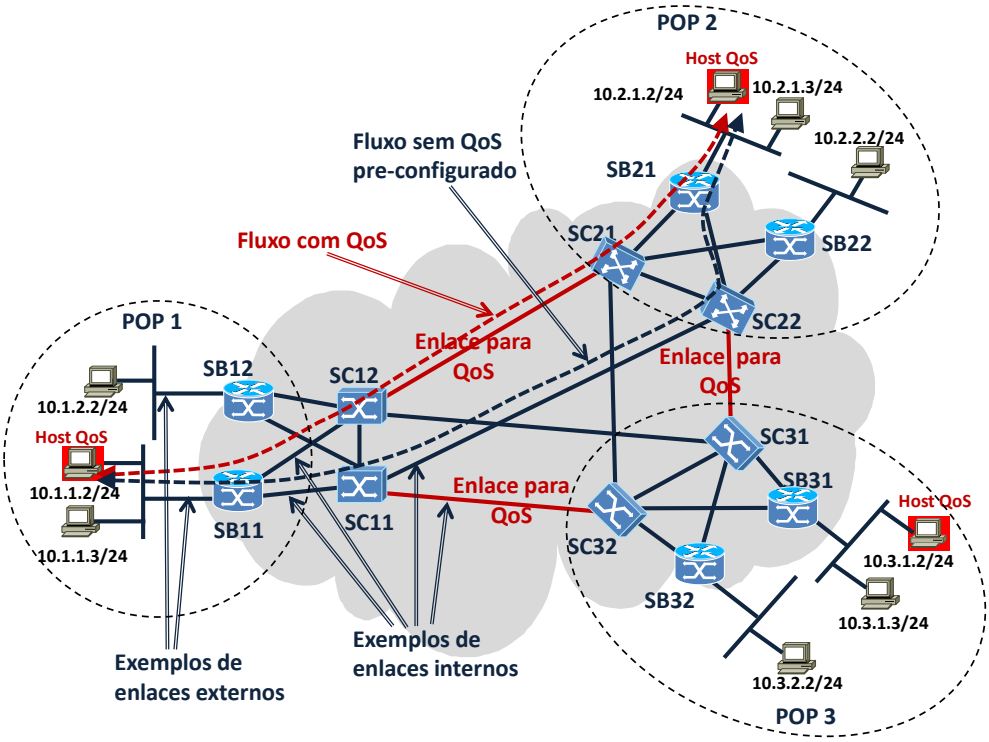

Figura 5.4: Topologia do Protótipo

APIs muito bem definidas, o que facilita para os desenvolvedores a criação de novas formas de gerenciamento de redes e aplicações de controle.

Sem perda de generalidade, decidiu-se implementar a topologia sem MPLS, com o objetivo de facilitar a implementação das regras e reduzir o número de tabelas a serem utilizadas. A ausência de MPLS não compromete a verificação do comportamento da rede SDN-OpenFlow e torna mais fácil a sua compreensão.

Na arquitetura, os pacotes de tráfego gerados pelos hosts com endereço IP 10.x.1.2 (x variando entre 1 e 3, de acordo com o respectivo POP) são pacotes com QoS e possuem o campo DSCP igual a 5 (prioritário). Quando um fluxo com QoS específico é criado, este tráfego é encaminhado entre os diferentes POPs através dos enlaces para QoS ilustrados na Fig. 5.4. Por outro lado, o tráfego que não utiliza as regras específicas de QoS é encaminhado através dos enlaces restantes.

As regras (*flow entry*) de coincidência dentro das tabelas são criadas, em caso de não especificar outro valor, com prioridade 32768, e são processadas na ordem que foram inseridas. Se for requerido que algumas regras de coincidência (como as linhas específicas QoS) sejam examinadas antes de outras linhas, elas devem ter maior prioridade.

5.2.2 Construção das regras gerais

O número escolhido para a implementação da tabela QoS foi o 5, e para a tabela de encaminhamento interno foi o 10. Com esta escolha é possível que regras dentro da tabela QoS encaminhem

o pacote à tabela de encaminhamento interno.

A seguir são descritas as regras que o controlador adiciona inicialmente (proativamente) para implantar a lógica definida nas diferentes tabelas:

- **Tabela 0:**
  - Nos equipamentos SBx1, são inseridas as regras para a interconexão dos hosts da mesma rede local - regras de camada 2 - com prioridade 40.000.
  - Em todos os equipamentos são adicionadas duas regras: uma para instruir que os pacotes entrantes marcados com DSCP = 0 sejam enviados à tabela 10 (tabela sem QoS), e outra para que os pacotes marcados com DSCP = 5 sejam enviados à tabela 5 (tabela com QoS).
- **Tabela 10:**
  - Esta tabela contém as regras gerais para encaminhar os pacotes a todas as redes destino (tabela similar à criada pelos protocolos de roteamento interno atuais). A tabela 10 encaminha independentemente do DSCP estabelecido.
- **Tabela 5:**
  - Nos equipamentos de borda, são inseridas regras de QoS gerais que só geram coincidência quando o pacote entra por uma porta externa (equipamento de borda de entrada na rede). Esta regra envia o pacote por duas vias: ao controlador (**instrução *Apply-actions***, porta reservada ***Controller***) e à tabela 10 (***Goto-Table 10***) para o encaminhamento imediato.
  - Também nos equipamentos de borda são inseridas regras de QoS gerais como última alternativa, que no caso de não ter coincidência com as regras do parágrafo anterior ou com regras específicas, envia o pacote somente à tabela 10.
  - Nos equipamentos de núcleo SCxx, é adicionada somente a regra relacionada à *table miss*, situação em que não há correspondência entre o pacote e a entrada de fluxos (*flow entry*) na tabela de fluxos. A regra indica que os pacotes sejam encaminhados à tabela 10 até que o controlador adicione os fluxos com QoS específicos solicitados pelos equipamentos de borda.

5.2.3 Modo de operação e regras de QoS específicas

A seguir, é descrito o modo de funcionamento:

- Os fluxos sem QoS (DSCP = 0) são processados em cada *switch* inicialmente pela tabela 0, a qual os envia à tabela 10 para o seu encaminhamento, conforme ilustrado na Fig. 5.5, opção 2 (OP. 2).

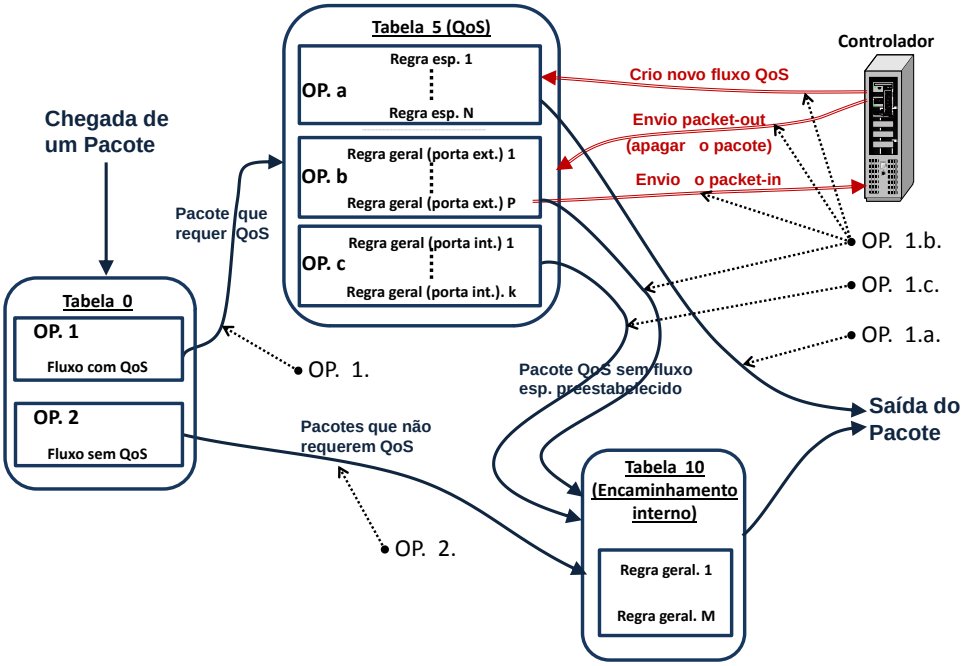

Figura 5.5: Arquitetura do Protótipo

- Os fluxos com QoS (DSCP = 5) são processados em cada *switch* inicialmente pela tabela 0, a qual os envia à tabela 5, conforme ilustrado na Fig. 5.5, opção 1 (OP. 1).

- Nos equipamentos SBxx, quando não existir ainda fluxo específico e houver coincidência com a regra geral de portas de entrada externas (Fig. 5.5, op. 1.b), o SBxx envia uma mensagem *packet-in* ao controlador e encaminha imediatamente o pacote pela tabela 10. O controlador recebe a mensagem *packet-in* do switch de borda de entrada à rede, calcula o caminho ótimo e cria um fluxo específico com *idle-timeout* e prioridade 45.000 nas tabelas 5 de todos os equipamentos intervenientes no caminho fim a fim e em sentido oposto ao do fluxo. Adicionalmente, envia um *packet-out* como resposta ao equipamento de borda que realizou a solicitação, indicando que deve retirar o pacote (ele já foi encaminhado pela tabela 10).

- Nos equipamentos SBxx, quando não existir ainda fluxo específico e não houver coincidência com regra geral de portas de entrada externas (Fig. 5.5, OP. 1.c.), o SBxx envia o pacote à tabela 10.

- Nos equipamentos SCxx, quando não existir ainda fluxo específico, o pacote é sempre encaminhado à tabela 10.

- Em qualquer equipamento, no caso de existir regra específica para o pacote entrante (Fig. 5.5, OP. 1.a.), ele será encaminhado por esta regra de QoS.

5.2.4 Cenários de testes

5.2.4.1 Teste 1

Para testar o comportamento na rede, comprovou-se inicialmente que todos os destinos eram alcançáveis utilizando a tabela 10. Posteriormente, testou-se o funcionamento do protótipo através da geração de um fluxo com QoS de seis pacotes de ida (na direção SB11 a SB21) e seis pacotes de volta (na direção SB21 a SB11) entre os *hosts* com endereços IP 10.1.1.2 e 10.2.1.2, e com intervalo entre pacotes de 1 s. Neste caso, comprovou-se que o primeiro pacote de ida e volta segue o fluxo sem QoS, e os cinco pacotes restantes seguem o fluxo específico com QoS (fluxos indicados na Fig.5.4).

Na Fig. 5.6 são apresentadas as tabelas de fluxos reais do equipamento SB11 na prova mencionada no parágrafo anterior, com suas linhas reagrupadas e sem as linhas de camada 2 para facilitar a visualização e compreensão. Como todos os pacotes do fluxo são gerados com tipo de serviço (*type of service*, ToS) igual a 20 (DSCP=5), os seis pacotes de ida e os seis de volta vão coincidir com a regra geral para pacotes com QoS que se encontra na linha 1 da tabela 0 na Fig. 5.6 (*cookie=0x0, duration=40.55s, table=0,* ***n_packets=12****, n_bytes=1176, ip,nw_tos=20 actions=goto_table:5*) para o equipamento SB11, gerando 12 coincidências e enviando os pacotes à tabela de QoS (tabela 5).

Quando o primeiro pacote do fluxo gerado pelo host 10.1.1.2 chegar ao equipamento SB11, sua tabela 5 ainda não possui fluxo específico pré-configurado e o pacote entra por uma porta externa (as portas 3 e 4 do SB11 são externas na topologia); então, o pacote encontra coincidência com a regra QoS geral com porta de entrada externa (linha 3, tabela 5, Fig.5.6). Esta regra indica que o pacote tem que ser encaminhado por duas linhas de processamento: ao controlador e à tabela 10 (*actions=CONTROLLER:65535,goto_table:10*). Este último encaminhamento gera uma nova coincidência na linha 1, tabela 10, Fig. 5.6. Posteriormente, quando a resposta ao primeiro pacote do fluxo proveniente do host 10.2.1.2 chegar ao SB21, este equipamento realiza o mesmo procedimento executado pelo SB11, enviando o pacote por duas linhas de processamento; uma delas encaminha o pacote por o SB11, encontrando correspondência com a linha geral de QoS para pacotes que não entram por uma porta externa (Fig. 5.6, tabela 5, linha 5), enviando o pacote somente à Tabela 10 para seu encaminhamento final (gerando uma nova coincidência na linha 1, tabela 10, Fig 5.6).

Quando os cinco pacotes sucessivos do fluxo gerados pelo host 10.1.1.2 chegarem ao SB11, este equipamento já possui os fluxos QoS específicos criados pelo controlador. Então, os pacotes encontram correspondência com a regra 2 da tabela 5 na ida do pacote e com a regra 1 da tabela 5 na volta (ambos os fluxos com prioridade 45.000, Fig. 5.6), sendo encaminhados diretamente por regras específicas de QoS. Portanto, com o teste descrito acima, verificou-se o funcionamento da arquitetura proposta, obteve-se o encaminhamento inicial imediato utilizando as regras pré-configuradas na tabelas 10 e verificou-se que nenhum pacote é perdido no momento que o fluxo QoS é reencaminhado.

TABELA 0

1. cookie=0x0, duration=40.55s, table=0, **n_packets=12**, n_bytes=1176, ip,nw_tos=20 actions=goto_table:5
2. cookie=0x0, duration=40.53s, table=0, n_packets=0, n_bytes=0, ip,nw_tos=0 actions=goto_table:10

TABELA 5

1. cookie=0x0, duration=9.966s, table=5**, n_packets=5**, n_bytes=490, idle_timeout=60, priority=45000,ip,nw_src=10.2.1.2,nw_dst=10.1.1.2,nw_tos=20 actions=dec_ttl,mod_dl_dst:00:00:00:01:01:02,output:3
2. cookie=0x0, duration=9.97s, table=5**, n_packets=5**, n_bytes=490, idle_timeout=60, priority=45000,ip,nw_src=10.1.1.2,nw_dst=10.2.1.2,nw_tos=20 actions=dec_ttl,output:2
3. cookie=0x0, duration=38.891s, table=5, **n_packets=1**, n_bytes=98, priority=35000,ip,in_port=3,nw_tos=20 actions=CONTROLLER:65535,goto_table:10
4. cookie=0x0, duration=38.783s, table=5, n_packets=0, n_bytes=0, priority=35000,ip,in_port=4,nw_tos=20 actions=CONTROLLER:65535,goto_table:10
5. cookie=0x0, duration=38.997s, table=5, **n_packets=1**, n_bytes=98, ip,nw_tos=20 actions=goto_table:10

TABELA 10

1. cookie=0x0, duration=40.215s, table=10**, n_packets=1**, n_bytes=98, ip,nw_dst=10.1.1.2 actions=dec_ttl,mod_dl_dst:00:00:00:01:01:02,output:3
2. cookie=0x0, duration=40.337s, table=10, n_packets=0, n_bytes=0, ip,nw_dst=10.1.2.2 actions=dec_ttl,output:1
3. cookie=0x0, duration=40.237s, table=10, n_packets=0, n_bytes=0, ip,nw_dst=10.1.1.3 actions=dec_ttl,mod_dl_dst:00:00:00:01:01:03,output:4
4. cookie=0x0, duration=39.751s, table=10, n_packets=0, n_bytes=0, ip,nw_dst=10.3.1.0/24 actions=dec_ttl,output:2
5. cookie=0x0, duration=39.778s, table=10, **n_packets=1**, n_bytes=98, ip,nw_dst=10.2.1.0/24 actions=dec_ttl,output:1
6. cookie=0x0, duration=39.737s, table=10, n_packets=0, n_bytes=0, ip,nw_dst=10.3.2.0/24 actions=dec_ttl,output:2
7. cookie=0x0, duration=39.764s, table=10, n_packets=0, n_bytes=0, ip,nw_dst=10.2.2.0/24 actions=dec_ttl,output:1

Figura 5.6: Tabelas reais de fluxo do equipamento SB11 reagrupadas (teste 1)

5.2.4.2 Teste 2

Neste segundo caso, foi simulada uma queda do controlador e durante a queda, foi gerado um novo fluxo entre os *hosts* com endereços IP 10.1.1.2 e 10.2.1.2. Posteriormente, restabeleceu-se o controlador para que o fluxo específico fosse criado por ele.

A Fig. 5.7 mostra a tabela de fluxos real no equipamento SB11 do teste mencionado no parágrafo anterior, obtido mediante a execução do comando: ***sudo ovs-ofctl O OpenFlow13 dump-flows sb11***. Similarmente à seção 5.2.4.1, são apresentadas as tabelas de fluxos com suas linhas reagrupadas e sem as linhas de camada dois, facilitando assim sua visualização e compreensão (Fig. 5.8).

Neste caso foram gerados 81 pacotes QoS no total, todos com DSCP=5, com coincidências na regra 1 da tabela 0 da Fig. 5.8. Destes 81 pacotes, 45 foram durante a queda do controlador (23 de ida e 22 de volta), e os 36 restantes (18 de ida e 18 de volta) aconteceram quando o controlador se encontrava em serviço .

É importante notar que o pacote 24 de ida e o pacote 23 de volta são os primeiros pacotes que

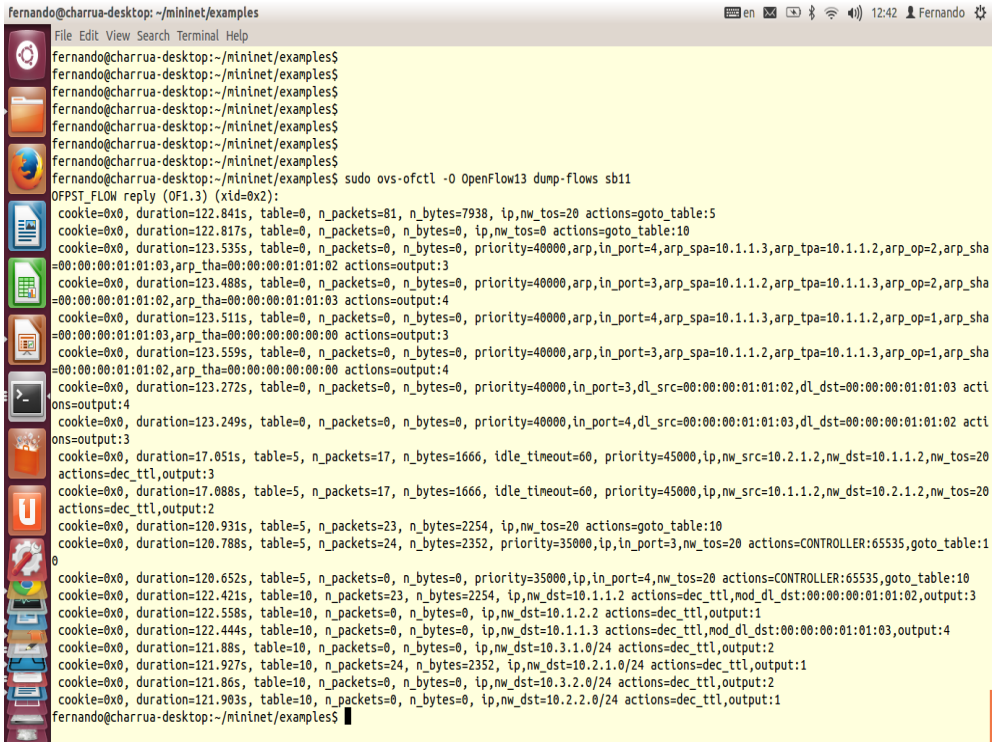

Figura 5.7: Tabelas reais sem reagrupamento no equipamento SB11

geram uma consulta ao controlador em serviço, isto é, que são os primeiros que têm resposta do controlador, criando assim os respectivos fluxos específicos. Contudo, estes pacotes (o 24 de ida e o 23 de volta) ainda são encaminhados pelas regras QoS gerais, pois ainda não se encontram configurados os fluxos específicos. Os primeiros 24 pacotes de ida são, portante, encaminhados pela regra geral QoS com portas de entrada externa (Fig. 5.8, tabela 5, linha 4), o qual encaminha os pacotes simultaneamente à tabela 10 (Fig. 5.8. tabela 10, linha 5) e ao controlador. Analogamente, os primeiros 23 pacotes de volta serão encaminhados pela regra geral QoS, com portas de entrada internas (Fig. 5.8, tabela 5, linha 3), regra que encaminha os pacotes somente à tabela 10 (Fig. 5.8, tabela 10, linha 1).

Os 17 pacotes seguintes de ida e os 17 de volta já têm o fluxo QoS específico configurado na tabela QoS (Fig 5.8, tabela 5, linhas 1 e 2). Assim, serão encaminhados diretamente por esta regra, sem a necessidade de serem processados pela tabela de encaminhamento interno (tabela 10).

Este teste permite resaltar a robustez da arquitetura proposta, mostrando como os novos fluxos são encaminhados, ainda durante uma queda do controlador. Adicionalmente, quando o controlador volta ao serviço, ele configura as regras específicas ótimas, melhorando assim o encaminhamento para o tráfego QoS.

5.2.4.3 Testes adicionais

Em seguida, foram adicionados atrasos na resposta do controlador, mostrando que o pacote continua sendo encaminhado pela tabela 10 até que a regra específica seja criada. Para o caso da

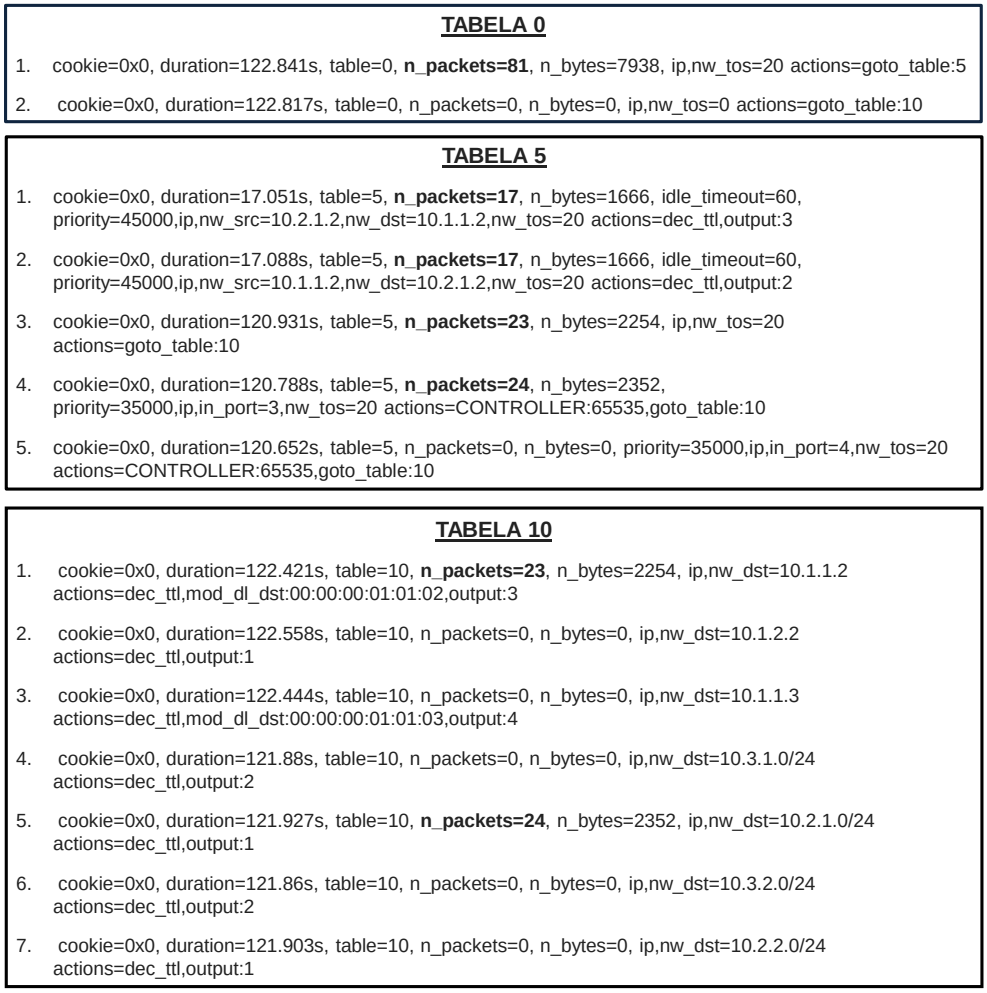

Figura 5.8: Tabelas reais de fluxo do equipamento SB11 reagrupadas (teste 2)

arquitetura proposta, o atraso dos primeiros pacotes é muito baixo e independente do tempo de resposta do controlador, enquanto que para a arquitetura OpenFlow padrão o atraso dos primeiros pacotes é diretamente proporcional ao atraso do controlador.

Por fim, foi emulada uma nova queda do controlador, mostrando que as regras QoS preestabelecidas renovam o *idle-timeout* no caso de continuar sendo utilizadas, mas no caso de não ser utilizadas, são apagadas depois do *idle-timout* (considerado 60s).

5.2.4.4 As mensagens OpenFlow

Nesta seção, são descritos com maior profundidade os pacotes trocados entre os elementos de rede e o controlador. Para uma melhor visualização, foi utilizado o analisador de protocolo *Wireshark*, que foi atualizado para implementar a versão OpenFlow 1.3.

Os fluxos utilizados nas seções anteriores foram gerados utilizando ICMP, mas poderiam ter sido gerados com qualquer outro protocolo. A seguir, serão mostrados exemplos de troca das mensagens para o estabelecimento dos fluxos QoS, que são encontradas em qualquer um dos testes previamente

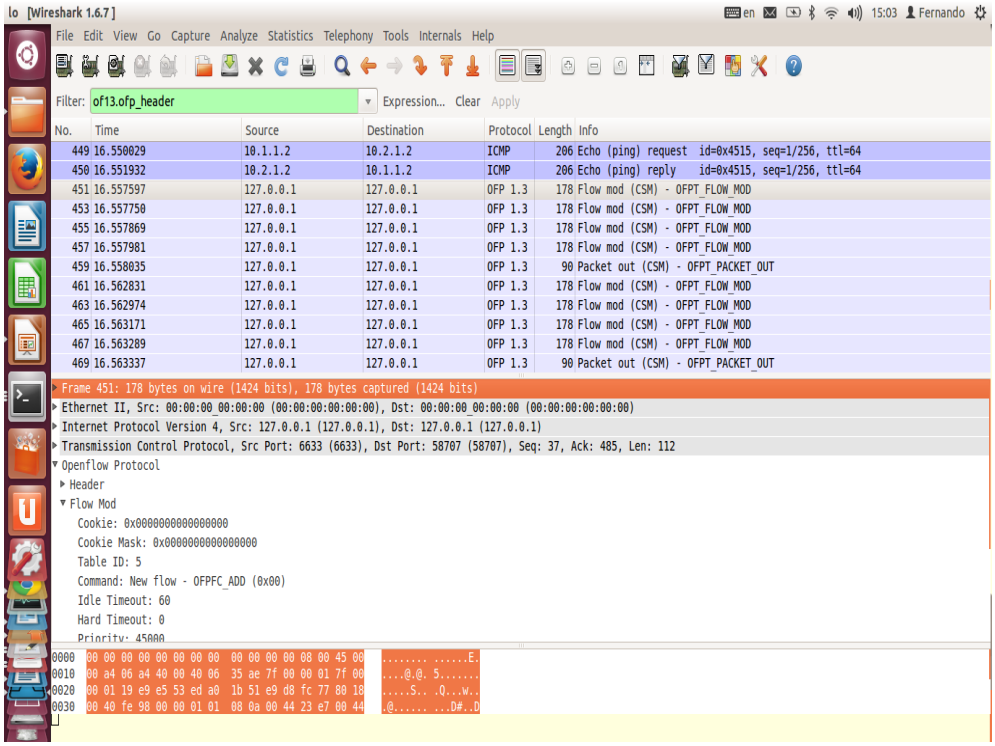

Figura 5.9: Mensagens trocadas na criação de um novo fluxo QoS específico

desenvolvidos. Todos eles geram a mesma sequência de mensagens trocadas entre o controlador e os equipamentos de rede e todos eles utilizam a geração de um fluxo ICMP com DSCP estabelecido em 5, entre os dispositivos com *IP:10.1.1.2* (SB11) e *IP:10.1.2.2* (SB21), representando, assim, um fluxo que requer encaminhamento QoS.

A Fig. 5.9 mostra uma janela do *Wireshark*, com uma captura realizada, em que podem ser observadas duas mensagens *Packet-In*, que representam a solicitação de um novo fluxo em ambas as direções: uma no sentido de ida, gerada pelo equipamento SB11, e outra no sentido de volta, gerada pelo equipamento SB21 (Fig. 5.4). Posteriormente são mostradas quatro mensagens *Flow-mod*, que são as mensagens utilizadas pelo controlador para adicionar ou modificar fluxos. Estas mensagens são enviadas a cada um dos equipamentos que se encontram dentro do caminho ótimo de ida, para assim configurar o fluxo QoS (no exemplo: SB11, SC12, SC21 e SB21. Fig. 5.4).

Posteriormente, é enviada pelo controlador uma mensagem *Packet-Out* para indicar que o equipamento que gerou a consulta (SB11) tem que ignorar o pacote, dado que o pacote já foi encaminhado pelo encaminhamento interno. A seguir, são enviadas outras quatro mensagens pelo controlador, necessárias para configurar o fluxo específico QoS para o caminho de volta (SB21, SC21, SC12 e SB11). Finalmente, é enviada uma mensagem *Packet-Out* pelo controlador ao equipamento que gerou a consulta de volta (SB21), indicando-lhe que o pacote tem que ser ignorado, dado que o pacote também neste caso foi encaminhado pelo encaminhamento interno.

Nas subseções seguintes, são visualizados e descritos o conteúdo de cada um dos pacotes intervenientes na criação do fluxo específico. As mensagens foram obtidas mediante a utilização de *Wireshark*.

5.2.4.4.1 *Packet-in* .

A Fig. 5.10 mostra o conteúdo do *Packet-In* que o equipamento SB11 envía ao controlador, ressaltando os campos da mensagem mais importantes para o trabalho. Da Fig. 5.10, podem ser destacados em negrito os pontos a seguir:

Frame 7350: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits)
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
Transmission Control Protocol, Src Port: 37943 (37943), **Dst Port: 6633 (6633)**, Seq: 729, Ack: 389, Len: 140
Openflow Protocol
Header
Version: 0x04
Type: Packet in (AM) - OFPT_PACKET_IN (10)
Length: 140
Transaction ID: 0
Packet in
Buffer ID: 0xffffffff
Total length: 98
Reason: Action explicitly output to controller - OFPR_ACTION (0x01)
Table ID: 0x05
Cookie: 0x0000000000000000
Match
Type: OpenFlow Extensible Match - OFPMT_OXM (0x0001)
Length: 12
OXM field
Class: Basic class for OpenFlow - OFPXMC_OPENFLOW_BASIC (0x8000)
0000 000. = Field: **Switch input port** - OFPXMT_OFB_IN_PORT (0x00)
0 = Has mask: No
Length: 4
Value: 00000003
Padding
Padding
Ethernet II, Src: 00:00:00_01:01:02 (00:00:00:01:01:02), Dst: Private_01:01:01 (01:01:01:01:01:01)
Internet Protocol Version 4**, Src: 10.1.1.2 (10.1.1.2), Dst: 10.2.1.2 (10.2.1.2)**
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x14 (**DSCP 0x05**)
Total Length: 84
Identification: 0x0000 (0)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: ICMP (1)
Header checksum: 0x248f [correct]
Source: 10.1.1.2 (10.1.1.2)
Destination: 10.2.1.2 (10.2.1.2)
Internet Control Message Protocol
Type: 8 (Echo (ping) request)

Figura 5.10: Mensagem *Packet-In*

- o pacote que contém a mensagem *Packet-In* não tem IPs origem nem destino. Isto ocorre porque a comunicação entre o controlador e os elementos de rede no protótipo é estabelecida por um canal lógico interno. É importante notar que esta característica é resultado da configuração escolhida; no protótipo, o controlador e os equipamentos se encontram no mesmo computador. Também é possível definir um controlador externo e configurar os equipamentos de rede para acessar ao controlador. Neste último caso, as mensagens OpenFlow trocadas entre o controlador e os equipamentos de rede teriam endereço IP;
- a porta destino é a *6633*, a qual é utilizada pelo protocolo OpenFlow para o envio das mensagens dirigidas ao controlador;

- o protocolo utilizado é OpenFlow e o tipo da mensagem é *Packet-In*;
- a regra no SB11 que encaminhou a mensagem, tem colocada explicitamente a ação *output Controler*. Adicionalmente esta regra encontra-se na tabela 5 do equipamento SB11;
- a porta de entrada do pacote que gera a consulta é a número três (porta externa para a topologia);
- o endereço IP origem do pacote que gera a consulta é *10.1.1.2*;
- o endereço IP destino é *10.2.1.2*;
- o pacote que gerou a consulta ao controlador tem estabelecido o campo DSCP igual a 5.

5.2.4.4.2 *Packet-Out* .

A Fig. 5.11 mostra o conteúdo da mensagem *Packet-out* que envia o controlador ao equipamento de borda que gerou a consulta. Como podem ser observados em negrito na Fig. 5.11:

```
No.     Time       Source          Destination     Protocol Length Info
 459 16.558035   127.0.0.1       127.0.0.1       OFP 1.3  90     Packet out (CSM) - OFPT_PACKET_OUT

Frame 459: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 58697 (58697), Seq: 165, Ack: 533, Len: 24
Openflow Protocol
   Header
      Version: 0x04
      Type: Packet out (CSM) - OFPT_PACKET_OUT (13)
      Length: 24
      Transaction ID: 885077106
   Packet out
      Buffer ID: 0xffffffff
      Input port: 3
      Actions length: 0
      Padding
      Data: <MISSING>
```

Figura 5.11: Mensagem *Packet-Out*

- o protocolo utilizado é OpenFlow e o tipo da mensagem é Packet-out;
- o controlador indica que não tem que se efetuar ação sobre o pacote de entrada, pois ele já foi encaminhado.

5.2.4.4.3 *Flow-Mod* .

Finalmente, a Fig. 5.12 mostra o conteúdo da mensagem *Flow-mod* que o controlador envia a todos os equipamentos que intervêm no caminho ótimo, com os campos mais importantes para o trabalho ressaltados em negrito. Particularmente, é mostrado o pacote enviado ao SB11, mas as mensagens *Flow-mod* só se diferenciam pela porta de saída a configurar. Como podem ser observados em negrito na Fig. 5.12:

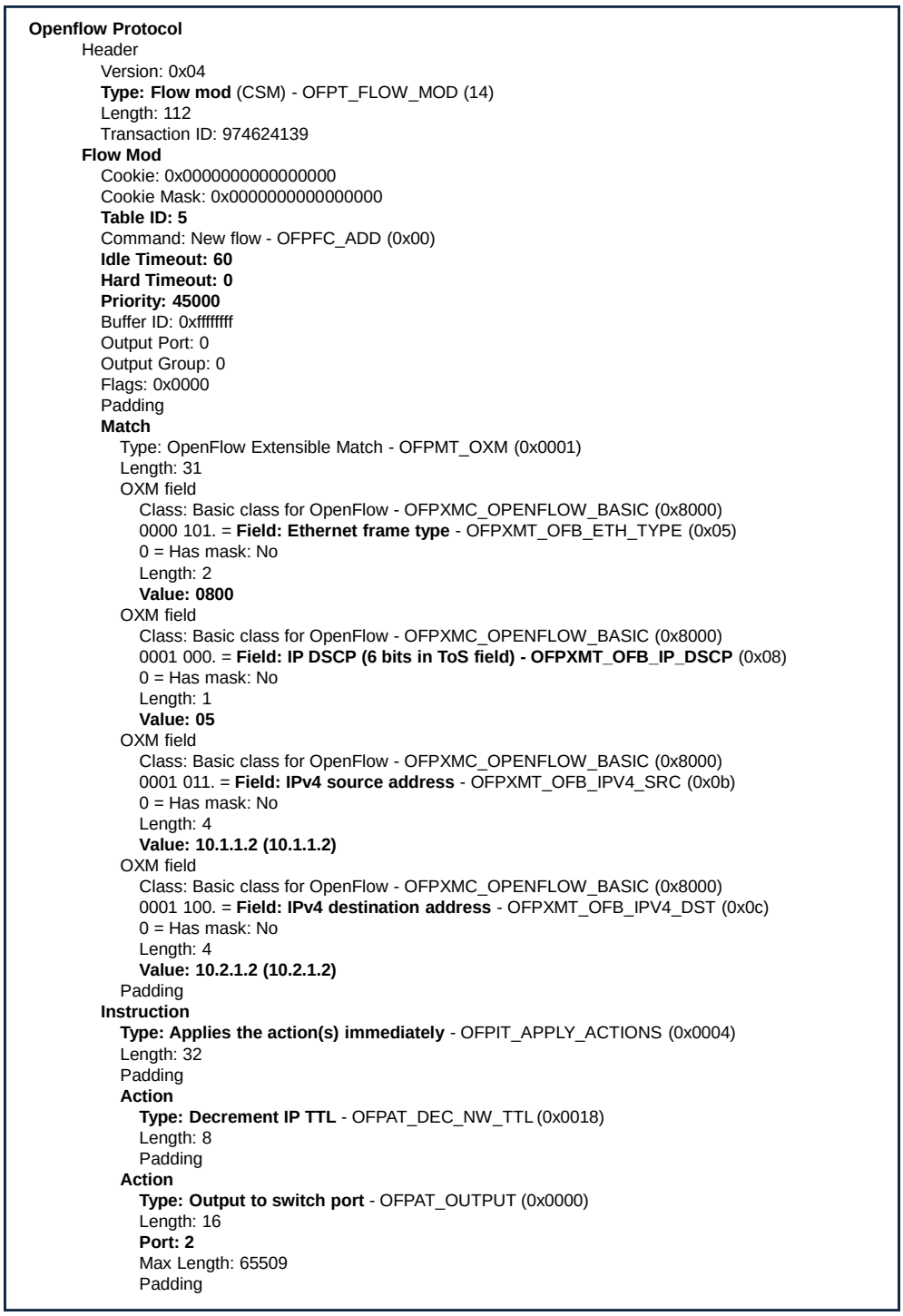

Figura 5.12: Mensagem *Flow-Mod*

- o protocolo utilizado é OpenFlow e o tipo da mensagem é flow-mod;
- é indicado que a regra específica QoS, a ser adicionada, tem que ser colocada na tabela 5;
- o *Idle-Timeout* que tem a regra criada é de *60* e o *Hard-Timeout* associado tem valor *0*;
- a prioridade da regra adicionada é *4.500* (fluxo específico prioritário). Desta forma, no

caso de chegar um pacote que requer QoS, os campos deste pacote sempre são comparados primeiramente com as regras específicas QoS configuradas e, no caso de não ser encontrada uma coincidência, continua-se a procura com as regras gerais de menor prioridade;

- Os campos de coincidência para a regra específica a adicionar no equipamento SB11 são:
  - *ethernet frame type = 0x0800* (o conteúdo do bloco tem que ser um pacote IP);
  - o campo DSCP tem que estar estabelecido em 5;
  - o endereço IP origem do pacote tem que ser *10.1.1.2*
  - o endereço IP destino do pacote tem que ser *10.2.1.2*;
- as instruções a serem executadas pelo pacote (que coincida com esta regra a adicionar) são do tipo imediata (*aply-action*) e são listadas a seguir:
  - decrementar o TTL (*Time to Live*);
  - enviar o pacote pela porta de saída número dois. Neste caso, indica que o equipamento SB11 tem que encaminhar o pacote ao SC12.

5.3 Comparativo das arquiteturas

5.3.1 Comparativo

Mediante a análise das seções anteriores e para facilitar a visualização da contribuição da arquitetura proposta, foi construída a Tabela 5.1, que mostra uma comparação entre as redes atuais, as redes implementadas com OpenFlow padrão, a arquitetura proposta com equipamentos híbridos (seção 3.2) e a arquitetura proposta com equipamentos OpenFlow (seção 3.1).

5.3.2 Observações

Para a arquitetura OpenFlow proposta, destaca-se:

- a robustez da rede ante a queda do controlador;
- a robustez da rede devido a atrasos na resposta do controlador;
- a diminuição da carga exigida ao controlador em razão de:
  - redução das mensagens que são enviadas ao controlador para o seu processamento (tráfego com QoS);
  - só o primeiro pacote de cada fluxo QoS realiza uma consulta ao controlador;
- a rápida resposta ao encaminhamento de pacotes de novos fluxos;
- a eliminação do plano de controle distribuído nos equipamentos.

Tabela 5.1: Comparativo das arquiteturas

	Rede atual	**OpenFlow padrão**	**Rede Híbrida proposta**	**Rede OpenFlow proposta**
Resposta ao encaminhamento	Imediata	Intermediária	Imediata	Imediata
Possibilidades de engenharia de tráfego	Boa	Muito Boa	Muito Boa	Muito Boa
Jitter para tráfego QoS	Meio	Baixo	Baixo	Baixo
Comportamento no caso de quedas do controlador	N/A	Ruim	Ótimo	Muito Bom
Comportamento ante atrasos na resposta do controlador	N/A	Ruim	Ótimo	Ótimo
Carga no controlador	N/A	Alta	Intermediária	Intermediária
Complexidade nos protocolos de roteamento	Muito alta	Intermediária	Muito alta	Intermediária
Carga por processamento nos equipamentos de rede	Alta	Baixa	Alta	Baixa

Para a arquitetura com equipamentos híbridos proposta, destaca-se:

- a robustez da rede devido a atrasos na resposta do controlador;
- a robustez da rede ante a queda do controlador;
- a robustez da rede ante mudanças topológicas durante uma queda do controlador;
- a diminuição da carga exigida ao controlador em razão de:
  - redução das mensagens enviadas ao controlador para seu processamento (tráfego com QoS);
  - só o primeiro pacote de cada fluxo QoS realiza uma consulta ao controlador;
- a rápida resposta ao encaminhamento de pacotes de novos fluxos.

Capítulo 6

Conclusões

Como foi mostrado ao longo do trabalho, e particularmente verificado mediante a realização de testes no protótipo, a arquitetura apresentada permite melhorar aspectos críticos de robustez que as implementações com OpenFlow padrão têm, mostrando como a arquitetura proposta mantém a rede em funcionamento ainda durante quedas do controlador, durante tempos de resposta elevados do controlador ou durante perdas das mensagens OpenFlow (devido a falhas nos enlaces ou equipamento intervenientes) no caminho fim a fim entre o controlador e o cliente OpenFlow.

Além disso, foi verificado mediante os testes realizados no protótipo, que a arquitetura proposta não adiciona tempos de espera na criação de novos fluxos. Como foi mencionado ao longo do trabalho, os primeiros pacotes de um fluxo QoS são encaminhados imediatamente pela tabela de encaminhamento interno e são paralelamente processados pelo controlador. Posteriormente, quando a tabela de QoS tem uma regra específica preenchida, o fluxo é reencaminhado sem tempos de espera nem perdas de pacotes.

Também foi reduzida a quantidade de mensagens que o controlador tem que processar. Diferentemente das implementações OpenFlow padrão (em que todos os novos fluxos são encaminhados ao controlador), na arquitetura proposta os fluxos sem QoS são encaminhados por tabelas construídas proativamente, que não geram consultas ao controlador (com regras ligadas a grupos de fluxos e não a fluxos individuais). Adicionalmente, foi proposto que só os equipamentos de borda enviem a solicitação de novo fluxo QoS ao controlador, reduzindo ainda mais as mensagens OpenFlow a serem processadas.

O presente trabalho permite solucionar problemas das redes atuais tais como a impossibilidade de construir caminhos para fluxos QoS individuais massivamente, as limitações na engenharia de tráfego que dificultam a diferenciação entre provedores de serviço, os complexos algoritmos de distribuição da informação que estão implementados em todos os equipamentos da rede, grandes dificuldades na unificação do plano de controle das diferentes redes que os ISPs operam e o fato de que a inteligência das redes se encontra oculta nos equipamentos, dificultando a inovação.

Como desvantagem, menciona-se o requisito de se utilizar um controlador centralizado, diminuindo, assim, a robustez da rede. Ainda que a arquitetura proposta com equipamentos OpenFlow seja muito robusta e de suporte a quedas do controlador, estas não podem ser por tempo indefi-

nido, pois se houver uma mudança topológica durante uma queda do controlador, este não poderá reconfigurar os fluxos na nova topologia. É importante notar que esta desvantagem não acontece na arquitetura híbrida proposta, mas neste caso os equipamentos têm que manter operacionais tanto o plano de controle distribuído quanto o plano de controle OpenFlow.

Como trabalhos futuros, propõe-se:

- Desenvolver algoritmos específicos para o cálculo do melhor caminho QoS, tendo em consideração as características dos diferentes tipos de tráfego.
- Implementar a lógica de encaminhamento interno, automática e adaptável às mudanças topológicas;
- Criar um controle OpenFlow para o gerenciamento da mobilidade IP, permitindo assim que usuários móveis mantenham seus fluxos ativos durante o processo de *handover*;
- Desenvolver o controle para o encaminhamento externo, tentando incrementar as possibilidade de controle;
- Propor técnicas, para a implantação de políticas de segurança, mediante a utilização de OpenFlow;
- Construir a lógica de controle para a criação de VPN, garantindo assim interconexão e isolamento entre os diferentes pontos que um cliente de este tipo requeira;
- Propor uma alternativa de migração da lógica de controle distribuída atual para às arquiteturas propostas e uma alternativa de interconexão entre os diferentes tipos de redes.

REFERÊNCIAS BIBLIOGRÁFICAS

[Agarwal et al., 2013] Agarwal, S., Kodialam, M., and Lakshman, T. (2013). Traffic engineering in software defined networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 2211–2219. IEEE.

[Bakshi, 2013] Bakshi, K. (2013). Considerations for software defined networking (SDN): Approaches and use cases. In *Aerospace Conference, 2013 IEEE*, pages 1–9. IEEE.

[Benton et al., 2013] Benton, K., Camp, L. J., and Small, C. (2013). OpenFlow vulnerability assessment. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 151–152. ACM.

[Bifulco et al., 2012] Bifulco, R., Brunner, M., Canonico, R., Hasselmeyer, P., and Mir, F. (2012). Scalability of a mobile cloud management system. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 17–22. ACM.

[Chial Sanchez, 2013] Chial Sanchez, B. (2013). Virtual aggregation in OpenFlow networks.

[Das, 2012] Das, S. (2012). *PAC.C.* PhD dissertation, Stanford University. `<http://www.openflow.org/wk/index.php/PACC_Thesis>`.

[Das et al., 2010] Das, S., Parulkar, G., McKeown, N., Singh, P., Getachew, D., and Ong, L. (2010). Packet and circuit network convergence with OpenFlow. In *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2010 Conference on (OFC/NFOEC)*, pages 1–3. IEEE.

[Das et al., 2011] Das, S., Yiakoumis, Y., Parulkar, G., McKeown, N., Singh, P., Getachew, D., and Desai, P. D. (2011). Application-aware aggregation and traffic engineering in a converged packet-circuit network. In *National Fiber Optic Engineers Conference*, page NThD3. Optical Society of America.

[Egilmez et al., 2011] Egilmez, H. E., Gorkemli, B., Tekalp, A. M., and Civanlar, S. (2011). Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 2241–2244. IEEE.

[Giorgetti et al., 2012] Giorgetti, A., Cugini, F., Paolucci, F., and Castoldi, P. (2012). OpenFlow and PCE architectures in wavelength switched optical networks. In *Optical Network Design and Modeling (ONDM), 2012 16th International Conference on*, pages 1–6. IEEE.

[Ivan Pepelnjak, 2012] Ivan Pepelnjak (2012). Hybrid OpenFlow, the brocade way. http://blog.ipspace.net/2012/06/hybrid-openflow-brocade-way.html.

[Jarschel et al., 2011] Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., and Tran-Gia, P. (2011). Modeling and performance evaluation of an OpenFlow architecture. In *Proceedings of the 23rd International Teletraffic Congress*, pages 1–7. ITCP.

[Lantz et al., 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM.

[Limoncelli, 2012] Limoncelli, T. A. (2012). Openflow: a radical new idea in networking. *Queue*, 10(6):40.

[López and Campelo, 2013] López, F. and Campelo, D. (2013). Rede SDN-OpenFlow para o caso de um ISP: Desafios e oportunidades. In *Simpósio Brasileiro de Telecomunicações*, pages 1–5. SBrT.

[Luo et al., 2012] Luo, T., Tan, H.-P., Quan, P. C., Law, Y. W., and Jin, J. (2012). Enhancing responsiveness and scalability for OpenFlow networks via control-message quenching. In *ICT Convergence (ICTC), 2012 International Conference on*, pages 348–353. IEEE.

[Matias et al., 2011] Matias, J., Jacob, E., Sanchez, D., and Demchenko, Y. (2011). An OpenFlow based network virtualization framework for the cloud. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 672–678. IEEE.

[McKeown et al., 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

[Mininet v.2.1, 2012] Mininet v.2.1 (2012). Open Source Project. https://mininet.org.

[Namal et al., 2013] Namal, S., Ahmad, I., Gurtov, A., and Ylianttila, M. (2013). Enabling secure mobility with OpenFlow. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, pages 1–5. IEEE.

[Open Networking Foundation, 2013] Open Networking Foundation (2013). Outcomes of the hybrid working group. https://www.opennetworking.org/images/stories/downloads/working-groups/summary-hybrid.pdf.

[Open vSwitch v.2.0, 2012] Open vSwitch v.2.0 (2012). Open Source Project. http://openvswitch.org/.

[OpenFlow v.1.3 , 2012] OpenFlow v.1.3 (2012). Open Networking Foundation. OpenFlow switch specification version.1.3.0. https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf.

[Pfaff et al., 2009] Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., and Shenker, S. (2009). Extending networking into the virtualization layer. In *Hot Topics in Networks*. HotNets.

[Ryu v.3.3, 2012] Ryu v.3.3 (2012). Open Source Project. http://osrg.github.io/ryu/.

[SDN Definition,] SDN Definition. Open Networking Foundation. Software-Defined Networking (SDN) Definition. https://www.opennetworking.org/sdn-resources/sdn-definition.

[SDN Product Directory,] SDN Product Directory. Open Networking Foundation. https://www.opennetworking.org/sdn-resources/onf-products-listing.

[Shirazipour et al., 2012] Shirazipour, M., John, W., Kempf, J., Green, H., and Tatipamula, M. (2012). Realizing packet-optical integration with SDN and OpenFlow 1.1 extensions. In *Communications (ICC), 2012 IEEE International Conference on*, pages 6633–6637. IEEE.

[Wamser et al., 2011] Wamser, F., Pries, R., Staehle, D., Heck, K., and Tran-Gia, P. (2011). Traffic characterization of a residential wireless internet access. *Telecommunication Systems*, 48(1-2):5–17.

[William Stallings, 2000] William Stallings (2000). Queuing analysis. http://www.telecom.otago.ac.nz/tele302/ref/Stallings_QT.pdf.