UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UMA SOLUÇÃO DE ROTEAMENTO COMO SERVIÇO BASEADA EM REDES DEFINIDAS POR SOFTWARE

Carlos Nilton Araújo Corrêa

Orientadores:
Sidney Cunha de Lucena
Christian Rodolfo Esteve Rothenberg

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2012

UMA SOLUÇÃO DE ROTEAMENTO COMO SERVIÇO BASEADA EM REDES DEFINIDAS POR SOFTWARE

Carlos Nilton Araújo Corrêa

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Sidney Cunha de Lucena, D.Sc. – UNIRIO

Christian Rodolfo Esteve Rothemberg, Ph.D. – CPQD

Carlos Alberto Vieira Campos, D.Sc. – UNIRIO

Michael Anthony Stanton, Ph.D. – UFF

RIO DE JANEIRO, RJ – BRASIL

FEVEREIRO DE 2012

Corrêa, Carlos Nilton Araújo.
C824 Uma solução de roteamento como serviço baseada em redes definidas por software / Carlos Nilton Araújo Corrêa, 2012.
xv, 117f. ; 30 cm

Orientador: Sidney Cunha de Lucena.
Coorientador: Christian Rodolfo Esteve Rothenberg.
Dissertação (Mestrado em Informática) – Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2012.

1. Arquitetura de redes de computador. 2. Roteamento (Administração de redes de computadores. 3. Openflow. 4. Redes definidas por software. I. Lucena, Sidney Cunha de. II. Rothenberg, Christian Rodolfo Esteve. III. Universidade Federal do Estado do Rio de Janeiro . Centro de Ciências Exatas e Tecnologia. Curso de Mestrado em Informática. IV. Título.

CDD – 004.65

À meus amados pais e esposa,
por sua dedicação e paciência infinitos.

Agradecimentos

Agradeço em primeiro lugar ao Altíssimo, que me concedeu a saúde e a disposição necessárias para alcançar esta meta.

Em seguida a meus pais, Sonilton e Zélia, por haverem feito de seus filhos seus projetos de vida. Meus acertos são todos seus: sua dedicação e amor me inspiram em tudo o que faço, e espero estar sempre à altura de seu exemplo de grandeza de caráter e espírito.

À minha esposa, motivadora, parceira e melhor metade: Nanda – por estar sempre ao meu lado, inclusive durante minha matrícula na Unirio. Por seu amor, carinho, dedicação e paciência para com os compromissos do mestrado. Eu jamais poderia fazê-lo sem seu apoio.

Ao Prof. Sidney Lucena, meu orientador e amigo, por haver me concedido o privilégio de ser seu aluno e por seus ensinamentos. E pela parceria de madrugadas a fio, em discussões via Skype. Você excedeu em muito o papel de orientador.

Ao Dr. Christian Esteve, meu co-orientador e amigo, pela chance de participar do projeto RouteFlow, por sua infinita boa-vontade e talento. Você é um grande líder.

A Ralf Batista, meu gerente e amigo, pelo apoio incondicional à realização deste curso. E por refletir este apoio, à Unimed Federação RJ – a melhor empresa para se trabalhar no país – e à sua Diretoria Executiva.

Aos professores do DIA/PPGI, com quem tive a satisfação de aprender, e que invariavelmente se colocam à disposição de todos nós, alunos. E aos profissionais de apoio da Unirio, que com gentileza ímpar concorrem para o sucesso da instituição.

Aos amigos: de estudos, que fizeram esta jornada muito divertida, em especial Pablo

e Daniel; e da vida, em especial ao Prof. Alexandre Luiz, pela generosidade e companheirismo.

"Nesta vida,
pode-se aprender três coisas de uma criança:
estar sempre alegre,
nunca ficar inativo
e chorar com força por tudo o que se quer."
Paulo Leminski

Resumo

Este trabalho propõe que a operação de um Sistema Autônomo (AS) da Internet seja orientada por uma descrição de alto nível de sua política de negócio, em detrimento da configuração de parâmetros de protocolo. Para concretizar esta visão, apresenta-se o conceito de uma plataforma de Roteamento como Serviço. Suas características são formalmente estabelecidas e, quando aplicável, experimentalmente validadas. Tal plataforma foi construída a partir da arquitetura RouteFlow, projeto de código aberto desenvolvido pelo CPqD que permite a virtualização de operações de roteamento IP em redes OpenFlow. Sobre esta plataforma, um serviço de roteamento foi construído como prova de conceito. Sua avaliação se dá num estudo de caso onde, diferentemente do modelo tradicional, um único processo rodando o protocolo BGP é usado para definir, de forma centralizada, o roteamento externo de um AS, sem ainda precisar de um protocolo IGP para o roteamento interno. Os resultados obtidos sugerem que a plataforma serve como um instrumento para a realização da política de negócio de um AS, sob uma perspectiva de baixa sobrecarga administrativa.

Abstract

This work proposes that the operation of an Internet Autonomous System (AS) should be oriented by a high-level description of its business policy, rather than protocols' parameters configuration. To further materialize this view, we introduce the concept of a Routing as a Service platform. Its characteristics are formally established and experimentally validated, as appliable. The platform was based on the RouteFlow architecture, an open source project from CPqD that allows for the virtualization of IP routing operations in OpenFlow networks. Over this platform, a novel routing service was built as a proof-of-concept. This service was evaluated in a case study where, differently from traditional operation, a single process executing BGP is used to define, in a centralized fashion, the external routing of an AS, and yet without the need to run an IGP for internal routing. The results obtained suggest that the platform serves as a tool for the execution of an AS' business policy, under the perspective of low administrative overhead.

Sumário

Lista de Figuras

Lista de Siglas

API	Application Programming Interface
ARP	Address Resolution Protocol
AS	Autonomous System
BGP	Border Gateway Protocol
DRAGON	Dynamic Resource Allocation via GMPLS Optical Networks
EBNF	Extended Backus–Naur Form
FIB	Forwarding Information Base
GMPLS	Generalized Multiprotocol Label Switching
IaaS	Infrastructure as a Service
iBGP	Internal Border Gateway Protocol
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISP	Internet Service Provider
LDP	Label Distribution Protocol
LPM	Longest Prefix Matching
LXC	Linux Containers
MAC	Media Access Control

MMV	Monitor de Máquinas Virtuais
MV	Máquina Virtual
OSPF	Open Shortest Path First
OVS	Open vSwitch
PaaS	Platform as a Service
RAM	Random Access Memory
RIP	Routing Information Protocol
RFC	Request for Comments
RCP	Routing Control Platform
RR	Route-Reflector
RSVP	Resource Reservation Protocol
SDN	Software-Defined Networking
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
VR	Virtualização de Redes

1 Introdução

1.1 Considerações iniciais

A virtualização, de uma perspectiva de negócios, cede aos provedores de serviços e conteúdo baseados na Internet a chance de concentrar-se na operação de seu *serviço*, ao invés de na operação de seus *componentes*. Esta oportunidade é potencializada pela computação em nuvem, que encapsula o mapeamento entre os contextos virtual e físico. Assim, a infraestrutura subjacente comporta-se menos como um conjunto de reservas individuais de capacidade, aproximando-se mais de uma reserva única de insumos computacionais, consumida conforme as instâncias de sistemas definidas pelo operador. Tais conceitos compõem os fundamentos de diversas contribuições à execução de redes virtuais e, mais especificamente, à operação de roteadores, com vistas à racionalização do uso de recursos (EGI et al., 2008) (SARRAR et al., 2010), aumento de disponibilidade (WANG et al., 2008) e centralização do plano de controle (BOLLA et al., 2009) (NASCIMENTO et al., 2010) (BOZAKOV, 2010).

Em Keller e Rexford (2010), no entanto, observa-se que apesar das inovações baseadas em plataformas virtuais oferecerem uma visão abstrata dos recursos disponíveis, ainda mantêm o *modus operandi* de redes tradicionais. Cabe ao operador estabelecer a conectividade entre seus roteadores virtuais, configurá-los individualmente e gerenciar aspectos de sua capacidade e dos canais de comunicação correspondentes. Relativamente aos desafios clássicos do encaminhamento de tráfego na Internet, isto significa que apenas parte do esforço de gerenciamento dos equipamentos de um provedor de telecomunicações possa ser reduzido ou eliminado (ex: unificação e automação de processos, auto-configuração)

pelo uso de tecnologias de virtualização. E mesmo que todo o plano de controle de sua rede venha a ser consolidado e centralizado, é necessário estabelecer comunicação entre diferentes processos de roteamento. Tais processos precisam estabelecer adjacências, hierarquias e trocar mensagens de controle, tal como se estivessem operando sobre dispositivos distintos. Neste contexto, a apresentação do plano de controle de roteamento em uma abstração de mais alto nível ainda é um problema em aberto.

Mais ainda, as decisões de cada elemento roteador são tomadas a partir de suas visões particulares da rede, e não a partir de uma política global de encaminhamento, aplicada uniformemente a todos os dispositivos. Assim, traduzir os requisitos de negócio de um provedor de telecomunicações (a operação de seu *serviço*) em atividades de configuração de todos estes componentes é um desafio da área de roteamento Internet (ZHANG-SHEN; WANG; REXFORD, 2008). E mesmo em outros tipos de organização, como *datacenters* e provedores de serviços de nuvem, as necessidades de encaminhamento entre clientes podem ser diferentes ou conflitantes - o que também pode constituir um problema para a operação de seu roteamento. O presente trabalho aborda ambos os problemas através da transição de uma arquitetura virtual de roteamento IP para uma *plataforma de roteamento como serviço*. Assim, o sentido de "serviço" aproxima-se daquele empregado no desenho de arquiteturas empresariais digitais (PAPAZOGLOU et al., 2007): pretende-se apoiar elementos que desempenhem partes independentes de um processo de negócio, opacos entre si, mas que por suas naturezas complementares alcançam um resultado conjunto (o "roteamento" de tráfego). Finalmente, para que um operador possa implementar serviços arbitrários e descrever de forma única o comportamento esperado de uma arquitetura virtual de roteamento IP, espera-se prover uma "plataforma", como um conjunto de interfaces e abstrações de alto nível compartilhadas pelos serviços.

1.2 Motivação

Um Sistema Autônomo da Internet (ou simplesmente "AS", do inglês "Autonomous System"), geralmente representa um domínio administrativo que hospeda sistemas finais,

ou de algum outro modo fornece serviços de comunicação, em uma escala suficiente para que se justifique anunciar, em nível global, a disponibilidade de acesso às suas próprias redes. Tal disponibilidade é "anunciada" por meio do protocolo BGP, mecanismo este que permite a um AS informar aos outros a quais redes sua infraestrutura está conectada. Portanto, para que seja possível a troca de dados entre sistemas de todo o mundo, os diversos ASs que compõem a Internet compartilham entre si as redes com as quais têm contato. Assim, quando um de seus roteadores precisa enviar dados para uma rede distante que esteja disponível, pesquisa em sua base de informações qual dos ASs vizinhos conhece o menor caminho rumo ao destino e transmite o respectivo tráfego através dele (KUROSE; ROSS, 2009).

Tal generalização, apesar de próxima da realidade, esconde o fato de que nem sempre é tão simples tomar decisões de roteamento. Principalmente porque se ASs geralmente se encontram em organizações que fornecem serviços ligados à comunicação Internet, então estes podem ser tratados como ISPs (do inglês "Internet Service Providers", ou "Provedores de Serviço Internet") (NORTON, 2000). Sendo assim, seu modelo de negócios está fortemente ligado tanto ao volume de tráfego trocado por seus clientes, quanto ao tráfego que enviam e recebem por meio de outros provedores.

As trocas de tráfego de um ISP podem ser então classificadas de três maneiras: aquelas estabelecidas com seus clientes; aquelas estabelecidas com ISPs maiores, que têm redes com muitos clientes, ou conexões com muitas redes, de forma que podem servir eles próprios como um "caminho rápido" para muitos pontos da Internet; e as que são formadas com parceiros ("peers", no inglês), provedores que se não tem a escala ou a infraestrutura para servir como melhor caminho para muitos destinos, podem abrigar serviços e sistemas que componham um grupo de especial interesse para seus clientes (NORTON, 2000).

Ainda segundo Norton (2000), da mesma maneira que um cliente final contrata seu ISP para enviar e receber informações da Internet, é possível que seja firmado um acordo entre *peers* para que eles utilizem o serviço de transporte de dados um do outro. Indo

além do interesse mútuo no acesso às respectivas redes (que é mais comum), acordos entre provedores podem envolver um pagamento em função dos volumes de dados trafegado. Nestes casos, os acordos geralmente se baseiam na diferença dos volumes de tráfego consumidos por cada um deles em um determinado período. Por exemplo, se grande quantidade de informações fluiu ao longo de um mês do ISP "A" em direção ao ISP "B", mas um volume pequeno viajou em sentido contrário, então o ISP "B" precisa pagar por esta diferença. Portanto, é desejável que a lógica de encaminhamento de pacotes entre redes reflita não só o conhecimento do "menor caminho" até o destino, mas também a política de negócio de um AS, que é constituída por seus objetivos ao firmar cada relação de troca de tráfego (ZHANG-SHEN; WANG; REXFORD, 2008). Assim, para o exemplo apresentado, talvez fosse desejável contar com um instrumento que permitisse que o ISP "B" utilizasse rotas através do ISP "A" apenas até o momento em que esta opção não implicasse em uma cobrança futura. Então, caso tal a relação de troca estivesse desequilibrada, outras rotas disponíveis poderiam ser selecionadas.

O protocolo BGP oferece instrumentos que permitem influenciar sua lógica de encaminhamento de maneira a obter-se um resultado próximo de uma política de negócio existente. Por diferentes motivos, porém, as decisões tomadas por um roteador são remetidas a outros dispositivos do mesmo tipo existentes na rede do AS e influenciam suas próprias decisões. Ao mesmo tempo, equipamentos diferentes podem haver sido designados para cumprir objetivos de negócio distintos, de modo que a influência mútua de decisões (que são tomadas independentemente) pode produzir resultados imprevisíveis e que violem a política de negócio pretendida.

Tal situação é um reflexo da operação do roteamento Internet basear-se na premissa de que ASs são elementos atômicos, mas de fato o comportamento de um domínio administrativo resultar da execução de um sistema distribuído (ZHANG-SHEN; WANG; REXFORD, 2008).

As arquiteturas virtuais de roteamento IP oferecem um cenário de execução centralizada deste sistema distribuído, em um plano de controle virtual na memória de um sistema

controlador. Porém, não oferecem um mecanismo que possa se privilegiar deste cenário, se limitando a reconstituir, a partir de novos elementos, os mesmos paradigmas da operação tradicional de protocolos de roteamento.

Através de intervenções no modelo de operação de uma arquitetura virtual de roteamento IP, este trabalho pretende obter meios e métodos para que o processo de transcrição de uma política de negócio em instruções de encaminhamento possa ser mais fiel a seus objetivos constituintes.

1.3 Objetivos

O objetivo geral desta dissertação é estabelecer uma plataforma de roteamento como serviço, que seja capaz de extrair decisões de roteamento de um plano de controle virtual e submetê-las a lógicas de encaminhamento arbitrárias, que realizem completamente ou em parte a política de negócio de um sistema autônomo da Internet. Cada possível unidade lógica aplicada à plataforma é aqui denominada um *serviço* de roteamento.

Esperamos responder à hipótese de que uma arquitetura virtual de roteamento IP como a RouteFlow (NASCIMENTO et al., 2011a) pode ser transformada em uma plataforma desse tipo. Esta transformação ocorreria pela incorporação de elementos de especificação dos serviços que traduzem a política de negócio do AS e da infraestrutura subjacente à sua operação. Tais elementos compõem parte dos objetivos específicos do trabalho, a saber:

- A definição de um modelo informacional que permita descrever o comportamento esperado para o roteamento de um AS, nos termos de um ou mais serviços de encaminhamento que o componham. Portanto, o roteamento que se deseja realizar é tomado como um sinônimo da política de negócio do provedor.
- A definição de um modelo informacional que permita descrever a conectividade física de um AS.
- Uma gramática de configuração que possibilite codificar em uma estrutura de alto

nível as informações presentes no modelo.

- Determinar que ferramentas de construção de redes virtuais melhor se aplicam ao estabelecimento do plano de controle virtual de uma plataforma de roteamento como serviço.

- Desenvolver um serviço de roteamento que possa ser executado sobre a infraestrutura da plataforma constituída e que possa ser aplicado à solução de problemas de roteamento do mundo real.

1.4 Metodologia de pesquisa

Com relação à sua natureza, o trabalho aqui desenvolvido se enquadra na classificação de pesquisa aplicada ou tecnológica, que tem como objetivo inovar um produto ou processo, frente a uma demanda ou necessidade preestabelecida (JUNG, 2003). Neste caso, espera-se inovar as arquiteturas virtuais de roteamento IP, transformando-as em plataformas de roteamento como serviço e constituindo uma nova classe de ferramentas para a gerência do encaminhamento de tráfego na Internet.

Ainda conforme Jung (2003), esta pesquisa pode ser classificada como exploratória em relação a seus objetivos, pois busca estabelecer novas práticas, com a construção de serviços de roteamento que traduzam requisitos de encaminhamento. Neste sentido, ela também oferece o próprio instrumento que pode servir de base a futuras inovações – uma plataforma de roteamento como serviço funcional. Quanto aos procedimentos empregados, eles podem ser classificados em dois tipos. O primeiro é o tipo experimental, em que se enquadram as avaliações de desempenho de ferramentas para planos de controle virtuais realizadas no Capítulo 4. Para realização destas avaliações, inicialmente um grupo de ferramentas do estado da arte é selecionado, com base em uma análise qualitativa. Em seguida, os elementos deste grupo são submetidos a testes de natureza quantitativa que buscam verificar quais deles podem apresentar o melhor desempenho quando aplicados a uma plataforma de roteamento virtual. A segunda categoria constitui-se do estudo de

caso apresentado no Capítulo 6, que avalia o serviço de roteamento desenvolvido para a plataforma em construção. O estudo baseia-se em um cenário hipotético, mas que se acredita que possa refletir as características de redes de produção do mundo real. Portanto, espera-se que seja possível generalizar, em algum grau, seus resultados.

1.5 Estrutura do texto da dissertação

Na Figura 1 é apresentada a distribuição do trabalho de pesquisa ao longo dos capítulos da dissertação. As linhas tracejadas indicam uma relação entre cada tópico e o respectivo capítulo.

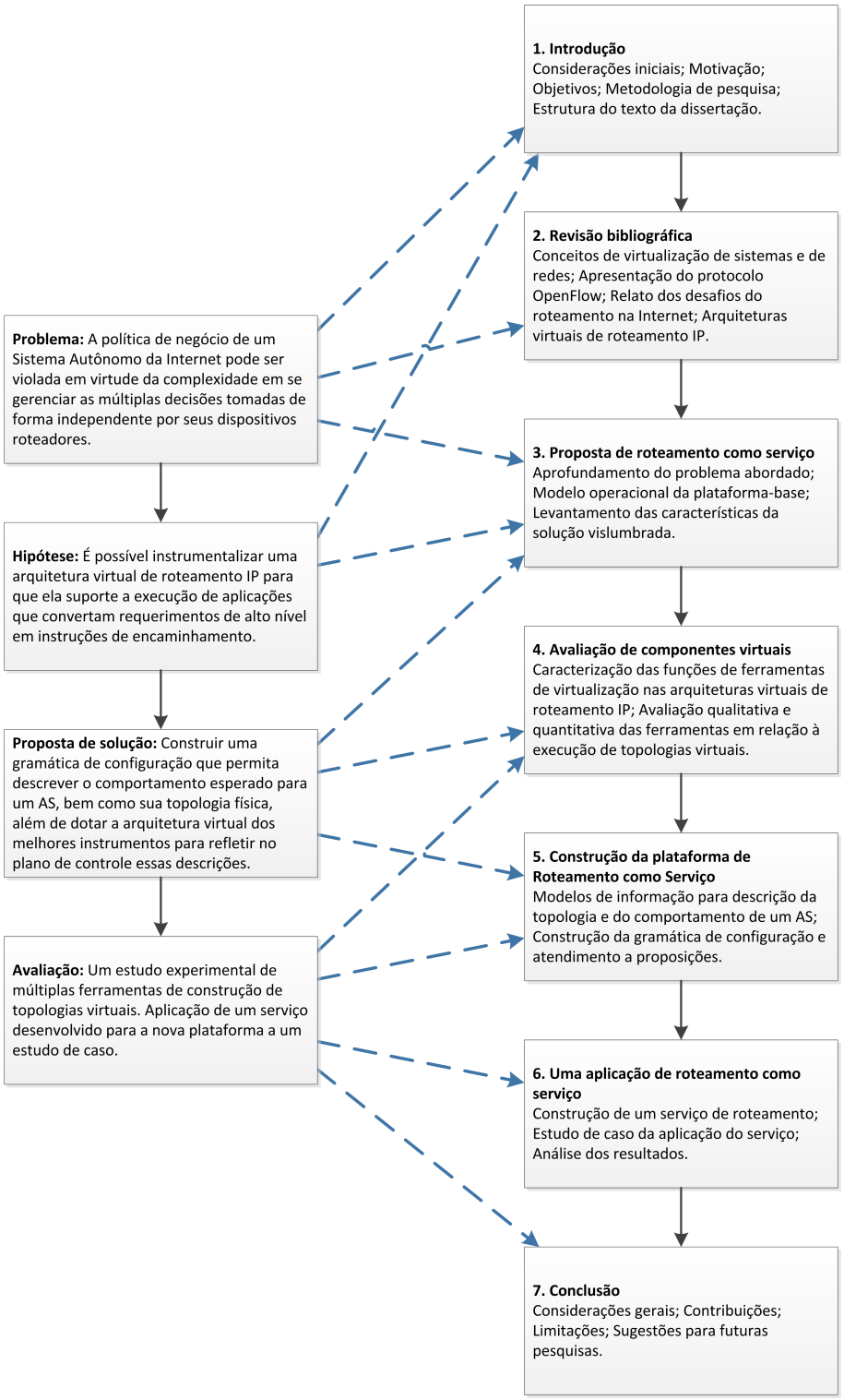


Figura 1: Organização do texto neste trabalho de pesquisa.

2 Revisão bibliográfica

Neste capítulo serão apresentados os conceitos considerados instrumentais para a realização e o entendimento deste trabalho.

Os conhecimentos sobre encaminhamento e troca de tráfego comporão um panorama dos desafios da operação de um Sistema Autônomo da Internet, aos quais os resultados desta pesquisa visam atender.

Os fundamentos de virtualização de sistemas e redes serão necessários para contextualizar a área em que se situa a pesquisa, além de embasar a seleção de ferramentas descrita no Capítulo 4. O elemento através do qual o conceito de virtualização de redes é materializado nesta pequisa é a arquitetura OpenFlow. Assim, a Seção 2.3 se detém em sua apresentação.

Os resultados produzidos ao longo deste trabalho visam compartilhar e expandir o espaço de soluções das arquiteturas virtuais de roteamento IP. Assim, estas arquiteturas também são objeto de estudo ao longo da Seção 2.4.

Paralelamente à apresentação de conceitos, os principais trabalhos relacionados a cada tema são referenciados e discutidos.

2.1 O BGP e os desafios do roteamento Internet

Um adágio popular da Internet reputa ao protocolo BGP a função de "colar" as múltiplas redes que a compõem. Pode-se dizer que isto é em grande parte verdade, de acordo com uma visão apresentada em Butler et al. (2004). O protocolo BGP é o ele-

mento responsável por divulgar a *alcançabilidade* entre diferentes redes, habilitando-as a comunicarem-se entre si e a fornecerem trânsito umas às outras. O foco no aspecto inter-redes é particularmente importante neste caso: enquanto grande parte dos protocolos de roteamento preocupa-se com a alcançabilidade entre nós individuais, a operação BGP assume que cada rede possa ser tratada como uma unidade atômica, ainda que isto nem sempre seja verdadeiro (ZHANG-SHEN; WANG; REXFORD, 2008).

Tal premissa implica em um alto custo de gerenciamento por parte dos operadores de sistemas autônomos. Uma vez que cada roteador sob seu domínio administrativo executa uma instância BGP distinta, geralmente em condições diferentes de conectividade e alcançabilidade, não é possível garantir que suas decisões individuais de encaminhamento serão sempre previsíveis. A despeito disto, os provedores de telecomunicações precisam que estas decisões estejam de acordo com seus contratos de transporte de dados, ou seja, suas políticas de negócio.

Até o momento, a solução mais comum para este impasse envolve configurar cada um dos roteadores em um AS de forma a influenciar o processamento da alcançabilidade BGP, na expectativa de que isto produza um comportamento consistente com a política de negócio vigente. Porém, isto nem sempre é possível, conforme problemas levantados na literatura e relatados a seguir.

A forma trivial de executar o protocolo BGP, em um sistema autônomo que conta com múltiplos roteadores, exige o estabelecimento de sessões iBGP entre todos eles (a organização *full-meshed*). Assim, todos os dispositivos podem trocar diretamente informações de alcançabilidade recebidas das redes de terceiros com as quais se conectam. Enquanto envolve mínima interferência sobre as informações propagadas (e assim, pode facilitar a conciliação das configurações de diferentes roteadores), o trabalho de Park et al. (2010) aponta que em uma rede de N roteadores esta saída demanda um total de $\frac{N*(N-1)}{2}$ sessões BGP intra-AS, o que pode ser inviável tanto sob o aspecto do esforço de configuração quanto do consumo de recursos em cada equipamento. Adicionalmente, tal abordagem também obriga o operador a reconfigurar todo o ambiente para refletir a mudança do

endereço de um único dispositivo.

Confederações BGP permitem constituir vários pequenos sistemas autônomos ("sub-ASs") dentro de um mesmo AS, que comunicam-se de forma análoga aos roteadores em domínios administrativos distintos. Esta técnica apresenta-se como uma saída para o *full-meshing* de sessões iBGP, pois ao invés de se estabelecerem sessões entre todos os equipamentos de um mesmo AS, basta que haja um *full-mesh* entre todos os dispositivos pertencentes ao mesmo sub-AS. Tal abordagem, porém, conta com uma limitação importante: o recurso de confederações precisa ser sempre utilizado de forma global em um AS, não sendo possível contar com equipamentos que façam parte do mesmo sistema autônomo, mas não sejam parte de algum de seus sub-ASs. É possível que esta seja uma das causas da pequena utilização de confederações relatada em Park et al. (2010).

Uma ferramenta amplamente adotada é o recurso de designar a um roteador (ou equivalente) a função de *route-reflector*. Para tanto, todos os roteadores do AS devem estabelecer uma sessão iBGP com o *route-reflector*, o que os dispensa de fazê-lo para os demais. Através das sessões iBGP estabelecidas consigo, o *route-reflector* recebe, consolida e divulga informações de alcançabilidade que, espera-se, tem como efeito uma operação uniforme em todo o AS. O trabalho em Park et al. (2010) porém, faz referência a problemas identificados em cenários com *route-reflectors* que afetam a resiliência da topologia da rede em relação a falhas (XIAO; WANG; NAHRSTEDT, 2003) (XIAO; WANG; NAHRSTED, 2003), introduzem atrasos de convergência (CAESAR et al., 2005) e *loops* de roteamento (DUBE, 1999) (DUBE; SCUDDER, 1999) (GRIFFIN; WILFONG, 2002) (SCUDDER; DUBE, 1999), dentre outros.

Os instrumentos e dificuldades elencados referem-se apenas ao controle da execução distribuída do protocolo BGP. No limite, buscam torná-la determinística. Mas, há ainda outro fator extrínseco, que pode influenciar o processo decisório de roteadores e levar a uma violação da política de roteamento desejada.

O universo de informações produzidas, consumidas e disseminadas pela operação

BGP compreende apenas a alcançabilidade e o número de saltos entre redes. Isto significa que, por definição, ao tomar o conhecimento de um prefixo P_1 alcançável exclusivamente através do roteador remoto RR_1, um roteador local RL_1 informa aos demais equipamentos de seu AS que o endereço IP de RR_1 é a rota escolhida para P_1. Caso os roteadores daquele AS não contem com outra opção para alcançar P_1, tentarão incorporar esta informação às suas tabelas de encaminhamento. A adoção de uma rota, porém, depende de uma condição: que cada roteador $RL_{2,3...n}$ realizando seu processamento saiba como alcançar RR_1 ou ela não será usável. Portanto, a execução de um protocolo de roteamento secundário do tipo IGP, capaz de disseminar informações de alcançabilidade entre nós faz-se necessária.

Desta forma, a execução de IGPs tem grande influência sobre as decisões de encaminhamento de cada dispositivo, que dependem de informações de alcançabilidade entre nós para utilizar uma rota aprendida via BGP. Isto implica na necessidade de se manter mais um artefato de configuração ativo e consistente com a política de negócio vigente, além de elevar a demanda por recursos de processamento em cada roteador.

2.2 Virtualização

A virtualização de sistemas é uma técnica que permite que múltiplos processos em execução compartilhem o mesmo *hardware*, enquanto lhes oferece a ilusão de executarem sobre uma infraestrutura dedicada (BHATIA et al., 2008). Entretanto, tem-se um computador ("anfitrião") cujos recursos são partilhados entre eles.

Dentre as vantagens oferecidas pela aplicação desta tecnologia, está o isolamento. Se uma aplicação possui erros, é possível que executá-la em conjunto com outro processo traga instabilidade ao conjunto. Carregá-los em máquinas virtuais distintas, neste caso, garante que o primeiro programa não causará problemas ao segundo. A virtualização também pode representar um fator de uso eficiente da capacidade computacional disponível (EGI et al., 2007).

Um conceito análogo surgiu no contexto das redes de computadores, dando origem à virtualização de redes (CHOWDHURY; BOUTABA, 2010). A virtualização de redes permite particionar a capacidade dos componentes de uma rede física. Torna-se possível então estabelecer múltiplas infraestruturas lógicas distintas sobre os mesmos componentes.

A seguir, são apresentados em detalhes cada um dos conceitos, bem como os trabalhos realizados em ambos os segmentos e que se relacionam a este.

2.2.1 Virtualização de sistemas

Pode-se realizar a virtualização de sistemas de duas formas: a *virtualização completa*, em que cada convidado executa seu sistema operacional, e a virtualização baseada em *containers*, em que o sistema operacional do anfitrião distribui e isola os recursos disponíveis entre os sistemas convidados (BHATIA et al., 2008).

Segundo Bhatia et al. (2008), na virtualização completa o anfitrião executa um Monitor de Máquinas Virtuais (MMV, ou *hipervisor*). É responsabilidade do MMV prover uma abstração de *hardware* (chamada *máquina virtual*) para que seja possível executar o sistema operacional convidado. Ele também deve mapear cada requisição dos convidados a seus respectivos dispositivos virtuais para o elemento físico correspondente. Existe uma variação desta técnica chamada *paravirtualização* (BHATIA et al., 2008), que consiste em otimizar a emulação de *hardware* provida pelo MMV com vistas a um ganho de desempenho. Nesta abordagem, porém, os sistemas operacionais convidados precisam ser modificados - o que não acontece na virtualização completa.

Em um virtualizador de *containers*, uma imagem de sistema operacional virtualizada é compartilhada entre os nós convidados. Isto pode ser mais eficiente quando é preciso apenas executar de forma isolada diversas cópias do mesmo *software*. Ainda assim os nós virtuais podem ser individualmente gerenciados, como na virtualização completa.

2.2.2 Virtualização de redes e conectividade virtual

Segundo Chowdhury e Boutaba (2010), existem quatro abordagens para a implementação de VR: as Redes Locais Virtuais (VLANs, na sigla em inglês), as Redes Virtuais Privadas (VPNs, também na sigla da língua inglesa), as redes ativas e as redes de sobreposição. Em Sherwood et al. (2010) apresenta-se uma quinta, a partir do conceito de redes definidas por *software* (do inglês *software-defined networking*, ou simplemente SDN).

Para constituir uma rede definida por *software* é necessário dividir os elementos de uma rede em duas categorias. Há aqueles dedicados a encaminhar o tráfego entre os sistemas finais conectados, os quais designamos elementos do *plano de dados*. Complementarmente, há elementos que atuam no *plano de controle* (controladores), sendo responsáveis por decidir de que forma o plano de dados deverá realizar seu trabalho. Acrescenta-se então ao plano de controle o suporte a uma interface programática, o que permite que aplicações de alto nível sejam criadas para instrumentalizá-lo.

Uma série de arquiteturas e protocolos implementam o conceito de rede definida por *software*, como os protocolos ForCES (KHOSRAVI; ANDERSON, 2003) e GSMP (DORIA et al., 2002).

Outra arquitetura de rede programável, baseada no protocolo OpenFlow, foi apresentada em McKeown et al. (2008) com o objetivo de permitir a pesquisadores a execução de múltiplos experimentos de redes e protocolos sobre uma única infraestrutura.

Independentemente do uso de VR, existe o desafio de se estabelecer comunicação entre MVs que fazem parte de uma mesma rede, especialmente quando estas executam sobre o mesmo anfitrião. Nestes cenários surge a necessidade de um mecanismo de *conectividade virtual*. Tal conectividade é geralmente estabelecida por dispositivos virtuais baseados em *software*, como o exemplo de Pfaff et al. (2009). Estes dispositivos são essenciais para as plataformas de roteamento estudadas na Seção 2.4.

2.3 A abordagem SDN OpenFlow

Conforme relatado na seção anterior, existem diferentes arquiteturas que implementam o conceito de rede definida por *software*, tratando-se a OpenFlow de apenas uma das opções disponíveis. Uma vez, porém, que esta foi a abordagem em que se baseou a plataforma construída ao longo desta pesquisa, seu funcionamento é apresentado em detalhes a seguir.

A concepção original OpenFlow introduzida em McKeown et al. (2008), envolve um protocolo de comunicação que implementa o conceito de redes definidas por *software*. Neste sentido, não há dúvida de que "OpenFlow" é um *protocolo*. Mas, no contexto de uma rede baseada em sua operação, equipamentos de conectividade que desempenhavam funções há muito estabelecidas passam a tê-las eliminadas ou redirecionadas. Para refletir a mudança de natureza destes elementos, uma nova terminologia é adotada, inclusive. Assim, por estabelecer um modelo operacional específico, "OpenFlow" também pode designar uma *arquitetura*. Por fim, as mudanças empreendidas pressupõem a capacidade de se alterar e inovar via *software* as próprias funções constituídas a partir do modelo. E pelo fato de serem desenvolvidos para desempenhar uma função bem definida, e baseados em paradigmas e interfaces padronizados, tais *softwares* implicam na modificação da dinâmica de agregação de valor às redes de computadores – anteriormente orientada à inovação baseada em *hardware*, mais custosa e menos comoditizada. Sob esta ótica mais ampla, "OpenFlow" torna-se um ecossistema.

A arquitetura OpenFlow prevê a generalização da função desempenhada pelos *switches* de uma rede, renomeando-os *datapaths*. Tal generalização faz com que cada *datapath*, ao receber um pacote de dados para encaminhamento, baseie sua decisão em entradas de uma tabela de fluxos. Esta tabela possui colunas que descrevem possíveis características das unidades de tráfego a serem encaminhadas. E um campo final descreve as próprias ações a serem realizadas para aquelas que possuam estas mesmas características.

A Figura 2 exibe uma estrutura detalhada da tabela de fluxos, além dos aspectos que

um *datapath* compatível com a especificação 1.1.0 do protocolo OpenFlow é capaz de avaliar. Observe-se que, em complemento às estruturas de avaliação e instruções de tratamento de dados, designam-se "contadores", que armazenam informações como o número de pacotes e o total de *bytes* que coincidiram com cada entrada até o momento. Também é possível notar que os termos em que as características de um pacote são descritas não abrangem apenas o universo da camada de enlace, o que poderia parecer natural. Isto porque, enquanto arquitetura, o OpenFlow preconiza que os *datapaths* tomem decisões baseados em múltiplas camadas da pilha de protocolos da Internet, o que se opõe ao escopo do encaminhamento em *switches* tradicionais.

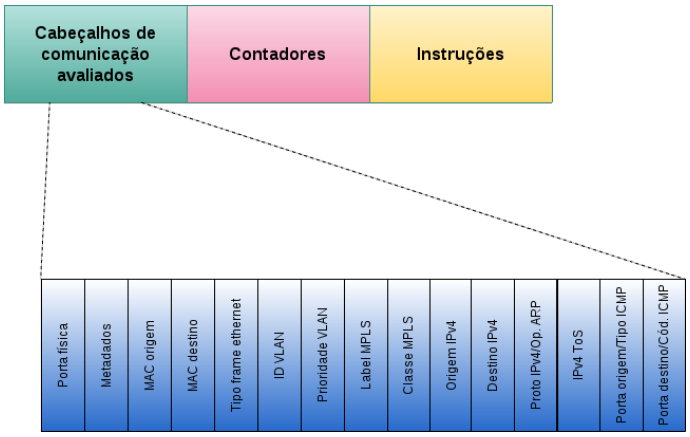


Figura 2: Visão esquemática de uma plataforma virtual de roteamento IP.

O OpenFlow implementa a noção de SDN através de seu protocolo. Ele é usado para transmitir a um *datapath* instruções quanto a inserir, modificar ou remover entradas de sua tabela de fluxos. Isto porque sua definição não acontece nos próprios dispositivos, mas originam-se em um novo elemento da rede, o controlador OpenFlow. O controlador pode se tratar de um servidor (ou outro tipo de nó computacional) que implementa uma lógica de decisão a partir da qual são derivadas as instruções a serem transmitidas.

Os dados de entrada para a lógica decisória dos controladores são os próprios cabeçalhos dos pacotes recebidos por um *datapath*. Assim, tão logo este tipo de elemento tenha uma unidade de tráfego a tratar, extrai os dados de seu cabeçalho e os remete, como uma pergunta, ao controlador. Isto dispara o processo de derivação de entradas que, após concluído, permitirá ao controlador instruir que tratamento o *datapath* deve aplicar ao

pacote.

Ao serem incorporadas à tabela de fluxos, as instruções de encaminhamento recebidas não precisam dar andamento apenas ao tratamento do pacote que as motivou. É da natureza da comunicação Internet basear-se na fragmentação do tráfego de uma comunicação em múltiplos pacotes. Assim, uma entrada de tabela de fluxos pode ser mantida por um tempo arbitrário da memória de um *datapath*. Porquanto ela seja mantida, todos os pacotes recebidos por um *datapath* que tenham características coincidentes com as especificadas por ela receberão automaticamente o mesmo tratamento. Isto tem o efeito de tornar duráveis as decisões tomadas por um controlador, aumentando a eficiência da arquitetura pela preservação de recursos de comunicação e processamento.

Uma vez que a lógica de decisão dos elementos controladores pode ser implementada como *software* e basear-se em combinações arbitrárias de campos de cabeçalhos, a arquitetura OpenFlow pode facilitar a inovação, experimentação e operacionalização de idéias no âmbito das redes de computadores. De fato, existem soluções do tipo que embutem à operação de redes noções tão diversas quanto segurança (BALLARD; RAE; AKELLA, 2010) (BRAGA; MOTA; PASSITO, 2010) e balanceamento de carga (HANDIGOL et al., 2010) (WANG; BUTNARIU; REXFORD, 2011). O próprio conceito de virtualização de redes é materializado na arquitetura por meio de uma aplicação (SHERWOOD et al., 2010). Trata-se da ferramenta FlowVisor, que permite a um operador particionar os recursos de um *datapath* (suas portas de comunicação, basicamente) e reagrupá-los logicamente em diferentes topologias virtuais, que apesar de serem suportadas pelos mesmos dispositivos, operam de maneira isolada.

Para reduzir a lacuna entre o tratamento de informações de baixo nível envolvidas no protocolo OpenFlow e a construção de aplicações que interajam com esta arquitetura, *softwares* controladores de uso geral também estão disponíveis (GUDE et al., 2008) (STANFORD, 2010). Estas ferramentas possuem projetos "plugáveis" de código, e oferecem aos programas que implementem suas interfaces abstrações de mais alto nível que aquela cabível em nível de protocolo. A semelhança entre este papel e a intermediação efetuada por

um sistema operacional em relação a *hardware* e *softwares* aplicativos, faz com que tais controladores também recebam a classificação de "sistemas operacionais de rede" (GUDE et al., 2008).

2.4 Arquiteturas virtuais de roteamento IP

Enquanto a tecnologia de redes definidas por *software* torna o encaminhamento de tráfego dentro da mesma sub-rede uma decisão trivial por parte dos elementos do plano de controle, ferramentas de virtualização são aplicadas nas plataformas virtuais de roteamento IP para permitir que um fluxo de dados seja transmitido entre domínios de *broadcast* distintos.

Nestas arquiteturas, o plano de controle é representado por um sistema controlador, que possui em sua memória uma representação da topologia da rede controlada. Isto pode ser visto na Figura 3, que mostra uma rede atendida por uma plataforma virtual de roteamento IP. Os n comutadores da rede estão interligados em série, tornando-a plenamente conexa no nível de enlaces. Cada comutador físico é representado na memória do controlador por uma instância de um mecanismo de conectividade virtual.

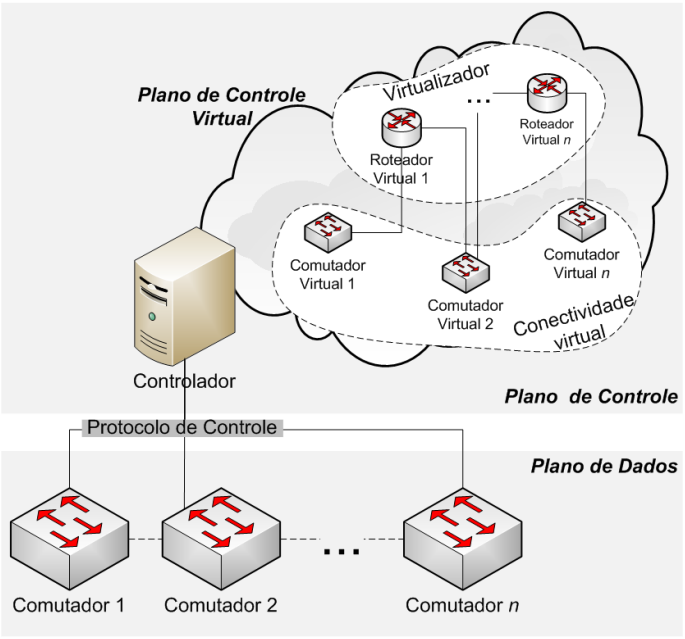


Figura 3: Visão esquemática de uma plataforma virtual de roteamento IP.

Em uma arquitetura tradicional, para permitir a comunicação entre clientes desta rede configurados em diferentes domínios de *broadcast* seria preciso empregar um dispositivo com função de roteamento (um roteador, por exemplo). No cenário apresentado, porém, MVs executando processos de roteamento são suficientes para que esta função seja desempenhada.

Para conhecer a rota a ser seguida por um fluxo do plano de dados entre domínios de *broadcast* distintos, o controlador consulta as bases de informações de roteamento dos processos do plano de controle virtual. Assim, é possível determinar por quais enlaces os dados devem ser transmitidos para chegar a seu destino, bem como comandar os elementos do plano de dados de acordo.

Foram pesquisadas na literatura propostas de plataformas virtuais de roteamento IP, descritas a seguir.

O trabalho em Wang et al. (2008) propõe um *hipervisor de plano de dados*. Ao operar em conjunto com um plano de encaminhamento IP baseado em roteadores virtuais, esta solução permitiria a migração de processos de roteamento entre diferentes nós físicos “a quente", isto é, sem que fosse necessário interromper seu funcionamento. As ferramentas adotadas são o virtualizador OpenVZ e o *framework* de conectividade virtual TUN/TAP (KRASNYANSKY, 2005). Não obstante, o autor do trabalho também relata haver testado o uso do virtualizador Xen (BARHAM et al., 2003), mas que os recursos de migração de MVs disponíveis para esta ferramenta não permitiam uma migração bem-sucedida do plano de controle.

Apesar do uso intensivo de virtualização, a arquitetura de Wang et al. (2008) não prevê a segregação do tráfego do plano de controle na memória de um controlador.

Em Bolla et al. (2009) se propõe uma arquitetura chamada DROP, em que a virtualização de processos roteadores é substituída pela execução de múltiplos processos de roteamento modificados para operar através de *links* virtuais. Tais canais de comunicação são estabelecidos a partir de uma API de comunicação inter-processos do sistema Linux,

e se destinam à troca de informações do plano de controle. Os processos modificados também podem executar sob diferentes nós físicos. Apesar disto, a arquitetura depende da existência de um segmento de rede dedicado ao plano de controle, o que limita a flexibilidade e a escalabilidade da arquitetura.

A arquitetura introduzida em Nascimento et al. (2010) propõe o uso de um virtualizador, mas sugere-se que a especificação da ferramenta a ser utilizada seja objeto de estudo posterior. A conectividade virtual neste caso é provida pelo *framework* TUN/TAP.

As funções desempenhadas pelos virtualizadores e pela conectividade virtual no contexto das arquiteturas estudadas são bem conhecidas. Ainda assim, não foram encontradas análises que visassem justificar a escolha das ferramentas para compor o plano de controle virtual das mesmas, quer sob o ponto de vista funcional, quer de desempenho.

A literatura também fornece poucos detalhes a respeito de como refletir no plano de controle eventos ocorridos no plano de dados, como a indisponibilidade de um enlace. A solução em Wang et al. (2008) parte do princípio de que tais eventos são previsíveis, enquanto em Bolla et al. (2009) processos e protocolos específicos para o tratamento dos mesmos são propostos. Em Nascimento et al. (2010) nenhuma abordagem específica é relatada, mas fica documentada a necessidade de uma avaliação aprofundada do tema.

3 Proposta de roteamento como serviço

Uma plataforma de roteamento como serviço pode ser definida como um artefato capaz de instruir o plano de dados de uma rede definida por *software* com relação a roteamento de pacotes IP e que conte com as seguintes características:

1. A capacidade de definir-se os critérios de roteamento desejados (a política de negócio) em um nível de abstração que dispense a necessidade de conciliar múltiplas configurações individuais e independentes.

2. Uma base de informações sobre a topologia física da rede a ser controlada. Esta base permite que a plataforma determine a reserva de insumos de conectividade disponíveis e faça uso dos mesmos conforme necessários.

3. Elementos do plano de controle virtual escaláveis e flexíveis, que permitam seu gerenciamento de forma programática. Isto visa a transferência do esforço da configuração destes elementos, antes realizada pelos operadores da rede, para a lógica da plataforma.

Este arcabouço está representado na Figura 4, onde pode se visualizar os três elementos citados, além do fluxo de dados e operações entre cada um deles. Uma pesquisa bibliográfica por contribuições que visassem o mesmo espaço de soluções foi realizada.

Talvez fosse possível solucionar os problemas identificados combinando-se uma arquitetura virtual de roteamento IP e o RCP (CAESAR et al., 2005), que permite propagar

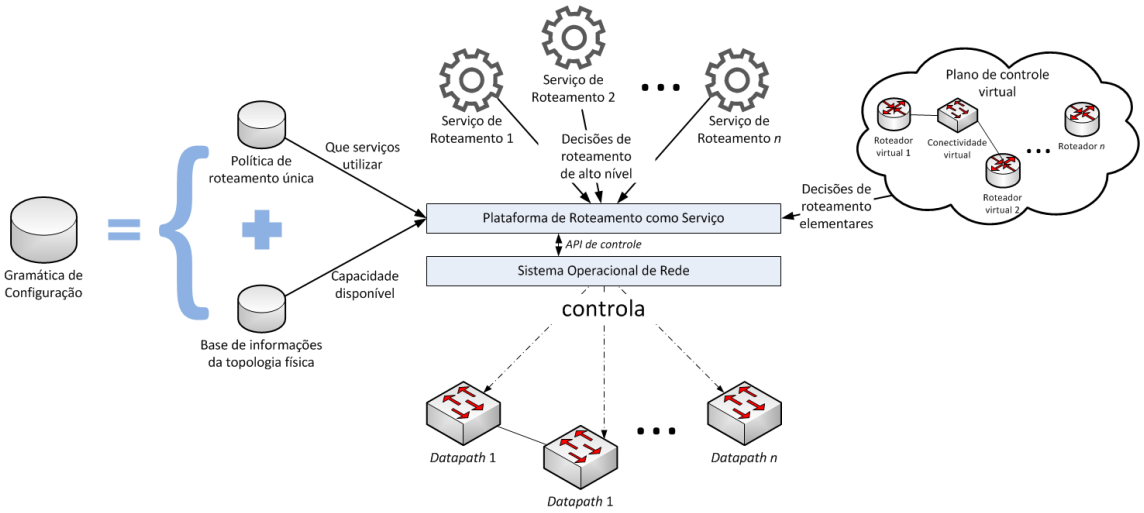


Figura 4: Representação conceitual de uma plataforma de roteamento como serviço.

decisões de roteamento a partir de um sistema central. Porém, tal abordagem implicaria em um aumento da complexidade do plano de controle virtual, o que também aumentaria o esforço de gerenciamento de seus componentes.

Há em Keller e Rexford (2010) uma proposta de modelo de PaaS para o roteamento IP. Esta proposta se aproxima da característica "1" definida anteriormente (i.e. abstração dos critérios de roteamento) ao sugerir que o serviço de encaminhamento de pacotes seja apresentado ao operador como a abstração de um roteador único. Porém, concentra-se na incorporação de uma interface programática ao plano de controle, através da qual diferentes aplicações-cliente informariam a política de encaminhamento desejada. Tal interface ficaria responsável por garantir que os requisitos fossem refletidos no tratamento do tráfego passante. A interface também seria responsável por estabelecer sessões de protocolos de roteamento e oferecer às aplicações clientes acesso compartilhado e isolado às mesmas. Apesar de promissora, esta abordagem exige a construção de artefatos que implementem a interface sugerida. Em contrapartida, a solução aqui apresentada já a incorpora na forma da gramática de configuração.

Uma última perspectiva de avanço pôde ser encontrada em Foster et al. (2011), onde uma linguagem de descrição das características operacionais de uma rede definida por *software* é apresentada. Esta poderia ser uma ferramenta útil na construção de uma pla-

taforma de roteamento como serviço, mas ainda seria necessário construir um arcabouço de tradução da política de negócio em instruções que permitissem realizar o encaminhamento conforme pretendido.

Isto posto, a arquitetura virtual de roteamento IP QuagFlow proposta em Nascimento et al. (2010) foi escolhida para um trabalho experimental de transição para uma plataforma de roteamento como serviço. Esta ferramenta foi selecionada por ter seu código-fonte livremente disponível e por encontrar-se em um estágio de implementação mais avançado em comparação com iniciativas semelhantes. O código-fonte para o trabalho em Bolla et al. (2009) não pôde ser obtido, e em Bozakov (2010) apenas um esquema de operação é apresentado.

A seguir, a arquitetura QuagFlow existente ao início do processo de transição é apresentada em detalhes. Nas seções posteriores, o texto concentra-se nas ações consideradas necessárias para incorporar ao QuagFlow as características desejadas.

3.1 Apresentação da arquitetura QuagFlow[1]

A QuagFlow é uma arquitetura virtual de roteamento IP. Assim, tem por objetivo primário realizar o encaminhamento de pacotes com base em seus endereços IP de origem e destino. Sua implementação de referência é baseada em um componente desenvolvido para o controlador de redes OpenFlow NOX chamado QuagFlow-C, mais uma ferramenta de coleta de informações do plano de controle chamada QuagFlow-Slave (ou QuagFlow-S). Todos os elementos têm o sistema operacional Linux como plataforma-alvo, e tratam-se de programas escritos na linguagem C++. As máquinas virtuais (MVs) da topologia virtual executam sob o virtualizador QEMU.

Uma visão esquemática da arquitetura pode ser vista na Figura 5. Os *datapaths* a

[1]Nesta seção, a visão geral da arquitetura QuagFlow foi escrita a partir das informações disponíveis em Nascimento et al. (2010). Muitos dos detalhes de sua operação, porém, foram obtidos consultando diretamente o código-fonte da solução. Uma pequena e última parte das informações referenciadas também foi adaptada de artigos publicados em conseqüência do progresso desta pesquisa, quando a arquitetura já havia sido rebatizada RouteFlow. Referências bibliográficas foram incluídas conforme aplicável.

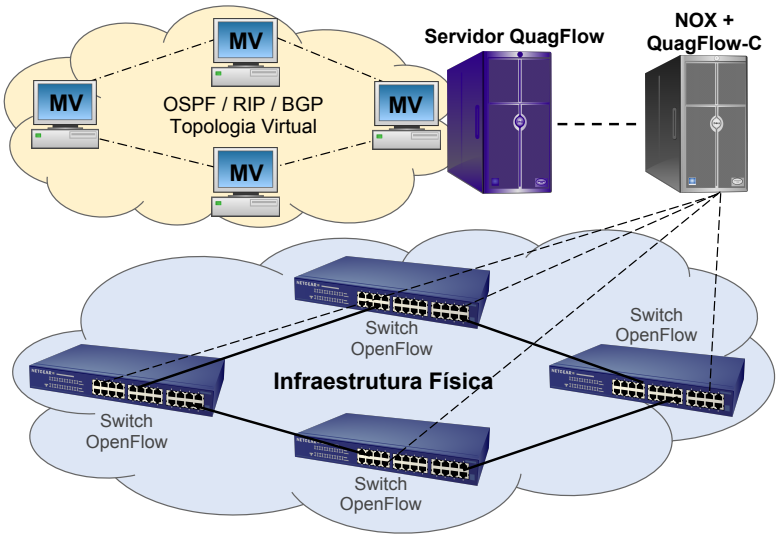
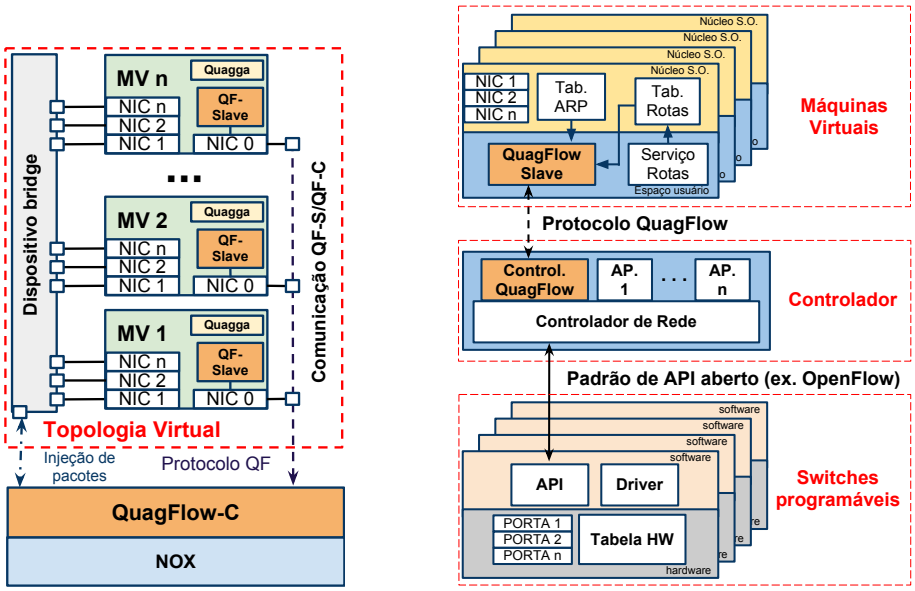


Figura 5: Visão esquemática da arquitetura QuagFlow (NASCIMENTO et al., 2010)

serem controlados conectam-se a uma instância do NOX que inclui a aplicação de rede QuagFlow-C. No mesmo servidor em que executa o QuagFlow-C, MVs são carregadas para a execução de processos de roteamento, cujas decisões serão utilizadas para comandar o encaminhamento no plano de dados. Neste caso, considera-se que cada *datapath* controlado deve ser mapeado em caráter 1:1 com uma MV executando a suíte de roteamento Quagga. Da mesma forma, cada porta em um *datapath* corresponde a um adaptador de rede virtual na MV associada.

As interfaces de rede mapeadas para portas de *datapaths* correspondem a dispositivos do tipo TAP, que podem ser usadas como uma via ponto-a-ponto entre as MVs e o anfitrião. De fato, é através delas que o controlador realiza operações de injeção de tráfego no plano de controle, detalhadas mais adiante. Isto também indica que a conectividade virtual não reproduz fielmente a conectividade do plano de dados, mas é uma aproximação. Além dos adaptadores vinculados a cada porta do plano de dados, cada MV conta com um dispositivo virtual que simula uma conexão de rede com o sistema operacional anfitrião. Através deste dispositivo, o componente QuagFlow-S pode se comunicar com o QuagFlow-C.

A Figura 6a apresenta os dois tipos de conexões de rede existentes, tanto a que representa o plano de controle quanto a utilizada para comunicação com o anfitrião.



(a) Conectividade das MVs do plano de controle QuagFlow

(b) Fluxo de operação QuagFlow (NASCIMENTO et al., 2010)

Figura 6: Operação e conectividade da arquitetura QuagFlow

O QuagFlow-S é carregado durante o processo de inicialização das MVs. Assim, tão logo uma delas torna-se plenamente operacional, apresenta-se ao controlador. Caso algum *datapath* tenha feito contato com o controlador antes da inicialização da MV, ambos são associados. Se não houverem *datapaths* aguardando no momento do carregamento de um nó virtual, o mesmo é posto em modo de espera. Constitui-se então uma reserva de máquinas virtuais, que poderão ser associadas a dispositivos que se conectem posteriormente.

A tabela de fluxos de um *datapath* é inicializada com uma série de entradas permanentes logo após a vinculação a uma MV. Tais entradas dividem-se em duas categorias:

1. *Tratamento de mensagens de roteamento e ARP.* Instruem o *datapath* a transmitir para o controlador todos os pacotes dos protocolos de roteamento OSPF, RIP e BGP. Sua função é garantir que roteadores legados que sejam conectados a um *datapath* possam estabelecer adjacência com os roteadores do plano de controle tal como em uma rede tradicional. Também é criada uma regra que transmite para o plano de controle pacotes de resolução ARP, o que viabiliza a conectividade de camada 2 entre as MVs do plano de controle e qualquer dispositivo do plano de dados. Este

grupo de regras tem a mais alta prioridade possível, o que garante que sua aplicação tem precedência sobre qualquer outra.

2. *Envio de tráfego para o controlador.* Trata-se de uma única instrução que encaminha todo o tráfego recebido para o plano de controle. Esta regra tem a mais baixa prioridade possível, de forma que pacotes que não sejam mensagens de roteamento só serão transmitidos para a rede virtual quando não casarem com nenhuma outra entrada da tabela de fluxos. Isto significa que, no instante seguinte à inicialização da arquitetura, todo o tráfego recebido por um *datapath* será enviado ao plano de controle.

Após a inicialização, pedidos de transmissão de pacotes nos planos de dados e de controle começam a ser tratados. Cada possível solicitação está listada a seguir, acompanhada de sua respectiva ação de tratamento:

- *Encaminhamento de pacote no plano de dados.* O pacote é injetado no dispositivo TAP correspondente à porta por onde o pacote foi recebido. Caso seu destino final seja a própria MV, a comunicação prosseguirá normalmente – transparentemente intermediada do controlador. Caso se destine a outro nó, a MV o receberá e o submeterá à sua lógica de roteamento.

- *Encaminhamento de um pacote do plano de controle.* De acordo com a interface TAP por onde o pacote foi recebido, o QuagFlow-C determina a porta de *datapath* correspondente. Assim, o pacote é copiado para o plano de dados e transmitido através dela.

Até então, o comportamento dos *datapaths* controlados pelo QuagFlow imita uma operação legada. Como cada porta está associada a uma interface de rede virtual, o efeito prático é o de que qualquer elemento conectado a ela pode usar a respectiva MV como um *gateway*. Pacotes transmitidos serão enviados ao plano de controle, a MV aplicará sua lógica de roteamento e repassará os pacotes pela interface adequada. A cópia destes

pacotes de volta ao plano de dados permitirá que o nó de destino – ou mesmo um roteador legado conectado a um *datapath* – receba-o apropriadamente. Se dois *datapaths* estiverem conectados entre si, analogamente a dois roteadores conectados, a mesma lógica se aplica.

Suponha-se um pacote a ser transmitido, recebido por uma porta de *datapath* associada à interface A_1 da MV A. Este pacote deve ser encaminhado através de outro *datapath* para alcançar seu destino final. O nó de destino é um computador conectado a uma porta vinculada à interface B_1 de uma MV B. Ambos os dispositivos têm uma conexão entre si. Esta conexão está vinculada ao adaptador A_2 no caso do primeiro *datapath*, e ao adaptador B_2 para o segundo. Assim, o pacote precisa ser encaminhado através do conjunto ordenado de interfaces $C = \{A_1, A_2, B_2, B_1\}$ para ser entregue corretamente.

Tão logo o pacote seja recebido no plano de dados, será copiado e entregue para A_1. Após consultar sua tabela de rotas, a MV A o transmite pela interface A_2. O pacote será copiado para o plano de dados e transmitido pela porta que está conectada ao segundo *datapath*. Portanto, ele será reinjetado no plano de controle, desta vez através da interface B_2. A MV B encaminhará o pacote para B_1, o que fará com que ele seja transmitido pela porta correta e entregue a seu destino final.

O encaminhamento de tráfego através do plano de controle certamente afetará a FIB de suas MVs. A transmissão de pacotes motivará o aprendizado de endereços MAC e o disparo de resoluções ARP por parte das MVs. Segundo o exemplo, a MV A adicionará o endereço MAC da origem do pacote recebido via A_1 imediatamente à sua FIB. Igualmente, a MV B poderá emitir um pedido de resolução ARP para obter o endereço do adaptador do nó de destino antes de efetivamente encaminhar seu pacote. A FIB das MVs também será afetada pela execução de seus respectivos processos de roteamento. Como se pôde ver, a comunicação entre os elementos do plano de controle ocorre normalmente (com o intermédio do plano de dados), e em pouco tempo após a inicialização os roteadores virtuais podem começar a trocar mensagens entre si.

A modificação da FIB das MVs é tratada conjuntamente pelo QuagFlow-C e pelos

processos QuagFlow-S executando em cada nó virtual, conforme representado pela Figura 6b. Periodicamente cada processo QuagFlow-S consulta a FIB da MV em que está executando. Em resposta à detecção de modificações nesta infraestrutura, as ações a seguir podem ser disparadas:

- *Aprendizado de um novo endereço MAC.* O QF-S envia uma mensagem ao QF-C informando o adaptador de rede ao qual a descoberta está vinculada, o endereço MAC e o IP para qual ele está mapeado. Ao receber esta informação, o QF-C adiciona uma entrada na tabela de fluxos do *datapath* associado à MV. Esta regra faz com que qualquer pacote recebido pelo plano de dados, com o endereço IP de destino do nó descoberto, seja transmitido pela porta física para a qual a interface da MV está mapeada. A regra também modifica o endereço MAC de destino do pacote para que coincida com o MAC recém-aprendido, para que o dispositivo recebendo-o possa identificá-lo corretamente.

- *Remoção de um endereço MAC da FIB.* O QF-S notifica o QF-C da remoção e de suas características, para que a entrada correspondente na tabela de fluxos do plano de dados seja removida.

- *Acréscimo de uma nova rota.* Caso uma nova rota seja incluída à tabela do sistema (possivelmente como parte do processo de convergência de protocolos) o QF-S também envia uma mensagem ao QF-C para informá-la, desta vez contendo o prefixo de destino, o endereço MAC do *gateway* e a interface de rede à qual a rota está vinculada. Novamente, o QF-C pesquisa as associações entre interfaces de MVs e portas de *datapaths*, e esta informação é utilizada para inserir uma entrada na tabela de fluxos dos dispositivos. A regra comanda o envio de pacotes que tenham como destino o novo prefixo para a porta respectiva à interface de rede de saída. Ela também modifica o endereço MAC de origem do pacote para que coincida com o endereço MAC da interface da MV, e o MAC de destino para que seja igual ao do *gateway*.

- *Remoção de uma rota.* Assim como no caso de remoção de um endereço MAC, o QF-S notifica o QF-C do evento, que fica responsável por remover a entrada correspondente no plano de dados.

As regras inseridas podem receber diferentes prioridades, de acordo com o tipo de evento que motivou sua criação. Aquelas resultantes de uma nova entrada MAC recebem sempre a mesma prioridade, que é apenas uma unidade superior à da regra de último recurso, de envio de pacotes para o controlador.

As regras criadas em resposta a uma nova rota recebem um valor de prioridade fixo v, superior ao atribuído às regras de entradas MAC, somado ao número de *bits* da máscara de rede de seu prefixo. Isto serve a dois propósitos. Primeiro, para que as instruções do plano de dados possam se dividir em quatro categorias de prioridades bem definidas e avaliadas nesta seqüência: tratamento de mensagens de roteamento, encaminhamento para destinos remotos, encaminhamento para destinos diretamente conectados e o último recurso, envio para o controlador. Em segundo lugar, para que as regras referentes a destinos remotos sejam avaliadas conforme a lógica de máscara de prefixo mais longo. Para citar um exemplo, rotas para prefixos mais específicos, com máscara "/24", sempre receberão prioridade superior às rotas para destinos de prefixo "/16" ($v+24 > v+16$).

O ponto central da operação da arquitetura QuagFlow reside no efeito prático do acréscimo das entradas às tabelas de fluxos. Elas farão com que os pacotes cujas características se aplicam a elas sempre sejam tratados diretamente no plano de dados, de forma similar às MVs de roteamento (ou um roteador legado) – tanto do ponto de vista de mudança de endereços MAC de origem e destino quanto do envio pela porta seguinte, até a entrega ao seu destino final. De fato, para uma conexão entre um par de nós de origem e destino cujos endereços MAC ainda não façam parte da FIB das MVs do plano de controle, apenas o *primeiro* pacote será encaminhado com a interação das MVs, alcançando a regra de último recurso dos *datapaths* e sendo injetado no plano de controle. Todos os pacotes seguintes serão tratados pelas instruções incorporadas ao plano de dados em

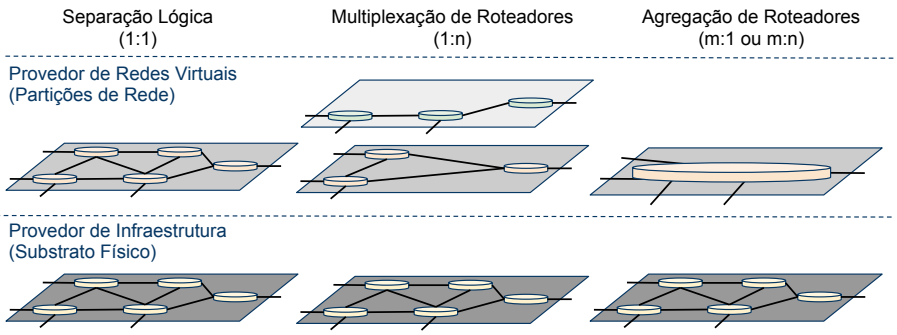


Figura 7: Casos de uso da arquitetura QuagFlow (NASCIMENTO et al., 2011b)

resposta ao encaminhaminhamento do primeiro.

Um estudo detalhado do desempenho desta arquitetura pode ser encontrado em Nascimento et al. (2011a). Este trabalho também marca a mudança do nome QuagFlow para RouteFlow. Uma vez que a presente pesquisa foi planejada e iniciada a partir da publicação de Nascimento et al. (2010), o uso nome QuagFlow foi mantido até o Capítulo 4. A partir do Capítulo 5, que documenta experimentos realizados após a mudança, o nome RouteFlow é adotado.

Em Nascimento et al. (2011b) o espaço de solução coberto pelo QuagFlow é amplicado através da apresentação de novos e diferentes casos de uso para o sistema. Não são fornecidos, porém, detalhes para a implementação de qualquer deles.

A operação aqui documentada é enquadrada no caso de uso "1:1", trivial, pelo fato de cada *datapath* controlado ter uma relação unívoca para com as MVs do plano de controle. Também são apresentados os casos "1:n" e "m:n". O primeiro considera a possibilidade de se mapear um mesmo *datapath* para múltiplas MVs – cada uma delas representando um roteador de diferentes redes virtuais que compartilhassem o mesmo substrato físico. O caso "m:n" suportaria não só esta multiplexação de MVs, mas também a vinculação de múltiplos *datapaths* a uma única MV, o que poderia ter o efeito prático de que tais *datapaths* operassem como um único dispositivo "agregado". A Figura 7 ilustra os três casos de uso e exemplifica, para cada um deles, de que forma a infraestrutura física poderia ser mapeada para o plano virtual.

Na próxima seção serão detalhadas as intervenções planejadas à arquitetura QuagFlow para reposicioná-la como uma plataforma de roteamento como serviço.

3.2 Transição para o roteamento como serviço

A noção de casos de uso apresentada em Nascimento et al. (2011b) aponta na direção de uma arquitetura virtual de roteamento IP que não só converta a FIB de máquinas virtuais em instruções do plano de dados (caso "1:1"), mas que aplique lógicas complementares neste processo. No caso "1:n", é necessário adotar uma técnica para isolar o tráfego e as decisões respectivas a diferentes redes virtuais.

E no caso "m:n", como tratar um pacote recebido pela porta *x* de um *datapath A* e que precisa ser encaminhado para a porta *y* de um *datapath B*, quando estes dois dispositivos não estão diretamente conectados? Encaminhando-o diretamente pelo plano de controle? Inserindo no plano de dados entradas que permitam compor uma "rota" até o destino final?

Esta pesquisa pode ajudar a responder tais perguntas, já que o caso de uso "m:n" é objeto da aplicação de roteamento como serviço apresentada no Capítulo 5. Aqui propõe-se que os casos de uso que flexibilizam o mapeamento entre *datapaths* e MVs fazem todo o sentido, mas tratam-se apenas de instâncias de um problema maior: o de que os instrumentos do roteamento tradicional não são mais suficientes para traduzir toda a complexidade e diversidade das políticas de negócio de operadores de redes.

As próximas seções detalham as características e intervenções propostas como necessárias para permitir que a plataforma de roteamento como serviço vislumbrada possa realizar não só os casos de uso "1:1", "1:n" e "m:n", mas ser estendida para incorporar virtualmente *qualquer* outra lógica de encaminhamento.

3.2.1 Abstração única da política de roteamento

Conforme apresentado no na Seção 2.1 do Capítulo 2, as ferramentas BGP usadas para tornar a convergência BGP aproximada ou inteiramente previsível possuem limitações relevantes. Soluções para sua modificação, ou mesmo externas ao protocolo, foram propostas na literatura.

Em Zhang-Shen, Wang e Rexford (2008) a proposta de roteamento atômico para um AS é apresentada como conseqüente a duas modificações no protoco BGP: 1) que cada roteador em um sistema autônomo possa escolher mais que uma "melhor rota"para um prefixo (decisão de roteamento), para então diferenciá-las em função de termos de mais alto nível que o simples número de saltos entre ASs; 2) que um roteador possa propagar rotas diferentes daquela selecionada por ele como a mais adequada para alcançar determinado prefixo. Assim, espera-se otimizar o espaço combinatório R à disposição de cada dispositivo, de forma que se possa sempre escolher um conjunto de rotas que permita satisfazer a função $P(R)$ que implementa a política de negócio vigente. *Como* sintetizar tal função, porém, é uma tarefa que fica reservada para trabalhos futuros.

O trabalho de Caesar et al. (2005) detalha a *Routing Control Platform* (RCP, ou “Plataforma de Controle de Roteamento”), uma estrutura responsável por estabelecer adjacências IGP e iBGP com um grupo de roteadores em um mesmo domínio administrativo, calcular as melhores rotas para cada prefixo alcançável por eles e disseminar esta informação. A proposta se assemelha ao conceito de *route-reflectors* quando exige sessões iBGP entre os roteadores e a RCP, mas se diferencia principalmente porque a plataforma é capaz de divulgar diferentes rotas para cada roteador, enquanto RRs disseminam uma única melhor rota para seus pares. Desta forma “a RCP combina a corretude intrínseca de uma configuração iBGP *full-meshed* e os benefícios de escalabilidade dos *route-reflectors*” (CAESAR et al., 2005). Não obstante, os autores ponderam que a abordagem de centralizar o cálculo de rotas para toda a topologia de um AS pode ser um grande desafio sob a ótica de desempenho.

O universo de soluções estudadas considera que uma vez que o cálculo de saltos entre redes e alcançabilidade do protocolo BGP resulte em um conjunto previsível em cada roteador, poder-se-á ajustar o processo decisório de cada dispositivo para que sempre satisfaça a política de roteamento vigente.

Nenhum dos trabalhos pesquisados propõe uma lógica de roteamento que não resulte de múltiplas decisões independentes sobre a conectividade e o vetor de distância entre redes. Zhang-Shen, Wang e Rexford (2008) sugere que uma função $P(\cdot)$ seja aplicada a tais informações para obter decisões compatíveis com a política de negócio, mas não a fornece. A premissa observada também parece refletir o fato de que o roteamento tradicionalmente vem sendo executado como um processo distribuído: mesmo a RCP, uma abordagem com viés centralizador, faz a ressalva de que um roteador pode calcular uma rota ótima, a partir de sua visão parcial da rede, antes da plataforma central – e de que tal resultado deve ter prevalência (CAESAR et al., 2005).

Neste momento é importante lembrar que nas arquiteturas virtuais de roteamento IP decisões triviais de um protocolo como o BGP são tomadas por processos individuais, virtualmente conectados. Em seguida seus resultados são coletados e submetidos a um processo controlador. Tal processo é o único responsável por traduzir decisões de encaminhamento em instruções do plano de dados, que darão efeito às próprias decisões!

Desta forma, esta pesquisa vem propor que a QuagFlow, por sua característica de "pós-processamento" centralizado, pode incorporar a lógica necessária para atender a uma política de negócio arbitrária, dependendo apenas da disponibilidade de meios para que o operador da rede a descreva – uma gramática de configuração.

3.2.1.1 Premissas para a abstração única

A proposta de uma gramática é apoiada por outras proposições, menos gerais e mais técnicas, elencadas a seguir.

Proposição 1 (Expressividade)**.** *Uma política de roteamento deve ser integralmente des-*

crita nos termos da gramática para que possa ser completamente atendida.

Toda lógica de processamento complementar àquela existente nos protocolos de roteamento suportados pelo QuagFlow deve ser implementada como parte do componente QF-Server. Cada unidade de lógica, que representa um *serviço de roteamento* específico, deve ser disponibilizada ao operador da rede como um elemento específico da sintaxe da gramática de configuração. Caso a execução da funcionalidade incorporada dependa de parâmetros de entrada, estes também deverão poder ser expressos como elementos da mesma sintaxe.

Para ilustrar tal proposição, é possível supor que o exemplo abaixo represente uma construção válida na gramática de configuração QuagFlow:

```
load-share 8002 8003
```

Onde "load-share" é um termo que ativa a lógica de compartilhamento de tráfego entre diferentes ASs. Os números 8002 e 8003 são dois exemplos de parâmetros a serem fornecidos ao primeiro termo que designam os ASs que serão alvo desta operação. Também se poderia supor que o comportamento implementado pelo termo "load-share" elencasse um conjunto de rotas para prefixos $R = \{r_1, r_2, ..., r_n\}$ que fosse anunciado por ambos os ASs e, para tais rotas, o plano de controle produzisse continuamente regras de encaminhamento com um tempo curto de expiração. Tão logo uma regra se expirasse, seria substituída por outra, e a cada regra instalada as rotas disponíveis para um mesmo prefixo seriam alternadas. Neste caso, poder-se-ia dizer que a expressividade da gramática de configuração é suficiente para modelar uma política de negócio que exija o *serviço* de compartilhamento de tráfego entre diferentes ASs.

A título de prova de conceito, o Capítulo 5 deste trabalho faz registro da incorporação de instrumentos de descrição para realização de uma lógica de processamento específica na plataforma QuagFlow.

Proposição 2 (Satisfabilidade)**.** *A política de roteamento descrita através da gramática*

de configuração deverá ser sempre satisfazível pela topologia subjacente.

Uma vez que o mapeamento ótimo de recursos físicos para redes virtuais é um problema NP-Difícil (ALKMIM; BATISTA; FONSECA, 2011), é responsabilidade do operador especificar e prover a infraestrutura capaz de atender à descrição de sua política de negócio. Não obstante, assim como o nível de abstração oferecido por uma plataforma de virtualização permite uma visão consolidada dos recursos disponíveis, mas não pode utilizá-los além de seus limites *per se*, é desejável que a plataforma vislumbrada rejeite uma descrição de política que obviamente exceda a capacidade da infraestrutura existente.

A segunda característica proposta neste trabalho, uma base de informações sobre a topologia física disponível, também pode instrumentalizar um mecanismo de avaliação de satisfabilidade elementar que faça parte da lógica de um serviço de roteamento implementado na plataforma QuagFlow.

Proposição 3 (Tempestividade)**.** *A função $P(R)$ deve ser sempre recalculada, bem como seus resultados transpostos para o plano de dados, em resposta a uma nova decisão de roteamento (convergência).*

Aqui propõe-se complementar o processo decisório dos protocolos de roteamento existentes, tal como outras propostas de instrumentos de modelagem de uma política de negócio (CAESAR et al., 2005) (ZHANG-SHEN; WANG; REXFORD, 2008). Porém, é importante ressaltar que tais decisões não são perenes, mas altamente voláteis. Em uma rede tradicional, convergências dos protocolos de roteamento acontecem em resposta a toda modificação de topologia, seja motivada por desconexões, reconexões ou o estabelecimento de novas enlaces, por exemplo.

Os trabalhos identificados na literatura podem garantir, por definição, que uma dada política seja continuamente atendida por suas contribuições, já que a lógica de processamento que propõem é incorporada ao processo decisório dos elementos roteadores. Na arquitetura QuagFlow, porém, a identificação e resposta a novas convergências baseia-se em uma coleta periódica das tabelas de encaminhamento de cada roteador virtual. En-

quanto pode ser suficiente para um amplo espectro de cenários, pode-se dizer que este comportamento não é ótimo. Caso uma nova rota para um prefixo seja escolhida por um roteador no instante t_1 imediatamente posterior a uma consulta à sua tabela, a política de encaminhamento permanecerá desatualizada até que a coleta seguinte aconteça.

Desta forma, para garantir que a política vigente sempre reflita as informações produzidas pela última convergência ocorrida na rede controlada, o QuagFlow deverá incorporar um mecanismo de resposta a eventos de roteamento. Tal mecanismo será responsável então por atualizar a lógica de encaminhamento de acordo com as decisões de roteamento obtidas.

A implementação desta proposição é objeto de estudo realizado no Capítulo 5 deste trabalho.

Espera-se que enquanto as três proposições detalhadas se mantiverem satisfeitas será possível especificar, por meio da plataforma de roteamento em construção, um conjunto de *serviços de encaminhamento* que atenda a uma política de negócio vigente. Porém, para alcançar o grau de correspondência desejado entre uma plataforma de roteamento como serviço e o nível de abstração provido por aplicações de virtualização como a computação em nuvem (ARMBRUST et al., 2010), é preciso incorporar à primeira uma visão consolidada dos recursos de infraestrutura disponíveis, dispensando o operador da rede de realizar um mapeamento direto entre os recursos existentes e o serviço desejado.

Os passos para a obtenção de tal característica é o objeto de análise da próxima seção.

3.2.2 Base de informações da topologia física

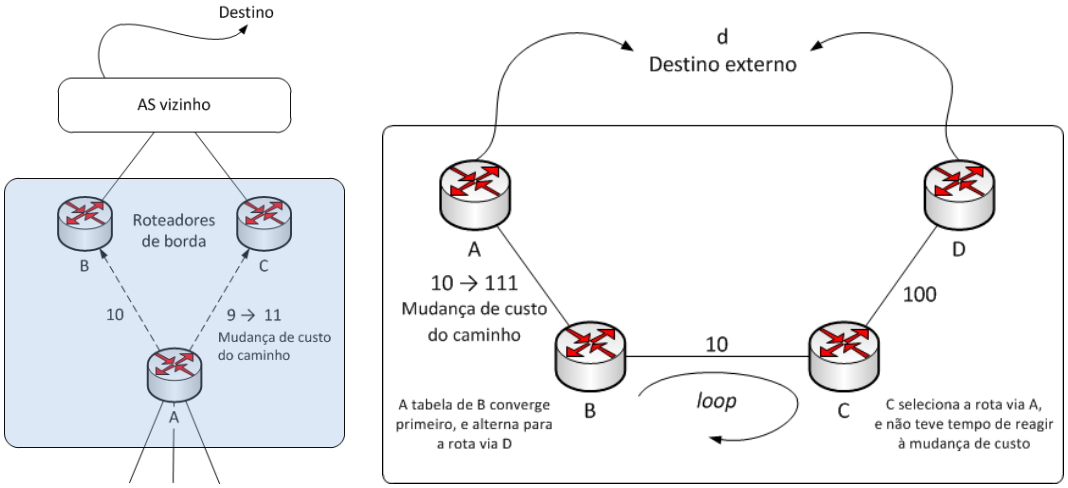
A operação de um sistema autônomo exige quase sempre a combinação de dois protocolos de roteamento: o BGP, para informes de alcançabilidade entre redes, e um IGP como o OSPF para tratar a alcançabilidade entre nós. O esforço de administração de ambos os protocolos é equivalente. Pode-se dizer porém que o papel do IGP é complementar, só afetando o roteamento em duas situações. Primeiro, porque uma rota para um prefixo

não pode ser adicionada à tabela de rotas local se não for possível alcançá-la a partir das informações do IGP. Depois, quando duas rotas se equivalem sob todas as outras métricas de distância e configurações de prioridade do BGP. Neste caso é escolhida a rota alcançável pelo caminho de menor custo segundo o IGP. Este comportamento é denominado roteamento *hot-potato* (TEIXEIRA et al., 2004).

Pode-se dizer que o *hot-potato* materializa, na falta de outro critério, o princípio da conservação de recursos e a própria filosofia de entrega por melhor esforço da Internet. Mas, esta noção de que "um único tamanho serve a todos" tem um custo elevado. Avaliações dos impactos negativos que o *hot-potato* pode causar à engenharia de tráfego em ASs são feitas em Teixeira et al. (2004) e Teixeira, Agarwal e Rexford (2005):

- Quando há dois caminhos de custo IGP muito próximo e que levam a roteadores de borda distintos, pequenas reconvergências do protocolo interno podem levar à mudança da rota BGP escolhida. Como no exemplo representado pela Figura 8a, onde uma pequena oscilação no custo do caminho entre A e C (de 9 para 11, no exemplo) poderá fazer com que um grande número de prefixos encaminhados através do AS 8000 passem a seguir a rota via B. É possível que isto se traduza em um congestionamento - ou viole uma política de negócio.

- Outra situação problemática pode ocorrer conforme representado pela Figura 8b, quando dois roteadores distintos têm uma sessão iBGP estabelecida entre si e um ao outro como parte de dois caminhos de custo aproximado. Caso o custo do caminho entre B e A seja alterado para 111 e B alterne para D como saída para um prefixo externo, é possível que B comece a encaminhar tráfego para C antes que sua tabela de rotas tenha reagido à mudança de custo do IGP. Neste caso C remeteria os pacotes de volta a B, pois ainda considera que o prefixo deva ser alcançado por intermédio de A, e um $loop$ de roteamento estaria formado. Os autores reportam em Teixeira et al. (2004) a possibilidade de $loops$ deste tipo perdurarem por 60 segundos ou até mais, levando à expiração de tráfego em trânsito e possíveis



(a) Mudança de roteamento *hot-potato*

(b) *Loop* temporário causado por mudança de custo do caminho

Figura 8: Impactos negativos da execução de um IGP sobre o roteamento inter-AS (TEIXEIRA et al., 2004)

congestionamentos pelo intenso reenvio de pacotes entre *B* e *C*.

Teixeira et al. (2004) relata que tais situações podem ser evitadas a partir de uma série de recomendações, que Teixeira, Agarwal e Rexford (2005) ratifica. São elas: 1) cada roteador no AS deve contar com um único roteador de borda alcançável por caminhos de baixo custo, e para o qual deve enviar a maior parte dos prefixos que encaminha; 2) deve haver dois ou mais caminhos de menor custo por onde o roteador de borda seja alcançável, o que evita que pequenas mudanças na topologia modifiquem o roteador de borda escolhido, ainda em uma eventual reconvergência do IGP. Por outro lado, realizar tais estratégias envolve um ajuste fino dos parâmetros de roteamento intra-AS e, possivelmente, a aplicação de ferramentas de modelagem de tráfego que levem em consideração os volumes de tráfego envolvidos, o que pode significar grande esforço operacional.

Uma visão diametralmente oposta à dos trabalhos comentados acima é documentada em Caesar et al. (2010). Os autores elencam uma série de ineficiências decorrentes de se imputar ao roteamento a tarefa de responder a falhas, disparando um processo de convergência quando a melhor rota atual se torna indisponível ou tem o custo aumentado. Entre as ineficiências elencadas destaca-se que a desconexão e reconexão seguidas de um canal, mais o atraso inerente ao processo distribuído de convergência, pode fazer com

que uma “melhor rota” só se propague completamente por um AS quando já não pode mais ser considerada a melhor opção disponível. A partir desta visão sugere-se que outras abordagens à escolha de caminhos em uma rede poderiam ser tentadas, “separando o roteamento dos roteadores” (CAESAR et al., 2010) pela centralização do cálculo de rotas em plataformas como a RCP (CAESAR et al., 2005), que então simplesmente propagariam as rotas adequadas a cada dispositivo.

Constatam-se semelhanças entre a visão de Caesar et al. (2010) e a plataforma de roteamento como serviço proposta.

Primeiro, porque a solução vislumbrada executa o roteamento sob uma supervisão única. Depois, porque conforme detalhado na Seção 3.2.1, espera-se que seja possível complementar a lógica de encaminhamento através de serviços, que seriam independentes dos processos de roteamento em si. Por fim, porque se coloca aqui o requisito de que o operador descreva a topologia física da rede a ser controlada pela plataforma. Tal descrição vem permitir que sejam implementados serviços que se baseiem diretamente nas conexões entre dispositivos para determinar a alcançabilidade entre eles, possivelmente minimizando ou até eliminando o uso de um IGP (CAESAR et al., 2010).

A descrição topológica também pode aproximar mais a plataforma proposta da abstração de nuvem computacional. Uma solução de IaaS pode ser capaz de provisionar um disco virtual a partir de um dispositivo *storage* sem que seu operador designe uma localização física específica para tal. Da mesma forma, a implementação de um serviço pode incluir a lógica necessária para calcular o caminho mais adequado para transmissão de determinado tipo de tráfego, onde a conectividade entre os equipamentos controlados certamente seria uma entrada importante. Um serviço também poderia determinar a viabilidade de desempenhar sua função a partir da topologia informada: não seria possível realizar balanceamento de carga em uma topologia que não contasse com uma porta de conexão com cada AS através do qual se desejasse enviar tráfego, por exemplo.

Espera-se que seja possível prover tais informações por meio da mesma gramática de

configuração proposta na Seção 3.2.1. Para efeito de ilustração, suponha-se o seguinte trecho de código:

```
dpid 0x1 ports 2
dpid 0x2 ports 3
dpid 0x1 port 2 trunk dpid 0x2 port 1
```

Neste caso, duas primeiras linhas do exemplo informam que a infraestrutura a ser controlada conta com dois *datapaths*, de identificadores "0x1" e "0x2". Eles possuem duas e três portas, respectivamente. A terceira e última linha informa que a porta de número 2 no *datapath* "0x1" está conectada à porta 1 do *datapath* "0x2". Um serviço hipotético de roteamento executando sobre a plataforma QuagFlow poderia utilizar o conhecimento sobre o plano de dados para instruir o *datapath* "0x2" a transmitir pela porta 1 o tráfego destinado a prefixos alcançáveis por meio do *datapath* "0x1", sem o uso de qualquer protocolo.

Como parte do processo experimental de validação da hipótese científica desta pesquisa, a implementação de termos de descrição de uma topologia física será discutida e avaliada como parte do Capítulo 5.

Nesta seção foi descrita a segunda característica proposta para a transição da arquitetura QuagFlow em uma plataforma de roteamento como serviço. Espera-se que descrever a política de negócio e a topologia subjacente sejam ações suficientes para viabilizar a elaboração de lógicas de encaminhamento que transcendam a avaliação de métricas básicas como vetor de distância e estado de enlaces. E como conseqüência, que se reduza ou elimine a necessidade de configurar manualmente métricas de protocolo para obter um serviço de roteamento previsível.

Ainda assim, processos de roteamento virtualizados continuam a desempenhar um importante papel no plano de controle QuagFlow. Portanto, a próxima seção trata da escolha de ferramentas de virtualização que garantam sua execução de forma consistente.

3.2.3 Elementos do plano de controle

A infraestrutura virtual adotada pela arquitetura QuagFlow Nascimento et al. (2010) pode representar uma limitação para a construção de uma plataforma de roteamento como serviço. A seguir, é realizada uma reflexão a respeito destas condições.

A operação das interfaces TAP utilizadas para comunicação com as MVs é de natureza ponto-a-ponto. Para que seja possível copiar os pacotes transmitidos pelo plano de controle para o plano de dados, o processo QuagFlow-C precisa monitorar todos os dispositivos deste tipo mapeados para portas de *datapaths*. O uso de um mecanismo de acesso múltiplo, como as *bridges* do núcleo do Linux, poderia ser uma alternativa mais escalável. Seria necessário monitorar apenas um dispositivo (a próxima *bridge*) para receber os pacotes transmitidos por *n* dispositivos. Tal escalabilidade poderia representar uma vantagem em cenários com muitos *datapaths* e portas a serem controlados.

As TAPs são uma abstração construída a partir de uma API de comunicação entre processos. Portanto, "desconectar" uma interface significa liberar seu descritor de arquivo (*handle*), o que o inutiliza. Neste caso, pode ser difícil refletir no plano de controle uma desconexão ou rompimento de enlace ocorrido no plano de dados. Seria possível apenas deixar de monitorar pacotes transmitidos pela interface correspondente, mas isto significaria manter o dispositivo TAP em estado operacional na MV. O que poderia levar a uma inconsistência de suas tabelas de roteamento em relação à conectividade física.

Uma única TAP não pode, por definição, ter cada uma de suas "extremidades" em dois nós físicos distintos. Portanto, o QuagFlow não poderia suportar nativamente um cenário em que partes do mesmo plano de controle executassem em anfitriões distintos.

O uso do virtualizador QEMU pode influenciar negativamente a escalabilidade da plataforma, por este se tratar de uma ferramenta de emulação e virtualização completa. Conforme visto no Capítulo 2, as técnicas de paravirtualização e virtualização de *containers* podem ser opções mais eficientes, já que espera-se que grande parte do plano de controle execute o mesmo *software* (a suíte Quagga), sob o mesmo sistema operacional.

O QEMU também exige que interfaces de rede sejam instanciadas durante a inicialização da MV de que farão parte (PROJETO QEMU, 2012). Isto significa que uma MV precisa ser sempre criada com um número de interfaces maior ou igual ao número de portas de qualquer *datapath* que seja associado à mesma, o que pode não ser possível garantir.

Face ao exposto, o próximo capítulo relata um processo experimental de avaliação de diferentes ferramentas de virtualização e conectividade virtual, com o objetivo de identificar possíveis novos componentes para o plano de controle da plataforma em construção. Espera-se que o uso de tais componentes flexibilize o processo de construção de topologias virtuais e as torne mais simples e escaláveis.

4 Avaliação de componentes virtuais

Conforme a característica enunciada no capítulo anterior para os elementos do plano de controle de uma plataforma de roteamento como serviço, espera-se que estes não demandem do operador da rede esforço de gerenciamento além daquele necessário para realizar a política de negócio desejada. Ora, o plano de controle opera através de uma rede inteiramente virtualizada e, conforme apresentado nos capítulos 2 e 3, esta é construída a partir de virtualizadores e dispositivos de conectividade virtual. Portanto, uma investigação das ferramentas mais aderentes ao comportamento pretendido faz-se necessária.

À guisa de fundamentar-se em referências do estado da arte, estes estudos se iniciaram por uma revisão dos critérios para a escolha das tecnologias usadas nas arquiteturas virtuais de roteamento IP identificadas na literatura. Não foram encontradas, porém, análises que visassem justificar tal escolha, quer sob o ponto de vista funcional, quer de desempenho. Acredita-se que isto, de algum modo, tenha contribuído para limitar sua aplicação fora do âmbito experimental.

Na falta de trabalhos específicos, propostas mais gerais de requisitos para infraestruturas de rede virtualizadas foram pesquisadas e descritas a seguir.

Em Bhatia et al. (2008), uma série de requisitos são elencados para uma plataforma que possa executar múltiplas redes virtuais programáveis sobre uma infraestrutura física de rede compartilhada: velocidade, isolamento, flexibilidade, escalabilidade e baixo custo. Neste trabalho, duas soluções de virtualização de código aberto são adotadas e combinadas com um mecanismo de conectividade virtual desenvolvido pelos autores,

para ter em seguida seu comportamento comparado com o de outras soluções. Não leva em conta, porém, a separação entre os planos de controle e de dados das redes estudadas.

O estudo encontrado em (FERNANDES et al., 2010) procura definir primitivas fundamentais para a virtualização de redes - instanciar, deletar e monitorar elementos, além de migrá-los entre nós físicos e definir seus parâmetros de alocação de recursos. Enquanto todas as primitivas relatadas são claramente importantes para as redes virtuais em geral, o presente trabalho assume que, no âmbito das plataformas de roteamento como serviço, as primitivas de migração e monitoramento podem ser providas com sucesso por instrumentos complementares aos virtualizadores e à conectividade virtual, ou pela combinação de funcionalidades de ambos, conforme apresentado em Pisa et al. (2010).

Na próxima seção este levantamento embasa a concepção de requisitos mais claros para os elementos do plano de controle. Espera-se que, a partir de então, seja possível classificá-los quanto ao grau de aderência para seu uso em uma plataforma de roteamento como serviço.

4.1 Requisitos para componentes do plano de controle

A partir dos critérios gerais verificados para a construção de redes virtuais, este trabalho propõe os seguintes requisitos para o componente virtualizador de uma plataforma de roteamento como serviço:

- *Isolamento*. O virtualizador deve permitir segregar tanto recursos de *hardware* quanto a pilha de rede dos sistemas executando sob sua supervisão. De outra forma, não seria possível executar versões concorrentes de processos de roteamento.

- *Eficiência*. O virtualizador deve impor a menor sobrecarga possível, mantendo o máximo de recursos à disposição das MVs, que têm responsabilidade direta sobre o desempenho da plataforma. Isto sugere que soluções baseadas em virtualização completa têm menores chances de atender ao requisito. Essa técnica, por defini-

ção, exclui a possibilidade de compartilhamento de código e dados entre anfitrião e convidados.

- *Escalabilidade*. A relação entre o consumo de recursos em um sistema e o número de MVs executadas deve ser aproximadamente linear. Isto visa permitir o planejamento da capacidade do nó anfitrião de acordo com a elasticidade desejada para o número de nós no plano virtual. Um virtualizador pouco escalável pode se tornar um fator limitante para as topologias que podem ser representadas por uma plataforma.

- *Flexibilidade*. A ferramenta de virtualização deve suportar a vinculação de múltiplos adaptadores de rede virtuais a cada MV, independentemente da existência de um adaptador físico ao qual eles possam ser mapeados. Isto é necessário para que possam ser representados, no plano de controle, nós que contam com um número de interfaces de rede diferente daquele disponível no nó anfitrião. Idealmente, o virtualizador também deve suportar, dinamicamente, o acréscimo, modificação e supressão das características dos adaptadores de rede vinculados a uma MV. Isto pode ser necessário para que uma topologia virtual reflita tempestivamente mudanças na conectividade do plano de dados.

Da mesma forma, este trabalho propõe os seguintes requisitos para o componente de conectividade virtual:

- *Acessos múltiplos*. Deve ser possível conectar múltiplos adaptadores de rede virtuais por meio de um único dispositivo virtual de conectividade. Isto é essencial tanto para a fidelidade da reprodução da conectividade do plano de dados na topologia virtual, quanto para a eficiência das arquiteturas em si. Enquanto a abstração oferecida por um *switch* virtual permite interligar n MVs por meio de n instruções de configuração, alcançar o mesmo efeito por meio de conectividade ponto-a-ponto exigiria a configuração de $n*(n-1)$ interfaces virtuais.

- *Isolamento*. Deve ser possível executar simultâneamente múltiplas instâncias de dispositivos virtuais de conectividade, sem que estas, no entanto, compartilhem seu tráfego virtual de dados. De outro modo a infraestrutura virtual poderia não refletir fielmente a conectividade física, além de possivelmente trazer resultados inesperados do ponto de vista dos processos de roteamento executados.

- *Eficiência*. A troca de tráfego entre os dispositivos de rede virtuais deve ser a mais rápida possível. Além disso, um instrumento de conectividade virtual deve impor a menor sobrecarga (processamento, memória) possível, não trazendo impacto negativo à execução dos nós do plano virtual. Este aspecto é importante porque o objetivo final da execução de um protocolo de roteamento, sua convergência, é um processo sensível a tempo.

- *Escalabilidade*. A relação entre o consumo de recursos em um sistema e o número de conexões efetuadas entre as MVs deve ser aproximadamente linear, o que viabiliza o planejamento flexível da capacidade do nó anfitrião e concorre para a elasticidade da topologia representada pelo plano virtual.

- *Flexibilidade*. O mecanismo de conectividade deve oferecer primitivas que permitam a associação dinâmica de interfaces de rede virtuais aos dispositivos em execução, permitindo a rápida propagação de eventos do plano físico para o plano virtual. Supondo-se uma topologia virtual com n roteadores ligados a um *backbone*, cada qual por um adaptador de rede, desconectar um dos nós por meio da abstração de um *switch* virtual exige uma única instrução de configuração, quando realizar a mesma ação em uma topologia construída sobre conectividade ponto-a-ponto exige a desmobilização de $n-1$ dispositivos virtuais.

- *Extensibilidade*. Preferencialmente, deve ser possível estender a conectividade virtual fornecida pelo mecanismo de maneira a suportar um plano de controle distribuído. Tal extensibilidade poderia ser provida através do suporte a estruturas lógicas externas à plataforma virtual de roteamento IP, como VLANs ou túneis, que

seriam utilizados então para que os dados do plano de controle cruzassem transparentemente uma infraestrutura física.

A próxima seção relata avaliações qualitativas e quantitativas realizadas com diferentes componentes de redes virtuais do estado da arte com base nestes requisitos, e que levaram à modificação da especificação original da arquitetura QuagFlow.

4.2 Avaliação qualitativa

Colocados os requisitos para os componentes das plataformas de roteamento como serviço, foram inicialmente selecionadas, dentre os elementos em uso nas arquiteturas virtuais de roteamento IP estudadas, aqueles que melhor pudessem atendê-los.

Além do QEMU, usado na arquitetura QuagFlow, foram identificados na literatura como sendo usados para a função de virtualizador, o hipervisor Xen (EGI et al., 2007) e as ferramentas de virtualização baseada em *containers* Linux-VServer (BHATIA et al., 2008) e OpenVZ (WANG et al., 2008). O Linux-VServer, porém, não suporta plenamente a virtualização da pilha IP do sistema. Isto significa que os containers compartilham de estruturas de rede do anfitrião, como por exemplo sua tabela de rotas. Tal condição afeta o requisito de isolamento e limita severamente seu uso em uma arquitetura de serviços de roteamento IP.

Também foram considerados para avaliação virtualizadores que, segundo os estudos realizados, ainda não haviam sido aplicados àquelas arquiteturas: os hipervisores KVM (HABIB, 2008) e VMWare vSphere (LAVERICK, 2010), além do mecanismo de *containers* LXC .

O hipervisor KVM realiza apenas virtualização completa, assim como o QEMU, o que concorre contra o requisito de eficiência e possivelmente contra a escalabilidade e a flexibilidade das arquiteturas que venham a adotá-lo.

O VMWare vSphere é um produto comercial, de código fechado, o que de alguma

maneira limita as possibilidades de extensão e adaptação da ferramenta em direção a uma arquitetura especializada, além de, assim como o QEMU e o KVM, não suportar sequer um modo de paravirtualização.

O LXC é um virtualizador de *container* recentemente integrado ao núcleo do Linux que inclui o recurso de virtualização da pilha de rede, o que o qualifica, no aspecto de isolamento, a operar como um roteador virtual. O LXC também inclui suporte à suspensão e retomada da execução de um *container*, o que pode representar um facilitador para a implementação de um mecanismo de migração entre nós físicos.

Assim, sob o ponto de vista funcional reúnem características compatíveis com os requisitos propostos para a função de virtualizador as ferramentas Xen, OpenVZ e LXC.

Para a função de conectividade virtual, além do *framework* TUN/TAP, amplamente utilizado pelas propostas encontradas na literatura, foram identificados como possíveis candidatos as *bridges* Linux (BENVENUTI, 2005) e o *switch* baseado em *software* Open vSwitch (PFAFF et al., 2009).

O *framework* TUN/TAP, porém, aplica-se à criação de interfaces virtuais para comunicação ponto-a-ponto, o que contraria o requisito de acesso múltiplo e o desqualifica para efeito desta avaliação. Em contrapartida, as demais alternativas atendem aos aspectos funcionais dos requisitos de isolamento, flexibilidade e extensibilidade estabelecidos para este tipo de componente.

Na próxima seção, as ferramentas de virtualização e conectividade virtual até aqui consideradas adequadas ao estabelecimento de uma plataforma de serviços de roteamento IP são quantitativamente comparadas.

4.3 Avaliação quantitativa

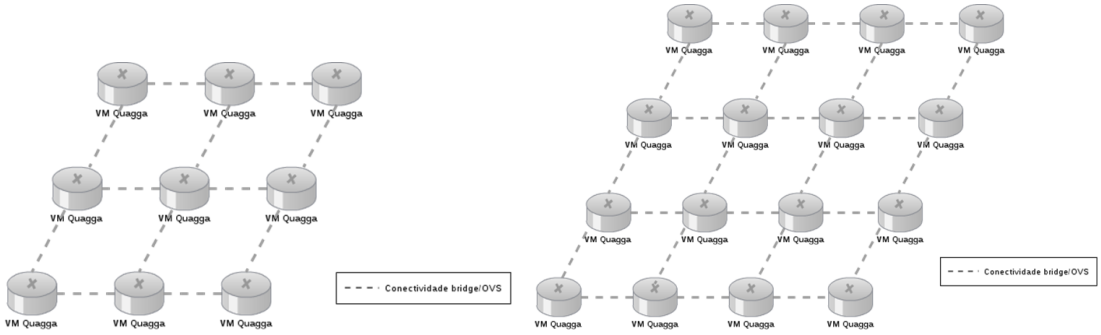
4.3.1 Plano de experimentação

Para realização da avaliação das ferramentas nos termos inicialmente propostos por este trabalho, elas foram alternadamente utilizadas para operar planos de controle virtuais. Estes representavam diferentes cenários de conectividade, a partir dos quais informações de desempenho puderam ser extraídas para as várias dimensões de requisitos previamente elencadas.

Os experimentos foram executados em um equipamento com uma CPU Intel Core 2 Duo 2.93GHz e 3GB de memória RAM. O computador executava o Debian GNU/Linux 5.0 e estava dedicado à execução dos experimentos. As versões dos sistemas de virtualização testadas foram o Xen 4.0 e as últimas versões estáveis do OpenVZ e do LXC, todas sobre o núcleo Linux 2.6.32.

Duas topologias de rede distintas foram testadas: uma rede com nós dispostos em uma grade 3x3, em seguida numa grade 4x4, executando processos de roteamento. Esta disposição foi adotada como uma simulação de crescimento tanto de nós como de conexões virtuais de rede, além do número destes elementos se aproximar com o encontrado em *backbones* de produção do mundo real (ALDERSON et al., 2005). De fato, os pontos de presença do *backbone* da Internet 2 que realizam roteamento são nove (INTERNET2 CONSORTIUM, 2012). Analogamente, a Rede Nacional de Ensino e Pesquisa brasileira soma 15 pontos de presença sendo atendidos por seu principal segmento, uma rede DWDM de 10Gbps (RNP, 2012).

Cada nó do plano de controle correspondia a um roteador virtual baseado na mesma versão de sistema operacional utilizada no anfitrião (Debian GNU/Linux 5.0), executando a suíte de roteamento Quagga, onde apenas o serviço OSPF se encontrava ativado. Com o intuito de conferir maior generalidade aos dados coletados, foram extraídas informações de desempenho para diferentes intervalos de anúncio (intervalo de *Hello*) do protocolo: 1, 5 e 10 segundos. Sempre que o intervalo de anúncio foi modificado, o intervalo de



(a) Topologia 3x3. (b) Topologia 4x4.

Figura 9: Visualização das topologias de rede utilizadas para experimentação.

detecção de enlace indisponível foi consistentemente ajustado, respectivamente para 4, 20 e 40 segundos.

A conectividade entre os roteadores virtuais também foi testada através de *bridges* Linux e do OVS. Dados de interesse que apoiassem a avaliação das ferramentas foram obtidos a partir da captura do tráfego passante nestes dispositivos de *software*.

Cada segmento de conectividade estabelecido entre dois nós contava com duas interfaces de rede virtuais endereçadas em uma sub-rede exclusiva, selecionada a partir do bloco privativo 10.0.0.0/8. Nenhuma das interfaces utilizadas pelo protocolo de roteamento possuía sumarização ativada (todas as sub-redes foram atribuídas à área 0 do protocolo OSPF).

Dentre todos os cenários possíveis, não foi avaliado o uso do virtualizador OpenVZ com conectividade OVS. Isto porque ambos eram incompatíveis até o momento do experimento, sem perspectiva recente de mudança deste aspecto.

Dados foram extraídos em diferentes etapas da operação de um plano de controle virtual, com o objetivo de compor diferentes métricas a partir das quais foi possível comparar os desempenhos relativos das ferramentas de virtualização testadas.

Para a etapa de inicialização das topologias em teste, um momento computacionalmente custoso em que normalmente as estruturas de dados e recursos de controle específicos de cada nó precisam ser alocados, as seguintes métricas e dados foram medidos:

- *Tempo de inicialização dos nós*. Trata-se do intervalo de tempo desde a solicitação de instanciamento dos nós de uma topologia virtual e sua plena operação. Dado o grande número de tarefas envolvidas no processo de carga de uma MV, esta medição oferece um termo de comparação de eficiência entre as ferramentas virtualizadoras.

- *Utilização de CPU na inicialização*. Permite inferir, em um momento de intensa demanda e através da comparação dos resultados obtidos através dos diferentes sistemas de virtualização, o grau de sobrecarga imposta por cada solução.

- *Utilização de memória na inicialização*. Também aplica-se à *eficiência* dos mecanismos de virtualização em teste durante a carga de uma MV. Neste caso, avalia-se sua sobrecarga espacial.

- *Tempo até a convergência inicial*. Medição do intervalo de tempo desde a inicialização da topologia até a convergência do protocolo OSPF, aqui definida como o momento em que mensagens de anúncio de enlaces não são mais trocadas entre os nós.

Com o intuito de caracterizar o consumo de recursos das ferramentas durante a operação regular, sem a ocorrência de eventos que pudessem desencadear troca de mensagens de controle OSPF, após dez minutos da inicialização de cada topologia virtual as seguintes métricas foram coletadas:

- *Utilização de CPU em operação regular*. Objetiva conhecer a sobrecarga de processamento imposta pelo uso das ferramentas aplicadas a cada um dos cenários testados quando a topologia encontra-se plenamente operacional.

- *Utilização de memória em operação regular*. Também busca conhecer a sobrecarga espacial imposta pelo uso das ferramentas de virtualização aplicadas a cada um dos cenários durante operação plena e livre de falhas.

Por fim, para caracterizar a eficiência dos componentes de virtualização experimentados durante a representação de eventos ocorridos no plano de dados, que motivam a

troca e o processamento de mensagens de anúncio de enlaces do protocolo OSPF, foram introduzidas falhas simultâneas nos enlaces das topologias virtuais de forma a reduzir o grafo de conectividade de cada uma delas a uma árvore geradora mínima. Os mesmos enlaces foram reintegrados à topologia 120 segundos após sua desconexão, de maneira que categorias de eventos distintos foram observadas nos experimentos.

Na introdução de tais falhas e durante as atividades de resposta das topologias em execução, foram avaliadas as seguintes métricas:

- *Utilização de CPU durante evento*. O consumo de recursos de processamento foi mensurado, para cada etapa de falha aplicada.
- *Utilização de memória durante evento*. Durante as etapas de falha, o uso de memória principal do sistema também foi medido.
- *Convergência pós-evento*. Avaliação da evolução do tráfego OSPF desde a introdução de uma falha até a conclusão da troca de mensagens de anúncio do estado de enlaces do protocolo.

Não obstante a afinidade exclusiva das métricas propostas com o requisito de *eficiência* estabelecido na seção anterior, a realização de medidas para duas escalas distintas de topologia - com nove e dezesseis nós - permite a averiguação de linearidade na evolução da demanda por recursos entre os dois cenários e, portanto, a análise do grau de escalabilidade das soluções em teste.

Com o objetivo de conferir relevância estatística aos resultados obtidos, cada uma das diferentes métricas teve seus dados coletados em 30 execuções distintas de cada topologia de rede para cada um dos cenários possíveis (combinação de ferramentas). Isto permitiu uma avaliação dos resultados em uma escala de confiança de 90% para todos os indicadores estabelecidos.

Tabela 1: Tempo Médio para Inicialização das Redes Virtuais

Virtualizador	Conectividade	Topologia	Tempo para Inicialização (s)
LXC	Bridge	3x3	31
		4x4	62
	OpenVSwitch	3x3	32
		4x4	63
OpenVZ	Bridge	3x3	39
		4x4	74
Xen	Qualquer	3x3	43
		4x4	78

4.3.2 Avaliação dos resultados

A Tabela 1 apresenta os resultados para o tempo de inicialização das topologias. O LXC é o mais rápido com a conectividade baseada em *bridges*, realizando a tarefa em um tempo em média 20% menor que o OpenVZ, no caso das grades 3x3. Para grades 4x4, a diferença se mantêm em 16%. Ambas as relações se mantém respectivamente em 18 e 15% ao comparar MVs LXC conectadas via OVS. O Xen é o mais lento, com tempos médios de inicialização de 43 e 78s, para as topologias 3x3 e 4x4, respectivamente, sob qualquer conectividade virtual.

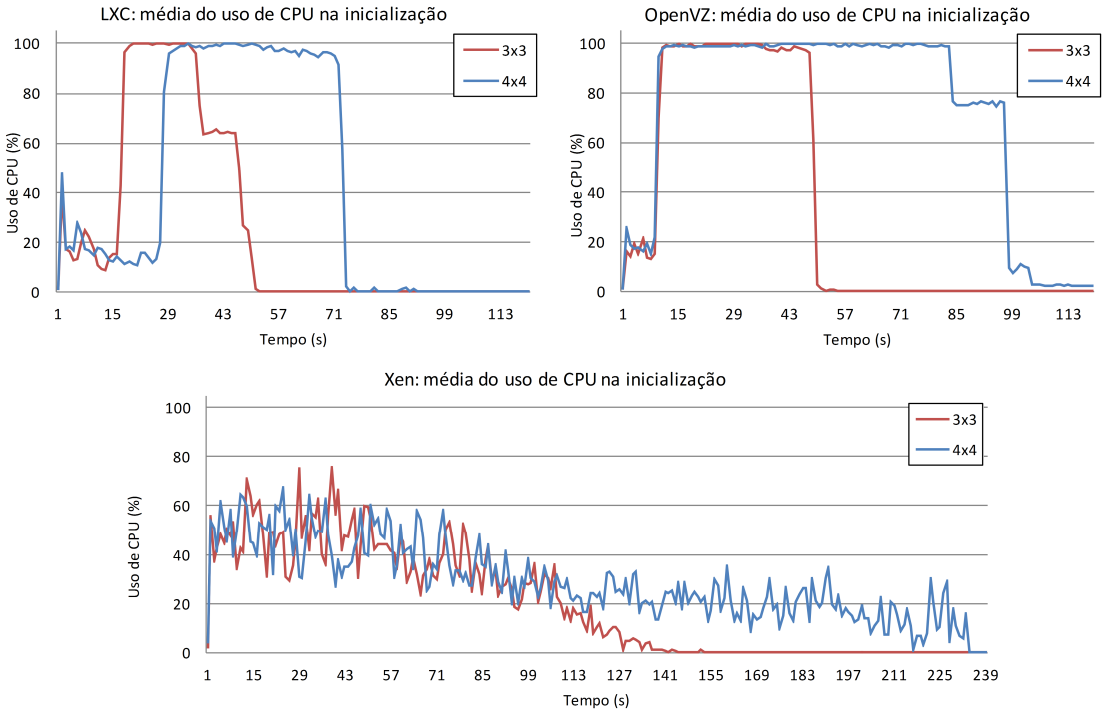


Figura 10: Consumo de CPU durante a inicialização das topologias virtuais.

Tabela 2: Utilização Média de Memória ao Longo da Operação das Redes Virtuais

Virtual.	Conec.	Topol.	Uso de RAM (MB), durante:		
			Inicialização	Op. Regular	Evento Falha
LXC	Bridge	3x3	169	180	206
		4x4	259,50	276	314
	OpenVSwitch	3x3	175	185	228
		4x4	269,75	286	369
OpenVZ	Bridge	3x3	180	226	268
		4x4	337,25	350,5	693
Xen	Qualquer	3x3	1152	1152	1152
		4x4	2048	2048	2048

A Figura 10 traz o consumo médio de CPU durante a inicialização das MVs. Conforme é possível visualizar, tanto o LXC quanto o OpenVZ levam a um consumo de 100% durante o processo de inicialização. Não obstante, a ocupação da CPU no LXC e no OpenVZ decaem aos valores de operação regular em menos de 50% do tempo verificado com o Xen. Todas as médias de consumo para o Xen e o LXC se mantêm dentro do intervalo de confiança daquelas apresentadas na Figura 10 quando a conectividade OVS é utilizada, não estando tais resultados, portanto, representados a seguir.

A Tabela 2 apresenta o consumo médio de memória das ferramentas avaliadas, para todos os cenários. A partir da média de uso da RAM durante a inicialização das MVs é possível constatar que as ferramentas LXC e OpenVZ apresentaram evolução de consumo aproximadamente linear com o aumento da escala da topologia, com vantagem significativa para a demanda do LXC, que é menor. O uso do OVS no caso do LXC representa uma demanda adicional de apenas 2,8 a 4% de RAM. No caso do virtualizador Xen, uma quantidade fixa de memória deve ser reservada para cada MV no momento de sua criação (neste cenário 128MB foram reservados). Assim, o uso de RAM variou muito pouco ao longo de seus testes (menos que 5%).

As medições do tempo para convergência inicial do protocolo OSPF apresentaram resultado diverso daquele apontado pelo tempo de inicialização das topologias virtuais. Na Figura 11 estão representados os tempos de convergência inicial para o LXC e o OpenVZ, que obtiveram o melhor desempenho, e também para o Xen.

Há atraso entre o instanciamento da rede virtual LXC e o início da troca de dados OSPF. Mesmo operando a grade mais rápida, carregada em 31s, essa topologia só troca anúncios OSPF 50s após o início do experimento. Seu desempenho é superado pelo do OpenVZ, cujas MVs iniciam a comunicação OSPF imediatamente. Os resultados da ferramenta Xen acompanham seu tempo de carregamento, com troca de anúncios só após 130s do ínicio das topologias virtuais, sob qualquer conectividade.

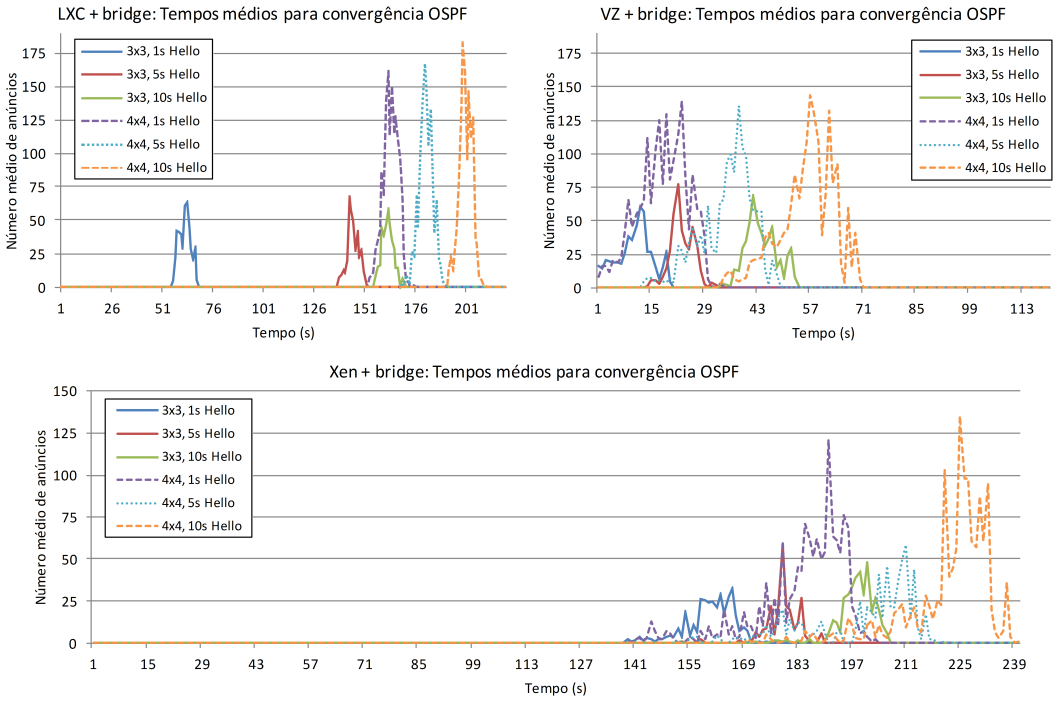


Figura 11: Variação da convergência OSPF inicial com LXC, OpenVZ e Xen usando conectividade via bridge

Após a sobrecarga inicial, todas as ferramentas de virtualização apresentam consumo de CPU médio em torno de 1% durante a operação de um plano de controle onde não estão acontecendo desconexões. O consumo médio de memória é ampliado em 6,4% no LXC, enquanto o OpenVZ apresenta aumento médio de 4%.

Por fim, aplicados os testes de convergência após eventos de enlaces, foi possível verificar que seus resultados não são afetados por atrasos verificados no início da operação das redes virtuais, apesar do desempenho do Xen ainda ficar aquém das demais ferramentas, sob qualquer conectividade utilizada.

A Figura 12 detalha os tempos médios de convergência após eventos de enlaces para

as combinações de melhor desempenho. Tanto a convergência do LXC + OVS quanto a do OpenVZ + *bridges* se aproximam do valor ótimo do OSPF (o quádruplo do tempo de Hello), com leve vantagem para o LXC.

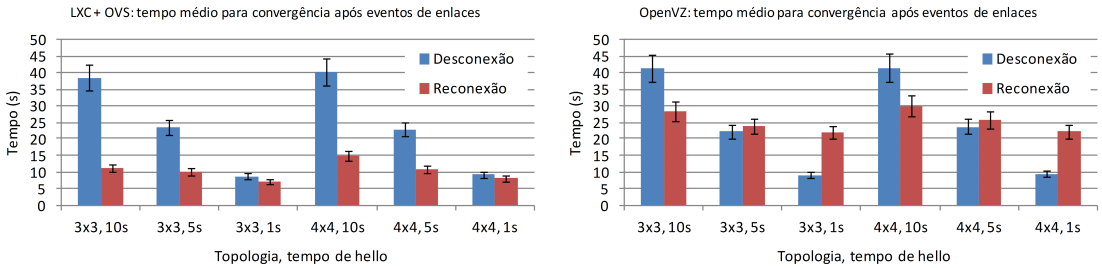


Figura 12: Convergência LXC + OVS e OpenVZ + bridges após eventos de enlaces.

O consumo de CPU não alcançou mais que 3% durante a realização dos testes com eventos de enlaces. Conforme pode ser visto na Tabela 2, o consumo de memória cresceu no período. No caso do LXC este crescimento foi aproximadamente linear para os cenários experimentados. Na topologia 4x4 OpenVZ, porém, o uso de RAM se acentua na ocorrência de falhas, o que pode indicar um problema de escalabilidade na ferramenta. Na ausência de novos eventos, o consumo de RAM observado retornou aos valores médios de operação.

4.4 Parecer final

As medições realizadas sugerem que as ferramentas de virtualização baseadas em *containers* reúnem as características mais adequadas à execução de um plano de controle virtual, em oposição à paravirtualização utilizada via Xen. Enquanto os *containers* permitem um uso mais flexível e escalável da memória, o Xen se baseia em uma alocação fixa.

A virtualização oferecida pela ferramenta LXC exige mais tempo que aquela baseada no OpenVZ para que uma topologia inicie e o protocolo de roteamento OSPF convirja. Ainda assim, a convergência ao longo da operação regular é mais veloz na combinação LXC + OVS. Esta conjunção de ferramentas também apresentou consumo de recursos

linear nos experimentos realizados, em oposição ao percebido no OpenVZ.

O OVS apresenta características de flexibilidade importantes para a função pretendida. Além de realizar conectividade virtual com acesso compartilhado, esta ferramenta implementa *datapaths* OpenFlow baseados em *software*. Portanto, seu uso implica que a topologia do plano de controle também seja programável, o que pode resultar em maior eficiência na operação do QuagFlow-C. Outro recurso interessante do OVS é a capacidade de se estender um *datapath* por diferentes sistemas anfitriões, uma possível oportunidade para a execução de um plano de controle distribuído.

Da mesma forma, o virtualizador LXC inclui funcionalidades que, combinadas ao desempenho apresentado, o apontam como uma opção promissora para a construção de redes virtuais. Ao contrário do QEMU, utilizado na plataforma QuagFlow no momento da realização desta avaliação, o LXC suporta dispositivos de conectividade virtuais mais flexíveis que os TAPs, além do próprio Open vSwitch. Em verdade, as interfaces virtuais LXC são elementos completamente independentes do virtualizador, disponibilizados pelo núcleo do sistema operacional Linux. Assim, apesar de também implementar uma abstração de natureza ponto-a-ponto, tais dispositivos podem ser vinculados dinamicamente a *datapaths* OVS, instanciados e atribuídos a uma MV antes ou depois de seu carregamento e conectar diretamente duas MVs.

Face ao exposto, os resultados da experimentação realizada, aliados aos aspectos de flexibilidade e extensibilidade do LXC e do OVS, permitem concluir que ambas são as ferramentas mais adequadas ao plano de controle de uma plataforma de roteamento como serviço.

5 Construção da plataforma de roteamento como serviço

O trabalho de experimentação realizado no capítulo anterior visava identificar, dentre as soluções de virtualização do estado da arte, aquelas que melhor se aplicassem ao plano de controle da arquitetura QuagFlow[1]. Em face dos resultados obtidos, os desenvolvedores do RouteFlow empreenderam mudanças na arquitetura original para que suportasse o uso do virtualizador LXC e da ferramenta de conectividade virtual Open vSwitch.

Tais mudanças, além de trazer impactos positivos sob os aspectos de flexibilidade e escalabilidade, modificam a operação da arquitetura em relação ao que foi apresentado na Seção 3. Por isto, este capítulo inicia-se com um pequeno relato do novo comportamento produzido. O avanço da infraestrutura virtual, porém, representa apenas uma das três características necessárias a uma plataforma de roteamento como serviço. O restante do texto se concentra na materialização dos dois outros requisitos.

5.1 A nova arquitetura RouteFlow

É possível visualizar a nova organização do plano de controle RouteFlow na Figura 13. A conectividade baseada no Open vSwitch permite estabelecer um segmento de rede exclusivo, identificado como "OVS de gerenciamento". Assim, fica disponível um meio de acesso compartilhado através do qual os agentes de coleta da FIB presentes em cada MV podem se comunicar com os processos de controle executando no anfitrião.

[1] Aqui, uma ressalva: a partir do trabalho publicado em Nascimento et al. (2011a), o projeto QuagFlow passou a adotar o nome RouteFlow. Assim, para consistência tanto cronológica quanto de referências bibliográficas, deste ponto em diante o texto sempre fará referência ao nome RouteFlow.

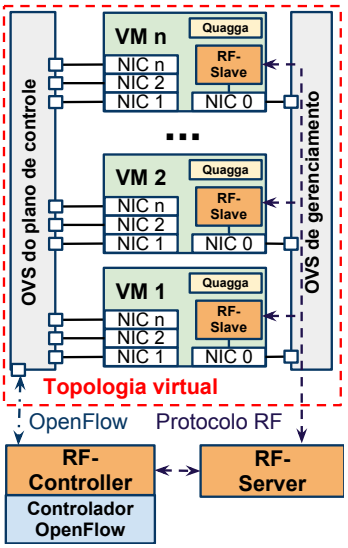


Figura 13: Visão esquemática da arquitetura RouteFlow

Isto se opõe ao cenário com TAPs, na medida em que era necessário monitorar um canal de controle distinto para cada agente.

A ferramenta Open vSwitch também substitui a conectividade TAP no caso das interfaces de MVs mapeadas para portas de *datapaths*. Estas interfaces agora são vinculadas ao "OVS do plano de controle". Portanto, um meio de acesso compartilhado também está disponível para representação da conectividade física no plano de controle. E uma vez que a própria ferramenta OVS implementa um *datapath* OpenFlow, o tráfego recebido e transmitido por cada MV também está sujeito a monitoramento e análise, mas, sem a necessidade de se gerenciar e monitorar um canal de comunicação distinto para cada dispositivo de rede virtual.

A própria lógica de gerenciamento da arquitetura foi modularizada. Um novo componente, RF-Server, foi construído para receber as mensagens de agentes RF-Slave, analisá-las e traduzi-las em comandos, enviados para o mecanismo de controle RF-Controller. O RF-Controller passa a ter a função exclusiva de se comunicar com *datapaths*, manipulando entradas de suas tabelas de encaminhamento.

A inicialização de um cenário de controle se mantém praticamente inalterada. A cada *datapath* conectado através do RF-Controller, o RF-Server associa uma MV. As portas

do *datapath* são mapeadas para as interfaces de rede da MV, e entradas permanentes são criadas em sua tabela de encaminhamento. Assim, como visto no Capítulo 3, tais entradas visam o tratamento de pacotes de roteamento e ARP, além da identificação de tráfego que ainda não possa ser encaminhado diretamente no plano de dados.

Pedidos de transmissão dos planos de dados e de controle são tratados de maneira mais homogênea. Ao receber um pacote para encaminhamento, o RF-Controller o repassa ao RF-Server para análise. Caso a transmissão tenha origem em uma porta do plano de dados, o pacote será entregue na porta correspondente do OVS do plano de controle. Caso ela se origine no plano de controle, será transmitida pela porta de *datapath* correspondente no plano de dados.

Uma vez que a troca de pacotes entre os planos de controle e de dados está garantida, também estão as condições para a ocorrência de resoluções ARP e convergência de protocolos de roteamento. Isto assegura a atualização da FIB das MVs e, portanto, a coleta de tais alterações a partir dos agentes RF-Slave. E ao receber dos RF-Slaves as mensagens contendo modificações de FIB, o RF-Server instrui o RF-Controller a instalar as entradas de encaminhamento adequadas no plano de dados. Garante-se assim que o sistema se mantém produzindo resultados semelhantes àqueles relatados na Seção 3.1.

Em complemento à definição deste novo plano de controle, a incorporação das demais características de uma plataforma de roteamento como serviço foi realizada no RouteFlow. Tal processo é detalhado a seguir.

5.2 A gramática de configuração RouteFlow

No Capítulo 3 foram colocadas como necessárias à obtenção de uma plataforma de roteamento como serviço três características, sendo duas delas: *abstração única da política de roteamento* e *base de informações da topologia física*. Apesar de diferentes, ambas têm funções complementares. A primeira tem a função de instruir o plano de controle a respeito de como devem ser "pós-processadas" as decisões de roteamento produzidas pe-

los protocolos correspondentes. Ela serve para determinar o comportamento esperado da operação de encaminhamento realizada em um AS, e que deve atender à política de negócio vigente. Em uma simplificação, pode-se dizer que tal abstração é o próprio conjunto dos serviços de roteamento a serem executados na plataforma. A base de informações, por sua vez, deve oferecer ao(s) serviço(s) em execução uma descrição precisa e consistente da infraestrutura subjacente. Tal descrição, acredita-se, pode suplantar a necessidade de protocolos IGP, que oferecem uma *aproximação* dos recursos de comunicação disponíveis traduzida em um *custo* de transporte.

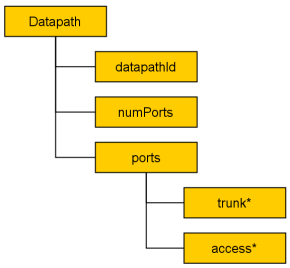
A abordagem escolhida para incorporar ambas as características ao RouteFlow foi a construção de uma gramática de configuração. Portanto, a referência para definição de seus elementos foram as entidades lógicas existentes em uma infraestrutura SDN. Espera-se que tais elementos sejam suficientes para descrever um conjunto de serviços de roteamento e sua infraestrutura no contexto de um AS cujo plano de controle é inteiramente baseado na plataforma RouteFlow.

A seguir, tais conceitos, a elaboração e a implementação da gramática serão apresentados em detalhes.

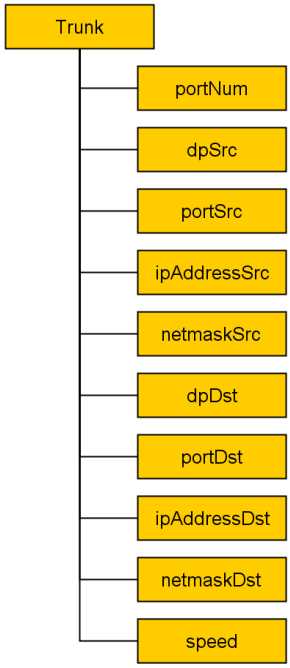
5.2.1 Representação de elementos

De acordo com os objetivos propostos, do ponto de vista do plano de dados a gramática limita-se à representação de *datapaths*, suas portas de comunicação e suas funções, definidas como duas para efeito desta pesquisa: a ligação entre dois *datapaths* e a conexão do *datapath* a outros tipos de dispositivo. Na perspectiva de controle é possível descrever os sistemas anfitriões que executam a topologia virtual e os serviços que se deseja incorporar ao roteamento, bem como seus parâmetros.

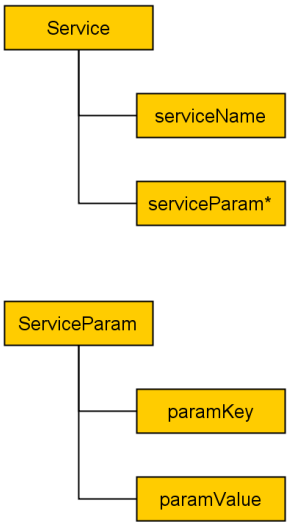
Aqui, uma ressalva: ainda que alguns dos casos previstos não se apliquem neste momento ao RouteFlow (como a possibilidade de se distribuir a topologia virtual por diferentes anfitriões), sua inclusão visa conferir maior generalidade à solução construída.



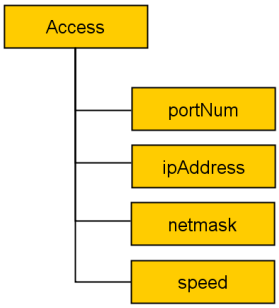
(a) Modelo Datapath.



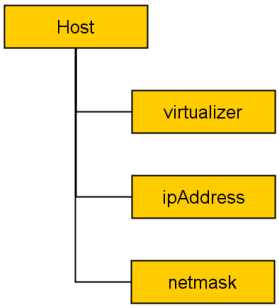
(b) Modelo Trunk.



(c) Modelo Service.



(d) Modelo Access.



(e) Modelo Host.

Figura 14: Modelos de dados utilizados para representar os elementos do plano de controle RouteFlow.

A Figura 14 apresenta o conjunto de elementos considerados para a construção da gramática, bem como seus atributos. Nas próximas seções está disponível uma descrição dos modelos ali representados.

5.2.1.1 Datapath

O modelo exibido na Figura 14a permite representar os dispositivos responsáveis pelo encaminhamento no plano de dados, os *datapaths*. Seu atributo "datapathid" permite armazenar a característica que distingue um elemento do tipo dentre todos os demais: seu identificador OpenFlow, uma valor hexadecimal de dezesseis dígitos. Também é possível informar o número de portas de comunicação disponíveis, através do atributo "numPorts". Esta informação pode ser dispensável em um *datapath* baseado em *software* como o Open vSwitch, já que seu número de "portas de comunicação" depende da quantidade de interfaces virtuais vinculadas, mas pode ser relevante no contexto de um dispositivo físico.

Um *datapath* possui, afinal, um conjunto de portas associadas, de diferentes tipos. Portas de tronco permitem interligar dois *datapaths* dentro do mesmo AS. Portas de acesso podem ter a função de interligar o dispositivo a um AS externo, a um roteador legado ou a um sistema final. Isto levou à definição de dois modelos distintos para representá-las.

5.2.1.2 Trunk

A Figura 14b apresenta o modelo de representação de uma porta de comunicação usada para interligar um *datapath* a outros aparelhos deste tipo, dentro do mesmo AS. A porta em questão é identificada pelo atributo "portNum" (número de porta). A outra extremidade do enlace, por sua vez, está designada nos campos "dpDst" e "portDst", análogos aos dois primeiros.

O endereço IP e a máscara de rede utilizados em uma porta do tipo *trunk* são especificados através dos atributos "ipAddress" e "netmask". Atualmente, a plataforma RouteFlow não é capaz de manipular diretamente as configurações de endereçamento das interfaces de rede do plano de controle. Ainda assim, ambos os dados podem ser relevantes para um serviço arbitrário.

O modelo também inclui uma designação de velocidade de comunicação no atributo

speed. Imagina-se que tal informação possa ser útil para o gerenciamento de capacidade dos serviços executando sobre a plataforma, operando de maneira análoga à noção de custo em um IGP.

5.2.1.3 Access

As portas de comunicação não utilizadas para conexão entre *datapaths* são consideradas portas de acesso, representadas pelo modelo da Figura 14d. Estas portas são identificadas por número (atributo "portNum") e identificador de *datapath*, como no caso de portas de tronco. Ainda de maneira análoga ao modelo "Trunk", também estão previstos os atributos de endereço IP e máscara de rede para estas conexões, além de um campo para especificação de velocidade de transmissão.

5.2.1.4 Host

O modelo "Host", visto na Figura 14e, provê uma representação dos possíveis sistemas anfitriões através dos quais a topologia virtual utilizada pela plataforma RouteFlow esteja distribuída.

O campo "virtualizer" denota a API de virtualização utilizada para executar MVs em um dado sistema. Ainda que o foco atual da plataforma seja a ferramenta LXC, podem haver ambientes legados executando MVs baseadas no QEMU. E uma vez que uma das opções disponíveis para manipular um virtualizador LXC remoto dá-se através da ferramenta libvirt (BOLTE et al., 2010), o atributo "virtualizer" pode ser usado para sinalizar esta condição. Por fim, o atributo "ipAddress" define o endereço para comunicação com o anfitrião.

É verdade que atualmente o RouteFlow não incorpora as funcionalidades necessárias para realizar a distribuição de seu plano de controle por diferentes anfitriões, e que obtê-las está fora do escopo da presente pesquisa. Ainda assim, considerando-se possíveis ganhos de escalabilidade, resiliência e flexibilidade com a incorporação deste recurso, é possível que a informação provida por este modelo seja relevante no futuro. Acredita-se,

portanto, que sua inclusão aprimore a generalidade da gramática.

5.2.1.5 Service e ServiceParam

Representado na Figura 14c, o modelo “Service” incorpora as informações pertinentes a um serviço de roteamento da plataforma RouteFlow. Sua função é identificar cada serviço por seu respectivo nome (o atributo “serviceName”), e estabelecer uma relação $1 : n$ com os parâmetros de operação especificados para o mesmo. Tais parâmetros estão incorporados ao modelo “ServiceParam”, também mostrado na Figura 14c.

O modelo “ServiceParam” oferece uma estrutura de chaves (atributo “ParamKey”) e valores (“ParamValue”) para informação de um serviço a respeito de seu comportamento esperado. Não espera-se um tipo ou formato específico para os conteúdos atribuídos a cada campo. Pelo contrário, é desejável que tais sejam tão genéricos e flexíveis quanto possível em sua implementação. Por exemplo, o conteúdo de um campo “ParamValue” poderia ser fornecido como um texto contendo múltiplos identificadores de *datapath*, ficando a critério do serviço correspondente interpretá-lo.

Através da flexibilidade vislumbrada para os modelos “Service” e “ServiceParam”, busca-se reduzir o acoplamento entre o código da plataforma RouteFlow e o dos serviços executando sob sua supervisão.

5.2.2 Atendimento a proposições

A partir dos modelos construídos para representar a conectividade de um AS arbitrário controlado pela plataforma RouteFlow, é possível derivar a gramática pretendida neste trabalho. Neste processo, porém, deve-se manter observância às proposições elencadas na Seção 3.2.1.1 do texto.

A seguir, a evolução da gramática RouteFlow é documentada, ao passo que se relaciona a ela os aspectos pertinentes da mesma às condições estabelecidas.

```
1. Grammar → Production*
2. Production → ( Host | DatapathConf | Service )
3. Host → "host" IPAddress Virtualizer
4. DatapathConf → Datapath ( NumPortsDecl | AccessDecl | TrunkDecl )
5. Service → "service" ServiceName ParamKey ":" ParamValue+
6. Virtualizer → ( "lxc" )
7. Datapath → "dpid" DatapathId
8. NumPortsDecl → "ports" NumPorts
9. AccessDecl → PortDecl "access" IPAddress* SpeedDecl
10. TrunkDecl → PortDecl IPAddress* "trunk" Datapath PortDecl IPAddress* SpeedDecl*
11. ServiceName → string
12. ParamKey → string
13. ParamValue → string
14. DatapathId → ("0x" [0-9a-f]+)
15. NumPorts → integer
16. PortDecl → "port" PortId
17. PortId → integer
18. SpeedDecl → “speed” Speed
19. Speed → integer
```

Figura 15: Gramática genérica para a configuração RouteFlow.

5.2.2.1 Expressividade

Na Figura 15 encontra-se a formalização da gramática de configuração construída a partir dos modelos de dados da Seção 5.2.1, transcrita em um dialeto EBNF (AHO; SETHI; ULLMAN, 1986). Por uma questão de clareza, a definição de símbolos terminais aproxima-se daquela definida em W3C CONSORTIUM (2012). Assim, a interpretação de símbolos terminais com regras de formação complexas – como endereços IP e as respectivas máscaras de rede – foram omitidas. Neste caso, foi considerada a existência de um símbolo terminal *IPAddress*, que encapsula ambas as informações. Não obstante, encontra-se disponível no Anexo A deste documento a listagem do arquivo *lex* usado para implementação, onde todos os símbolos terminais complexos foram apropriadamente definidos por meio de expressões regulares.

A produção da gramática se dá a partir de múltiplos elementos “Production”, que podem corresponder aos não-terminais “Host”, “DatapathConf” ou “Service”. Note-se que nenhum deles é obrigatório. Portanto, também atende à regra “1” um texto de configuração sem qualquer conteúdo. Isto determina que a incorporação desta gramática ao RouteFlow não implique na obrigatoriedade de se utilizá-la: na ausência de especificações da topologia e de serviços a serem ativados, o comportamento “1:1” pode ser ativado.

O símbolo da regra “3” (“Host”) tem a função de descrever o(s) sistema(s) anfitrião(ões) disponível(is). Para seu uso, os dados que vão alimentar os atributos “ipAd-

dress", "netmask" e "virtualizer" do modelo "Host" devem ser necessariamente especificados.

Na regra "4", o símbolo "DatapathConf" é definido de forma a abranger as características previstas para *datapaths* e suas portas de comunicação nos modelos da Seção 5.2.1. Ele se deriva em uma expressão ("Datapath") necessariamente iniciada por "dpid 0x..." (terminal "DatapathId", regra "14"), que designa o elemento a ser descrito, mais os não-terminais "NumPortsDecl" (regra "8"), "AccessDecl" ("9") ou "TrunkDecl" ("10").

O não-terminal "NumPortsDecl" se decompõe em dois terminais, um deles numérico, para compor as informações do modelo "Datapath".

Já o símbolo "AccessDecl" permite designar uma porta de acesso, conforme o modelo "Access". A porta é identificada numericamente (pela avaliação do símbolo "PortDecl" na regras "16" e "17"), e pode-se atribuir a ela, em caráter opcional, dados de endereço IP (terminal "IPAddress") e velocidade de transmissão, definida através do símbolo "SpeedDecl" nas regras "18" e "19".

Através do símbolo "TrunkDecl", uma conexão entre *datapaths* pode ser especificada, de acordo com o modelo "Trunk". Ao identificador de *datapath* existente (especificado no início da expressão "DatapathConf" que é derivada em "TrunkDecl"), acrescenta-se um número de porta ("PortDecl") e opcionalmente seu endereço IP. O símbolo é composto ainda por uma nova especificação de *datapath* e porta (os atributos "dpDst" e "portDst"), obrigatórios. A expressão se conclui com um endereço IP opcional (utilizado para a porta de destino), mais a velocidade de transmissão do enlace ("SpeedDecl").

O símbolo da regra "5", "Service", corresponde aos modelos "Service" e "ServiceParam". Para cada serviço especificado através do símbolo terminal "ServiceName", espera-se ao menos um parâmetro (possivelmente aquele que ativa sua operação). Um parâmetro é composto por um símbolo-chave "ParamKey" que o identifica e por um ou mais terminais "ParamValue", que atribuem-lhe um valor.

Por fim, faz-se necessário elencar algumas premissas para a gramática descrita:

- Sabe-se que um identificador OpenFlow é composto por um valor hexadecimal de dezesseis dígitos. A representação desta informação, em formato textual ou numérico, fica a cargo da implementação em si;

- As declarações de número de portas de um *datapath*, portas de acesso e de tronco são independentes entre si. Isto se justifica pela existência de *datapaths* baseados em *software*, com número de portas dinâmico. Assim, torna-se aceitável especificar propriedades para uma porta arbitrária, ainda que se desconheça o número total de interfaces virtuais conectadas a um dispositivo qualquer.

- As informações de endereçamento IP opcionais são definidas desta maneira pela inexistência de funcionalidades que as utilizem. Mesmo que tais venham a ser incorporados à plataforma, a proposição da Seção 3.2.1.1 (satisfabilidade) determina que a lógica de cada serviço avalie a viabilidade de se operar sobre uma descrição fornecida;

- As mesmas considerações colocadas para o endereçamento IP valem para a especificação de velocidade de enlaces.

Sob tais condições, acredita-se estar demonstrada a correspondência entre os modelos de descrição previamente estabelecidos e a gramática de configuração RouteFlow. A seguir, é apresentada uma arquitetura de classes que se espera que possa representar computacionalmente uma topologia arbitrária descrita por meio dela.

5.2.2.2 Satisfabilidade

A linguagem de desenvolvimento da plataforma RouteFlow é a C++, razão pela qual se busca representar uma descrição topológica expressa através da gramática de configuração da seção anterior em classes de objetos. Tais elementos são, portanto, a interface apresentada a um serviço RouteFlow para conhecimento da infraestrutura controlada. Assim, é importante que se demonstre a consistência entre a expressividade da gramática e as classes propostas.

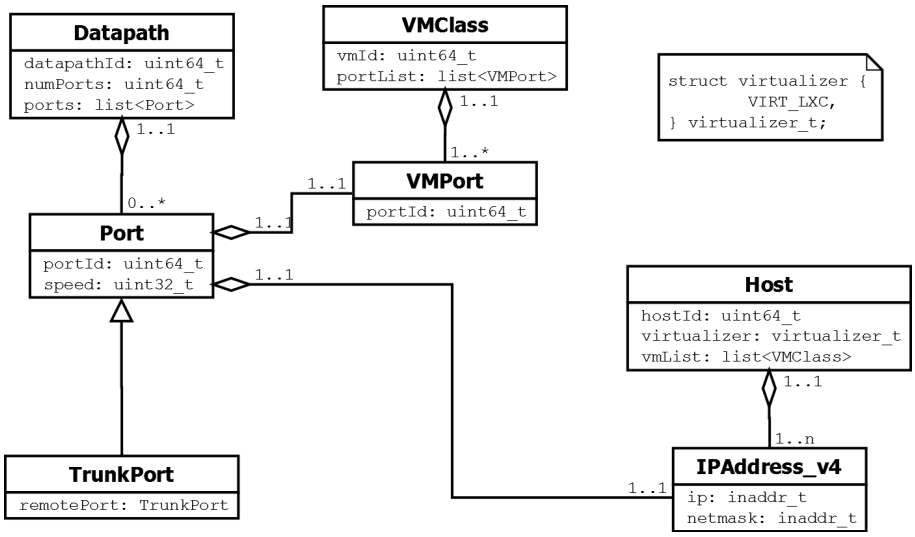


Figura 16: Arquitetura de classes sugerida para armazenamento dos dados da gramática de configuração.

A arquitetura pode ser vista na Figura 16. Os objetos da classe "Datapath" tem o campo identificador adequado, além de um atributo de número de portas. Uma relação $1 : n$ está estabelecida com a classe "Port", usada para representar de forma genérica as interfaces nestes dispositivos. A classe "Port", por sua vez, tem relação $1 : 1$ com a classe "IPAddress_v4", o que visa representar a atribuição de endereços IP a uma porta. O uso do sufixo "v4" justifica-se tanto pelo fato do RouteFlow e do próprio protocolo OpenFlow (STANFORD OPENFLOW TEAM, 2012) não suportarem ainda topologias IPv6 (KUROSE; ROSS, 2009), quanto para facilitar a incorporação deste suporte no futuro.

Os atributos da classe "Port", mais seu relacionamento com a classe "IPAddress_v4", são suficientes para representar uma porta de acesso. Para representação de uma porta de tronco, porém, uma hierarquia com uma classe "TrunkPort" foi estabelecida. Tal classe inclui um atributo que permite que uma instância da mesma esteja relacionada a outras do mesmo tipo – o que caracteriza o modelo de dados pretendido.

As características preconizadas por um nó anfitrião de topologias virtuais podem ser todas encontradas na classe "Host". Guarda-se uma relação para com a classe "IPAddress_v4" da mesma natureza daquela existente na classe "Port". Uma vez que a implementação-alvo é uma linguagem OO não-pura (LAFORE, 2002), um tipo definido por usuário, "virtu-

alizer_t", é usado para determinar a ferramenta de virtualização em uso por uma instância de "Host".

Estão representadas na arquitetura duas classes que não fazem parte dos modelos de representação e da gramática previamente colocados: "VMClass" e "VMPort". Isto é desejável e justificável, uma vez que não se espera que o operador estabeleça um mapeamento 1 : 1 entre portas dos planos de dados e de controle, devendo tal relação ser mantida como parte da lógica da plataforma de roteamento como serviço. Seu objetivo no diagrama, portanto, é de apenas exemplificar uma forma de correlação entre as portas de um *datapath* e as MVs do plano de controle.

Não estão representados os modelos de dados "Service" e "ServiceParam" pois, conforme detalhado na Seção 5.2.1.5, a interpretação de seu conteúdo é delegada ao serviço de roteamento a que dizem respeito – devendo portanto ser transcritos de maneira simples e flexível, de acordo com a implementação realizada. No caso do protótipo desenvolvido como parte do presente trabalho, as informações de nome de serviço, chave de parâmetro e respectivo valor compõem as tuplas de um vetor entregue ao serviço implementado.

Uma vez que tal arquitetura de classes cobre o universo descritivo provido pela gramática de configuração RouteFlow, e esta atende aos modelos de dados concebidos, pode-se inferir que a arquitetura é fiel à descrição topológica almejada. Conforme apresentado, tal descrição é suficiente para modelar a infraestrutura física de um AS arbitrário inteiramente baseado na plataforma RouteFlow. E assim sendo, basta apenas que um serviço escrito para esta plataforma implemente as interfaces de acesso a tais objetos, e será possível dotá-lo da lógica necessária para avaliar a satisfabilidade dos plano de dados em relação aos seus próprios requisitos de operação.

A disponibilidade das informações modeladas através de classes também confere dinamismo à representação. Uma vez que eventos do plano de dados estão disponíveis para os processos do plano de controle, torna-se possível replicar, diretamente nesta representação em memória, mudanças de conectividade ocorridas durante a execução da

plataforma – mantidas as premissas elencadas para a representação. Tal propriedade assegura que um dado serviço possa avaliar a satisfabilidade da infraestrutura subjacente *continuamente*.

Na próxima seção, a proposição de tempestividade enunciada na Seção 3.2.1.1 é colocada em perspectiva frente a mudanças propostas e incorporadas ao RouteFlow.

5.2.2.3 Tempestividade

Conforme detalhado na descrição da proposição 3 da Seção 3.2.1.1, era necessário incorporar ao RF-Slave mecanismos de coleta dados que reagissem a eventos de convergência ocorridos no plano de controle. Assim, poderia-se garantir a aplicação contínua da lógica de encaminhamento dos serviços disponíveis na plataforma, conforme ativados e parametrizados através da gramática de configuração construída.

Na implementação original a coleta de dados da FIB das MVs é realizada por meio de *scripts shell* executados periodicamente, através da chamada de sistema operacional "system()" (STEVENS; RAGO, 2005). Tal abordagem é funcional, mas implica na bifurcação de um processo interpretador de comandos ("/bin/sh") e ainda outros, subordinados a este. O RF-Slave também é efetivamente suspenso durante a coleta, por ser implementado em um único fluxo de execução.

Isto permitiu identificar três passos necessários à implementação da funcionalidade planejada:

1. Incorporar a coleta de dados da FIB ao próprio RF-Slave, eliminando o uso de *scripts* externos;
2. Condicionar o processo de coleta à ocorrência de eventos na FIB;
3. Garantir que múltiplos fluxos de execução possam ser utilizados, de maneira que diferentes eventos (ou *tipos* de eventos) possam ser tratados paralelamente.

Alternativas para sua realização foram pesquisadas na literatura.

A suíte de roteamento Quagga, utilizada nas MVs do plano de controle para executar protocolos de roteamento, é desenvolvida com base em uma filosofia modular. Cada protocolo é suportado por um *daemon* (STEVENS; RAGO, 2005) diferente. O protocolo BGP, por exemplo, é implementado através do *daemon* "bgpd".

O conjunto de *daemons* executando em um dado sistema é supervisionado por um último processo, chamado "zebra"[2]. Além da supervisão, também é função deste programa atualizar a FIB do sistema operacional em resposta a solicitações dos *daemons* de protocolos. Estas solicitações são recebidas através de um arquivo especial em disco, que simula uma conexão de *socket* de rede (STEVENS; RAGO, 2005) entre os vários *daemons* de protocolo e o *zebra*, e que permite a todos trocarem informações entre si.

Existe uma referência (URIARTE, 2001) disponível com instruções para o desenvolvimento de novos *daemons* que se integrem à solução Quagga. Portanto, seria possível adaptar o RF-Slave, transformando-o em um *daemon* que se comunicasse com o restante da plataforma e, assim, obtivesse informações sobre os protocolos de roteamento em execução. Sempre que um evento de convergência fosse identificado a partir de algum deles, seria possível consultar a FIB e reagir apropriadamente. A ferramenta de construção de circuitos GMPLS dinâmicos DRAGON (DRAGON TEAM, 2008) adota uma estratégia do tipo[3]: os circuitos virtuais construídos geralmente têm requisitos estritos de capacidade, que são atendidos através da execução de um novo *daemon*, que implementa o protocolo RSVP (KUROSE; ROSS, 2009).

Uma consequência importante desta abordagem, porém, é a dependência da funcionalidade construída (o serviço RSVP, no caso) em relação à suíte Zebra. Enquanto isto pode trazer pequeno impacto à operação do DRAGON, no caso de uma plataforma de roteamento como serviço poderia ser visto como uma limitação. Outras suítes de roteamento do estado da arte, como a XORP (HANDLEY; HODSON; KOHLER, 2003) e a BIRD(FILIP et al., 2012), não mais poderiam ser aplicadas ao seu plano de controle. Caso um ope-

[2]Isto se deve ao fato de o Quagga ser uma evolução não-oficial de outra plataforma de roteamento, chamada "Zebra", que foi descontinuada.

[3]A única diferença neste caso é que o DRAGON tem como alvo a suíte Zebra original, não a Quagga.

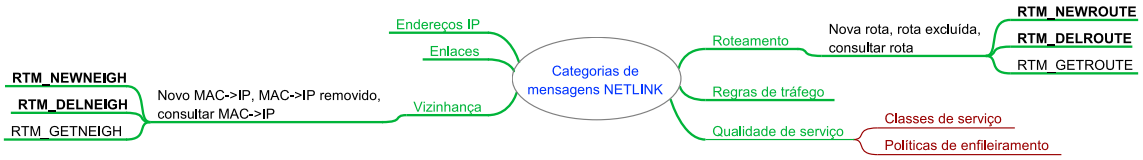


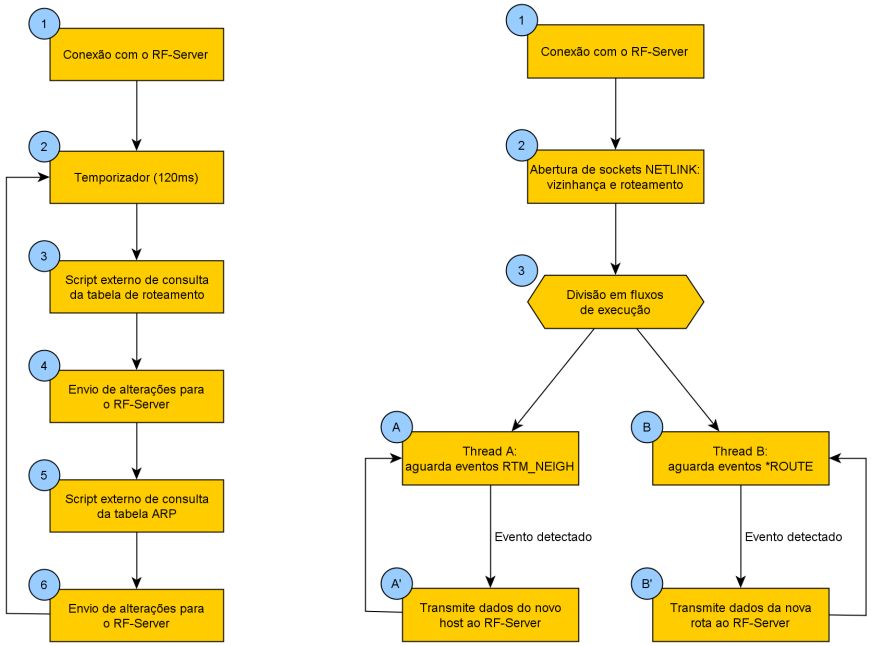
Figura 17: Hierarquia de mensagens RTNETLINK, com destaque para aquelas tratadas pelo *daemon* RF-Slave.

rador desejasse utilizar o XORP para implementar seus processos de encaminhamento, por exemplo, teria que modificar seu código para que replicasse o comportamento do RF-Slave.

O sistema Linux também conta com uma infraestrutura interna que simula uma comunicação de rede (*socket*) entre o núcleo do SO e um processo de espaço de usuário, chamada NETLINK. Na RFC apresentada em Salim et al. (2003) propõe-se adotá-la para a sinalização entre os mecanismos de encaminhamento e o plano de controle de uma rede. Isto porque em complemento à NETLINK está disponível um mecanismo de coleta de informações da FIB do sistema operacional, chamado RTNETLINK (UDUGAMA, 2006).

Todas as alterações realizadas na FIB são anunciadas via *sockets* RTNETLINK. Estes anúncios são transmitidos na forma de pacotes de dados *multicast*, onde cada grupo de destino corresponde a uma categoria de informação, como “tabela de roteamento” ou “tabela ARP”. Aos processos que precisem destes dados, cabe instanciar um canal de comunicação *socket* do tipo e especificar que grupos de informações desejam receber. A Figura 17 apresenta uma lista dos diferentes grupos de informações disponíveis, com destaque para aqueles aplicáveis ao RF-Slave. Nestes casos, também estão representados, nas extremidades de cada categoria, os eventos e tipos de mensagens de interesse para o agente.

A obtenção de eventos de modificação da FIB a partir do sistema operacional faz o RTNETLINK apresentar-se como uma opção de coleta de dados tempestiva e independente da suíte de roteamento em uso. Assim, esta ferramenta foi selecionada para a execução dos passos elencados anteriormente nesta seção. Porém, para que um dado programa possa usufruir de seus recursos, ele deve utilizar uma biblioteca de desenvolvi-



(a) Fluxo de execução original do RF-Slave.

(b) Fluxo de execução após a implementação de tempestividade.

Figura 18: Diferentes modelos de fluxos de execução para a lógica do RF-Slave.

mento homônima. Esta biblioteca não possuía, no momento da realização desta pesquisa, uma interface de acesso C++, que é a linguagem de desenvolvimento do RouteFlow. Portanto, a implementação iniciou-se pela adaptação do código da RTNETLINK para a convenção de chamada C++, o que permitiu o uso de suas funções como parte do código do RF-Slave.

A Figura 18 permite comparar a operação do RF-Slave antes e depois da implementação RTNETLINK. A Figura 18b mostra que o fluxo de execução monolítico do RF-Slave foi dividido em dois, através da implementação de *threads*. Uma *thread* do agente passou a estar inscrita no grupo *multicast* de anúncios sobre a tabela de roteamento, enquanto outra trata exclusivamente das atualizações da tabela de vizinhança (ARP). Quando uma mensagem é recebida em um dos fluxos de execução, o tipo de anúncio é avaliado: caso se refira a uma inclusão ou exclusão de entradas nas tabelas monitoradas, os dados relevantes são extraídos e enviados ao RF-Server para instrução do plano de dados.

Por definição, isto atende à proposição de *tempestividade* e conclui a incorporação

dos elementos tidos como necessários para a transição do RouteFlow em uma plataforma de roteamento como serviço. A partir do próximo capítulo, uma aplicação de roteamento baseada na plataforma definida por estas características é apresentada como prova de conceito.

6 Uma aplicação de roteamento como serviço

Após a construção de elementos que sugere-se serem suficientes para tornar a arquitetura RouteFlow uma plataforma de roteamento como serviço, faz-se necessário verificar se ela se comporta como tal.

Note-se que o sentido aqui empregado para o termo *plataforma* busca capturar a mesma noção expressa na Figura 4, do Capítulo 3: o RouteFlow deve prover a infraestrutura necessária para que uma lógica de negócio arbitrária seja desenvolvida e complemente a sua própria. Em outras palavras, sua missão é proporcionar uma infraestrutura de suporte a aplicações de roteamento, tanto quanto o sistema operacional de rede NOX proporciona uma fundação para a execução de controladores OpenFlow.

Assim, considera-se que esta pesquisa não estaria completa sem a construção de um exemplo de aplicação de roteamento operável sobre a então plataforma de roteamento como serviço RouteFlow.

Esta aplicação trata-se de um serviço de roteamento que permite a operação de um conjunto de *datapaths*, funcionalmente equivalentes a roteadores no contexto RouteFlow, a partir de seu mapeamento para uma única MV do plano de controle. Assim, ao invés de se gerenciar de forma distribuída *n* ou mais sessões BGP estabelecidas entre um AS e seus *n* pares, todas elas passam a ser terminadas em um único roteador, virtual. Mesmo a disseminação das informações de cada sessão dentro do AS deixa de ser fonte de sobrecarga operacional, pois são todas direcionadas para a mesma FIB. A definição de preferências por um enlace também se espera simplificada, já que o transporte de dados

por um canal específico passa a derivar-se de um processo único, com uma única lógica de encaminhamento e configuração.

Não foi possível identificar uma categoria específica na literatura que permita enquadrar esta ferramenta, razão considerada suficiente para nomeá-la "RFAgg", cunhando-se também a expressão "aplicação de roteamento agregado". Seu desenvolvimento permite não só uma prova de conceito dos artefatos construídos, mas também pode aplicar-se às noções de operação atômica de um AS discutidas ao longo das propostas deste trabalho, conforme relato a seguir.

A aplicação depende da característica de *expressividade*. É preciso informar a ela, por intermédio da gramática RouteFlow, quais dispositivos farão parte da operação agregada. Ela também implementa o teste de *satisfabilidade* previsto, de forma a avaliar a infraestrutura física disponível, descrita também por meio da gramática de configuração. Sua operação é compatível com os *instrumentos de virtualização* existentes na plataforma. E, por fim, sua lógica de operação é complementar àquela de um protocolo de roteamento como o BGP e o OSPF, na medida em que decisões baseadas em uma visão global da conectividade são tomadas. Isto se opõe à execução distribuída para a qual estes mesmos protocolos foram concebidos.

Ao incorporar a noção de roteamento global, o alvo da aplicação descrita assemelha-se àquele definido para o RCP (CAESAR et al., 2005) e sugere que seja possível construir, a partir de *datapaths*, um AS logicamente capaz de desempenhar roteamento atômico conforme a noção apresentada em Zhang-Shen, Wang e Rexford (2008). O serviço RFagg também pode ser comparado aos conceitos de *route-reflectors* (DUBE; SCUDDER, 1999) e comunidades BGP, que oferecem um tratamento centralizado para as informações de rotas para prefixos entre ASs.

O requisito para o desempenho da função proposta é a existência de conectividade física *full meshed* entre os *datapaths* que se deseja agregar. Sabe-se que nem sempre é possível implementar uma topologia do tipo em ASs do mundo real. Porém, recursos

existentes no arcabouço OpenFlow permitem que este modelo de operação seja extrapolado para redes onde a configuração física não é, de fato, uma malha completa. Isto faz parte da argumentação desenvolvida na Seção 6.4, onde o comportamento verificado para a plataforma RouteFlow e a aplicação RFAgg são comparativamente avaliados frente a soluções do estado da arte.

Antes disto, o modelo de operação da nova ferramenta será apresentado em detalhes. Em seguida, é documentado um estudo de caso realista e que comprova o funcionamento do serviço em um cenário de conectividade de malha completa.

6.1 Modelo de operação

O serviço desenvolvido considera apenas dois parâmetros para sua execução: "active" e "datapaths". O primeiro ativa ou desativa a funcionalidade provida pela aplicação, de acordo com os valores "yes" ("sim") ou "no" ("não"). O segundo é utilizado para informar a lista de *datapaths* que compõem o cenário de agregação. A seguir, pode-se visualizar um exemplo destes parâmetros para um caso hipotético de agregação de quatro dispositivos, com identificadores OpenFlow "0x01", "0x02", "0x03" e "0x04".

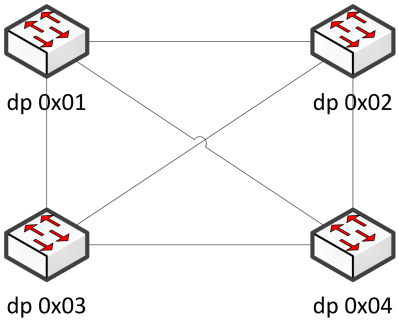
```
service rfagg active: yes
service rfagg datapaths: 0x01 0x02 0x03 0x04
```

A partir da interpretação da lista de dispositivos, torna-se possível consultar os objetos da classe "Datapath" mantidos pela lógica principal da plataforma e construídos com base na mesma gramática de configuração. Pela própria definição da arquitetura de classes apresentada na Seção 5.2.2.2 do Capítulo 5, também é possível acessar suas respectivas listas de objetos "Port" e "Trunk". Portanto, é possível explorar exaustivamente a conectividade entre *datapaths* descrita por estes objetos. E através desta exploração, obtém-se a matriz de adjacências (CORMEN et al., 2001) correspondente à infraestrutura da rede, conforme o algoritmo descrito na Figura 19.

Dados: Vetor E contendo cada $trunk$ do plano de dados.
Resultado: Matriz de adjacências da topologia física controlada.

```
1 início
2 |  E ⟵ tuplas(dp_origem, dp_destino);
3 |  Adj[] ⟵ ∅;
4 |  para e ∈ E faça
5 |  |  Adj[dp_origem, dp_destino] ⟵ 1;
6 |  fim
7 fim
```

Figura 19: Algorimo para construção da matriz de adjacências da infraestrutura de rede controlada.



Linhas/Colunas	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

Figura 20: Matriz de adjacências derivada da topologia física informada para uma rede hipotética.

Durante a inicialização, na linha 2, atribui-se a E todas as tuplas correspondentes a portas *trunk* do plano de controle. Um conjunto vazio é atribuído à matriz Adj na linha 3. O laço da linha 4 faz então com que, para cada par $(dp_{origem}, dp_{destino})$ de E, o valor 1 seja atribuído a um elemento $Adj[i, j]$.

A Figura 20 apresenta um exemplo de matriz de adjacências derivada da conectividade física descrita para um conjunto de *datapaths* "0x01", "0x02", "0x03" e "0x04". Estes elementos e suas conexões também estão representados.

Constituída a matriz de adjacências, torna-se trivial o teste de satisfabilidade do plano de dados. Conforme a Seção 5.2.2.2 do Capítulo 5, o objetivo deste teste é avaliar a conectividade física em relação aos requisitos de operação deste serviço – a existência de uma malha completa entre todos os *datapaths* controlados. Ora, uma vez que as arestas do grafo de conectividade física representadas na matriz não são orientadas, e pelas propriedades intrínsecas a esta estrutura de dados descritas em Cormen et al. (2001), para

realizar o teste é suficiente uma busca por elementos $Adj[i, j]$ que estejam fora da diagonal principal da matriz e cujo valor seja diferente de 1. Caso algum seja encontrado, então não há uma malha completa estabelecida.

É importante observar que a matriz construída não leva em consideração o atributo de velocidade eventualmente associado a cada enlace *trunk*, haja visto que esta prova de conceito não se propõe a implementar o comportamento *hot-potato*. Ainda assim, seria possível construir uma nova versão do algoritmo da Figura 19 que atribuísse a $Adj[dp_{origem}, dp_{destino}]$ o valor $\frac{1}{speed}$ de cada porta. Tal ponderação de arestas poderia suportar a execução de um algoritmo como o de Dijkstra (CORMEN et al., 2001). Uma vez que isto permitiria obter os caminhos ótimos entre todos os vértices (*datapaths*) existentes, torna-se possível uma implementação de comportamento análogo ao *hot-potato* no contexto deste serviço.

Satisfeitas as pré-condições de conectividade física necessárias para a realização do roteamento agregado, a plataforma de roteamento entra em execução conforme estabelecido na Seção 5.1 do Capítulo 5. Neste caso, porém, tão logo uma MV do plano de controle apresente-se ao processo RF-Server, o fluxo de execução é desviado para o código do serviço de agregação. Suas interfaces de rede são então mapeadas para todas as portas de comunicação de *datapaths* do tipo acesso listadas. Caso não houvesse um serviço de roteamento executando sobre a plataforma, o comportamento original, de mapear as portas de *um único datapath* para *uma única* MV, seria ativado (e mais MVs ainda deveriam se apresentar para que fosse possível controlar a rede corretamente).

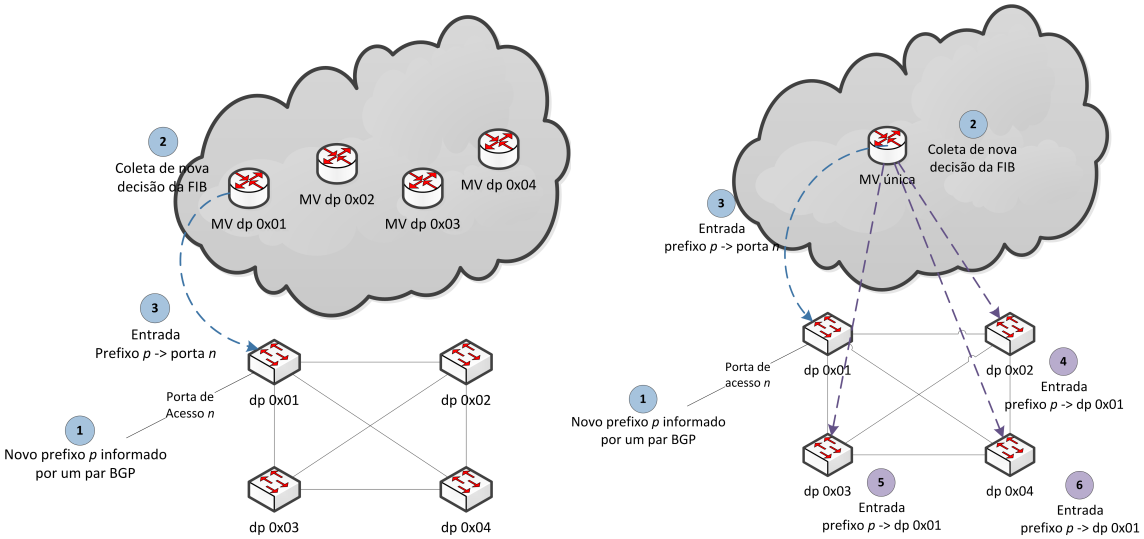
O mapeamento de portas de acesso para interfaces de uma única MV modifica ligeiramente o processo de coleta e processamento de alterações da FIB. Neste caso, a conversão de uma entrada de rota em uma regra exigirá consultar a que porta de *datapath* corresponde a interface de rede vinculada à nova rota. Na ausência do serviço de agregação, primeiro seria necessário determinar o *datapath* para o qual a MV que originou a rota está mapeada, e só então verificar a que porta corresponde a interface de rede à qual tal rota está associada.

Aqui é preciso ressaltar, à luz do que representa o controle do plano de dados, a ação crucial desempenhada pelo RouteFlow: a instalação de entradas de encaminhamento no plano de dados, que sintetizam o conhecimento do processamento da alcançabilidade entre pares de endereços de origem e destino. Portanto, para o caso de agregação, após a instalação de uma entrada *D* em um *datapath*, é necessário um último passo: instalar uma entrada correspondente à primeira em cada um dos outros *datapaths* do domínio. Isto porque o conhecimento topológico não deve mais ser individual (sob a perspectiva de roteadores "1:1") e nem parcial (em relação a decisões de roteamento tomadas individualmente).

Na Figura 21 os dois processos de instalação de entradas, regular e agregado, estão ilustrados. No caso da Figura 21a, que mostra o comportamento regular da plataforma, a informação de alcançabilidade de prefixo *p*, previamente desconhecido e disponível através da porta *n*, motiva a inclusão de uma nova rota na FIB da MV para a qual o *datapath* "0x01" está mapeado. Tal rota é coletada e traduzida em uma nova entrada, instalada no dispositivo correspondente. O paralelo traçado com o serviço de agregação na Figura 21b mostra que, para este caso, não só o *datapath* "0x01" é instruído, mas também os outros dispositivos do domínio são informados da disponibilidade de acesso ao prefixo *p* através dele.

Esta disseminação de informações de alcançabilidade é realizada através do algoritmo exibido na Figura 22.

A execução do algoritmo se dá a partir da instalação de uma nova entrada no plano de dados. Durante a inicialização, na linha 2, os dados desta entrada são armazenados em *D*. Espera-se obter o prefixo de destino (*destino*) e o dispositivo onde a entrada foi instalada (*datapath*). Na linha 3, o vetor *V* é construído a partir da topologia descrita pelas propriedades de objetos da classe "Trunk", previamente definida: cada enlace entre duas portas corresponderá a uma tupla $(dp_{origem}, porta_{origem}, dp_{destino}, porta_{destino})$. No *loop* da linha 4, cada enlace do vetor *V* é tomado, *v* a *v*. Para cada enlace *v* que tenha como destino o *datapath* que acabou de receber uma nova entrada (teste da linha 5),



(a) Instalação de entradas regular.

(b) Instalação de entradas com o uso do serviço de roteamento agregado.

Figura 21: Diferença entre o processo de instalação de entradas regular da plataforma RouteFlow e aquele adotado para realização de roteamento agregado.

uma nova instrução de encaminhamento é produzida e instalada nas linhas 6-9. Uma vez que a existência de uma malha completa é pré-condição para que se chegue a este ponto, isto significa que todos os dispositivos controlados receberão uma nova instrução. E tal instrução ordena a cada *datapath* que encaminhe pela sua porta de comunicação conectada ao $datapath(D)$ os pacotes endereçados ao novo *destino* descoberto (linhas 7 e 8).

Note-se que o algoritmo não faz menção aos endereços físicos (MAC) dos pacotes transmitidos entre *datapaths*. Isto porque, como parte das instruções da entrada D que motivou a execução do algoritmo da Figura 22, inclui-se que o endereço físico de origem de uma mensagem transmitida por uma porta p seja substituído pelo MAC da interface virtual para qual a mesma porta se encontra mapeada. Igualmente, instrui-se que o endereço físico de destino da mensagem seja substituído pelo MAC do nó que a receberá em seguida. Logo, tal entrada garante a consistência dos cabeçalhos dos pacotes que deixam a rede controlada.

É necessário considerar o caso de recepção de tráfego de resposta aos pacotes enviados por meio da entrada D, e que tenham como destino uma rede r conectada a um *data-*

Dados: Decisão de encaminhamento D transmitida para um $datapath$. Vetor V contendo cada $trunk$ do plano de dados.
Resultado: Decisão de encaminhamento disseminada pelo plano de dados.

```
1  início
2  |  D ⟵ (destino, datapath);
3  |  V ⟵ tuplas(dp_origem, porta_origem, dp_destino, porta_destino);
4  |  para v ∈ V faça
5  |  |  se dp_destino(v) = datapath(D) então
6  |  |  |  N ⟵ NovaEntradaOpenFlow(∅);
7  |  |  |  destino(N) ⟵ destino(D);
8  |  |  |  porta_destino(N) ⟵ porta_origem(v);
9  |  |  |  AdicionaEntradaOpenFlow(dp_origem(v), N);
10 |  |  fim
11 |  fim
12 fim
```

Figura 22: Algorimo para disseminação de decisões de encaminhamento pela rede controlada.

path D' que não $datapath(D)$. Uma vez que a transmissão original para a rede $destino(D)$ ocorreu, então informações para alcance de r fazem parte da FIB da MV agregadora. Neste caso, a incorporaçao destas informações à FIB levou à coleta de dados e à inserção de uma nova entrada em D'. Pressupõe-se também que o algoritmo da Figura 22 sempre será executado em resposta à instalação de novas entradas no plano de controle. Logo, ao receber o tráfego de resposta a uma comunicação tratada por D, o $datapath(D)$ pode transmitir os pacotes ao próximo salto correto, a partir das informações de alcançabilidade previamente disseminadas pelo domínio.

A seguir, são apresentados os procedimentos técnicos realizados para a materialização e a avaliação do conjunto completo de soluções.

6.2 Implementação do protótipo

Para confirmação da hipótese norteadora da presente pesquisa, foi necessário materializar um conjunto de abordagens e técnicas propostas até aqui. Esta seção detalha os passos para o alcance deste requisito.

O código RouteFlow a partir do qual o desenvolvimento foi realizado foi recuperado

de seu repositório público (FUNDAÇÃO CPQD, 2012) após o *commit* “fe84700c9fe...”.

A implementação da gramática de configuração descrita nas últimas seções foi realizada através das ferramentas *lex* e *yacc*. Todos os símbolos terminais definidos via *lex* estão listados no Anexo A. O código *yacc* não foi incluído como parte deste texto por concisão. A linguagem de programação utilizada para implementar tanto o *parser* da gramática, quanto o próprio RouteFlow e o serviço de roteamento proposto foi C++.

Para os testes preliminares do código em desenvolvimento, foi utilizada a ferramenta Mininet (LANTZ; HELLER; MCKEOWN, 2010). Esta ferramenta permite a definição de topologias virtuais baseadas em MVs LXC e *datapaths* Open vSwitch. Tais topologias foram designadas ao controle da versão da plataforma RouteFlow em desenvolvimento, para verificar tanto sua operação regular (“1:1”) quanto o comportamento do serviço de roteamento em implementação.

Os cenários de testes acompanham o que se pode ver na Figura 23. O plano de dados compunha-se de quatro MVs (*h1*, *h2*, *h3* e *h4*), cada uma delas conectada a um de quatro *datapaths* (*s5*, *s6*, *s7* e *s8*) interligados em malha completa. A MV de agregação também se baseou no LXC, cujo suporte, a esta altura, já se encontrava disponível no RouteFlow.

A este cenário, foram aplicadas alternadamente uma configuração da plataforma com um mapeamento “1:1” entre MVs e *datapaths* e uma com o serviço de agregação, onde todos os dispositivos eram mapeados para uma única MV. Nesta etapa, nenhum roteamento dinâmico foi considerado. Assim, apenas atualizações da FIB envolvendo resoluções ARP para os nós diretamente detectados foram testadas. Os testes realizados com mapeamento “1:1” tinham apenas a função de verificar que as alterações empreendidas não interfeririam com a operação regular da plataforma, o que foi confirmado.

A seguir, ilustra-se o comportamento obtido para o caso de agregação e que interessava para o intuito de sua avaliação. A Figura 24 apresenta o conteúdo da tabela ARP da MV de agregação após a execução de uma conexão do tipo *ping* entre todos os nós. A Tabela 3 exibe os fluxos de encaminhamento de cada um dos *datapaths* consistentemente

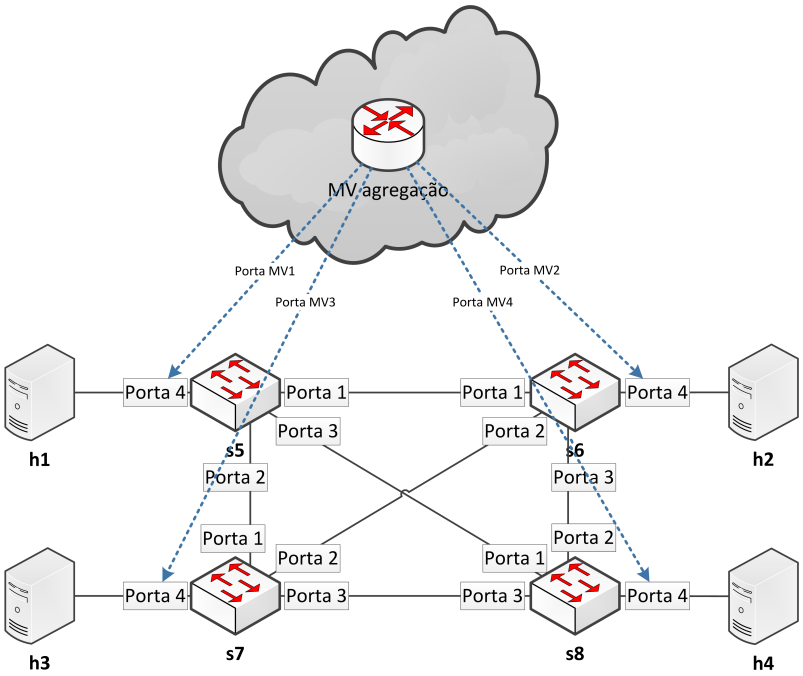


Figura 23: Cenário preliminar de testes para o serviço de agregação.

instalados.

```
Address         HWtype  HWaddress           Flags Mask  Iface
192.169.1.100   ether   00:23:20:f1:1d:41   C           eth0
200.1.254.6     ether   2a:61:6d:67:43:68   C           eth2
200.1.254.14    ether   8e:b0:47:61:f1:c7   C           eth4
200.1.254.2     ether   96:27:4e:af:dd:05   C           eth1
200.1.254.10    ether   ca:14:2f:73:04:2a   C           eth3
```

Figura 24: Entradas da tabela ARP da MV de agregação nos testes preliminares.

A obtenção destes resultados levou à evolução dos testes em um estudo de caso, que é objeto da próxima seção.

6.3 Estudo de caso

A aplicação experimental da aplicação de roteamento como serviço RFAgg desenvolvida ocorreu em uma rede inteiramente construída a partir de elementos de virtualização (evitando a necessidade de se dispor de dispositivos físicos). O elemento de central atenção nesta rede, representada na Figura 25, é o AS identificado pelo número 1000. Apesar de hipotético, ele guarda alguma semelhança para com a média dos sistemas autônomos

Tabela 3: Entradas de tabelas de encaminhamento após teste de comunicação entre MVs

Datapath	Destino	Ações
s5	200.1.254.14 (h4)	output:3
	200.1.254.10 (h3)	output:2
	200.1.254.6 (h2)	output:1
	200.1.254.2 (h1)	$ARP_{dst} = MAC_{h1}, ARP_{src} = MAC_{MV1}$, output:4
s6	200.1.254.14 (h4)	output:3
	200.1.254.10 (h3)	output:2
	200.1.254.6 (h2)	$ARP_{dst} = MAC_{h2}, ARP_{src} = MAC_{MV2}$, output:4
	200.1.254.2 (h1)	output:1
s7	200.1.254.14 (h4)	output:3
	200.1.254.10 (h3)	$ARP_{dst} = MAC_{h3}, ARP_{src} = MAC_{MV3}$, output:4
	200.1.254.6 (h2)	output:2
	200.1.254.2 (h1)	output:1
s8	200.1.254.14 (h4)	$ARP_{dst} = MAC_{h1}, ARP_{src} = MAC_{MV4}$, output:4
	200.1.254.10 (h3)	output:3
	200.1.254.6 (h2)	output:2
	200.1.254.2 (h1)	output:1

em produção na Internet – ou até os ultrapassa em complexidade. O AS conta com 6 conexões a outros ASs, quando a média global de interconexões entre ASs é 3,37 conexões, e a brasileira, 3,54 (ALVES et al., 2010). Ele tem uma relação de fornecimento de acesso a quatro redes, que compõem os ASs 1001, 1002, 1003 e 1004. Também troca trânsito com os ASs 8001 e 8002, que têm por sua vez uma relação cliente-fornecedor para com o AS 8000. Espera-se que as particularidades destes pares tenham pequena significância em relação aos objetivos perseguidos por este estudo: mais uma vez, a questão de interesse é a operação do AS 1000.

Tal operação importa porque demanda a manutenção de dois protocolos de roteamento distintos no AS 1000: o BGP, para troca de informações de alcançabilidade entre redes, e um IGP (como o OSPF) para obtenção de saltos dentro do próprio AS. Por si só, este aspecto já representa significativa sobrecarga administrativa. Porém, colocadas tais condições, podem se apresentar também os problemas de *loops* de roteamento e atrasos de convergência relatados em Dube e Scudder (1999). Além disso, a aplicação de uma política de negócio que envolva o uso de um caminho específico entre nós para o transporte de tráfego entre duas redes requer significativa engenharia: é preciso garantir que

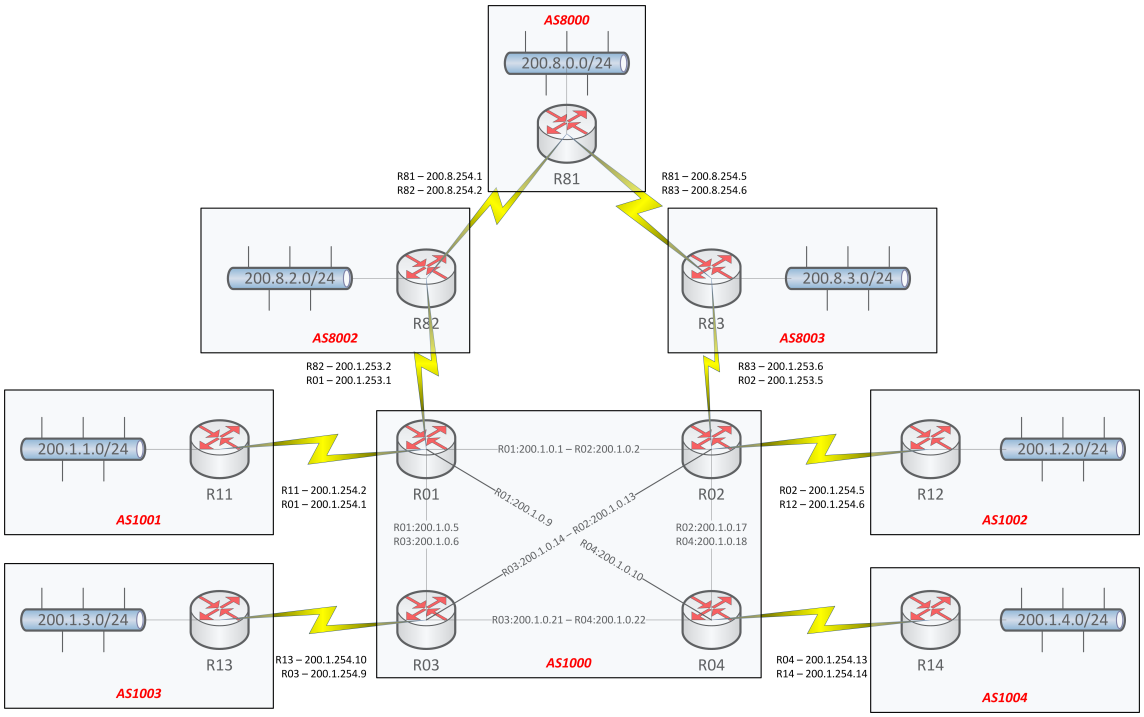


Figura 25: Apresentação do cenário ao qual se pretende aplicar as ferramentas previamente desenvolvidas.

as decisões independentes tomadas por cada dispositivo de encaminhamento R01, R02, R03 e R04 sejam consistentes entre si. Mas, conforme observado em Zhang-Shen, Wang e Rexford (2008), tal meta pode ser inalcançável em diferentes situações.

Algumas tentativas de solucionar um problema do tipo envolvem a incorporação de mais elementos à rede 1000 (CAESAR et al., 2005) ou modificações do protocolo BGP (ZHANG-SHEN; WANG; REXFORD, 2008), o que pode ser inconveniente ou custoso. Dadas as proporções da Internet e o número de ASs existentes, aplicá-las exigiria um grande esforço de implementação.

Abandonando por um momento estas questões de gerenciamento, as arquiteturas virtuais de roteamento IP são parte de uma nova abordagem para a operação de redes de computadores. Estas se fundamentam no paradigma de redes definidas por *software*, o qual tem como premissa o isolamento e separação de funções na conectividade digital. A arquitetura RouteFlow original (NASCIMENTO et al., 2011a) pode ser considerada um exemplo de proposta do tipo.

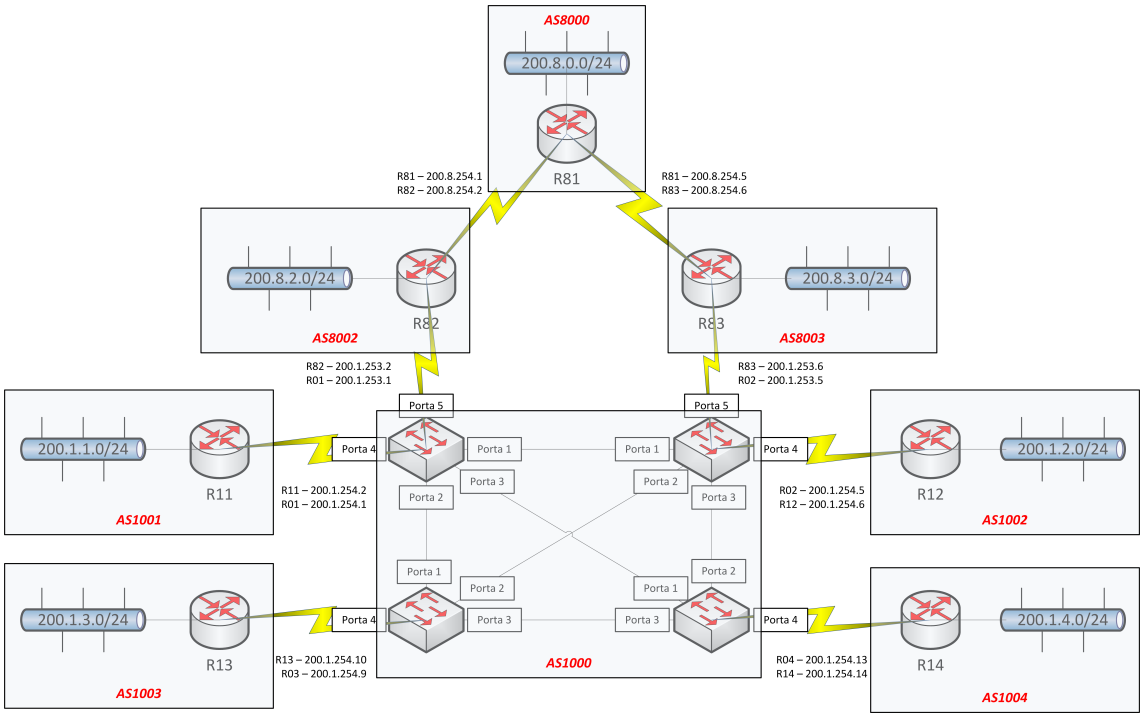


Figura 26: Visão da conectividade do caso estudado, com substituição de roteadores no AS 1000 por *datapaths*.

Para que se possa considerar a aplicação do RouteFlow original ao controle do AS1000 (e aqui tornamos a nos concentrar nos desafios de gerência de sua topologia) é necessário substituir os seus roteadores R01, R02, R03 e R04 por *datapaths*. Cada um destes dispositivos tem então suas portas de comunicação mapeadas para as interfaces de comunicação de uma MV. Cada MV faz parte de um plano de controle em que desempenha o antigo papel dos roteadores que foram eliminados. A partir das decisões tomadas por cada MV e refletidas em suas respectivas FIBs, instruções são transmitidas ao plano de dados que permitem a seus elementos encaminhar tráfego tal qual equipamentos tradicionais o fariam.

A substituição dos quatro roteadores por *datapaths* foi considerada para o desenho da Figura 27, onde a conectividade entre as diferentes redes foi consistentemente ajustada.

A troca do controle distribuído por uma abstração mantida em uma arquitetura de controle pode ter grandes resultados sob o aspecto de desempenho. Conforme visto na Seção 5.2.2.3, ao longo do tempo todas as informações de alcance de prefixos e sistemas finais mantidas na FIB das MVs do plano virtual são transferidas para as tabelas de

encaminhamento de cada *datapath*. Uma maior velocidade de encaminhamento então torna-se compreensível, já que não mais é necessário submeter cada pacote de tráfego a um complexo fluxo de decisão de roteamento (representado pela pilha de protocolos que um pacote deve atravessar para ser encaminhada em um roteador tradicional). Porém, do ponto de vista de gerenciamento os ganhos são menos perceptíveis, se é que existem: para operar a rede do AS 1000 é necessário gerenciar a topologia virtual que compõe seu plano de controle, e que está sujeita aos mesmos desafios de configuração da versão "tradicional" da rede. Esta foi uma percepção construída a partir das sugestões de Keller e Rexford (2010), e que ajudou a nortear este trabalho de pesquisa.

Mas aplicando-se ao cenário do AS 1000 a plataforma de roteamento como serviço construída a partir do RouteFlow, uma nova perspectiva se configura. Passa a ser possível executar a aplicação RFAgg e, assim, modificar a maneira como se opera o sistema autônomo.

Esta mudança passa pelo novo paradigma de mapeamento entre a conectividade física e o plano virtual. Uma ilustração do mesmo pode ser vista na Figura 27. Note-se que a conectividade com outras redes não está representada, para preservar a clareza do diagrama. Existem seis interfaces virtuais de uma MV "agregada" vinculadas ao OVS do plano de controle. Elas estão mapeadas em uma relação "1:1" com as portas de acesso no plano de dados: duas pertencentes ao *datapath* 0x01, respectivamente conectadas aos roteadores R11 e R82; duas pertencentes ao *datapath* 0x02, conectadas aos roteadores R12 e R83; uma relativa ao 0x03, conectada ao roteador R13; uma ainda referente a 0x04, interligada ao equipamento 14. As portas de conexão entre *datapaths* que fazem parte do AS 1000 não são representadas na topologia virtual.

Assim espera-se, com a aplicação do conceito de roteamento agregado, que seja possível não só desempenhar a função de controle esperada do AS 1000 (o resultado já entregue pelo RouteFlow original) mas gerenciar este papel a partir de uma abstração de um único roteador – representado pela MV agregadora. E uma vez que os elementos necessários para tal já foram elencados e documentados nos capítulos anteriores, vejamos como

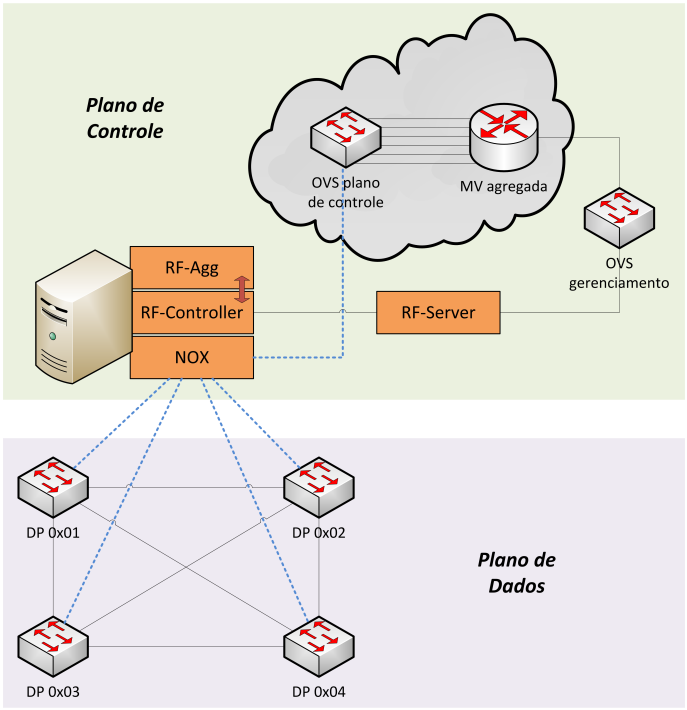


Figura 27: Mapeamento entre a conectividade física e o plano virtual a partir do uso na aplicação RFAgg.

isto se realiza.

O primeiro passo é o uso da gramática de configuração RouteFlow para descrever a topologia do AS 1000. Esta descrição encontra-se reproduzida na Figura 28. Pode-se ver que ela sintetiza a gama de informações embutida nos modelos enunciados na Seção 5.2.1 do Capítulo 5. Cada *datapath* é declarado com seu respectivo número de portas de comunicação. As portas utilizadas para conexão entre os dispositivos (portas de tronco) são todas consistentemente descritas. Não se faz menção às portas de acesso, que seriam usadas para conectar os equipamentos do AS 1000 a outros sistemas autônomos. Este é um reflexo da implementação realizada, onde portas não descritas são automaticamente consideradas como sendo daquele tipo.

Outra particularidade da implementação é a necessidade de se produzir apenas uma linha de descrição dos enlaces formados por duas portas de tronco: a própria lógica de operação previamente descrita para o serviço RFAgg os supõe bidirecionais. Portanto, ao se descrever que a porta p_1 do *datapath* d_1 está conectada à porta p_n do dispositivo d_n, o código da aplicação infere que o inverso também é verdadeiro.

```
dpid 0x1 ports 5
dpid 0x2 ports 5
dpid 0x3 ports 4
dpid 0x4 ports 4

dpid 0x1 port 1 trunk dpid 0x2 port 1 speed 1000
dpid 0x1 port 2 trunk dpid 0x3 port 1 speed 1000
dpid 0x1 port 3 trunk dpid 0x4 port 1 speed 1000

dpid 0x2 port 2 trunk dpid 0x3 port 2 speed 1000
dpid 0x2 port 3 trunk dpid 0x4 port 2 speed 1000

dpid 0x3 port 3 trunk dpid 0x4 port 3 speed 1000

service rfagg active: yes
service rfagg datapath: 0x1 0x2 0x3 0x4
```

Figura 28: Arquivo de configuração RouteFlow para controle da rede proposta.

Conforme o estudo pretendido, o comportamento a ser desempenhado pela aplicação RFAgg também precisa ser explicitado no arquivo de configuração. Isto foi feito, como se pode ver nas linhas finais da listagem.

Após descrever a topologia subjacente e os serviços de roteamento que se deseja ativar, é preciso um último passo: a ativação do protocolo BGP, desta vez sob a perspectiva global almejada. As sessões de troca de informações de prefixos estabelecidas para com os outros ASs partirão agora de um único ponto. Este ponto passa a ser responsável pela execução do único processo de roteamento ativo em todo o AS. Eventuais interferências ou modificações promovidas na operação deste processo também afetarão a todos os elementos do domínio administrativo de maneira uniforme. Cada decisão de encaminhamento, enfim, será disseminada para todos os *datapaths*. Isto resulta na necessidade de uma única configuração BGP para o AS 1000, listada na Figura 29. Na configuração estão reunidas as declarações de todas as sessões BGP fechadas com outros ASs. Pela definição do próprio serviço RFAgg, nenhuma outra configuração de protocolo é necessária, nem mesmo de IGPs.

O cenário descrito na Figura 30 foi implementado em um sistema executando o Debian GNU/Linux 5.0 e que estava dedicado à execução deste estudo. Todas as configurações propostas foram inseridas em uma versão do RouteFlow que implementava os conceitos previstos ao longo desta pesquisa.

```
password routeflow
enable password routeflow
!
router bgp 1000
        no synchronization
        redistribute connected
        neighbor 200.1.254.2 remote-as 1001
        neighbor 200.1.254.6 remote-as 1002
        neighbor 200.1.254.10 remote-as 1003
        neighbor 200.1.254.14 remote-as 1004
        neighbor 200.1.253.2 remote-as 8002
        neighbor 200.1.253.6 remote-as 8003
!
log file /var/log/quagga/bgpd.log
!
```

Figura 29: Arquivo de configuração “bgpd” para a rede proposta.

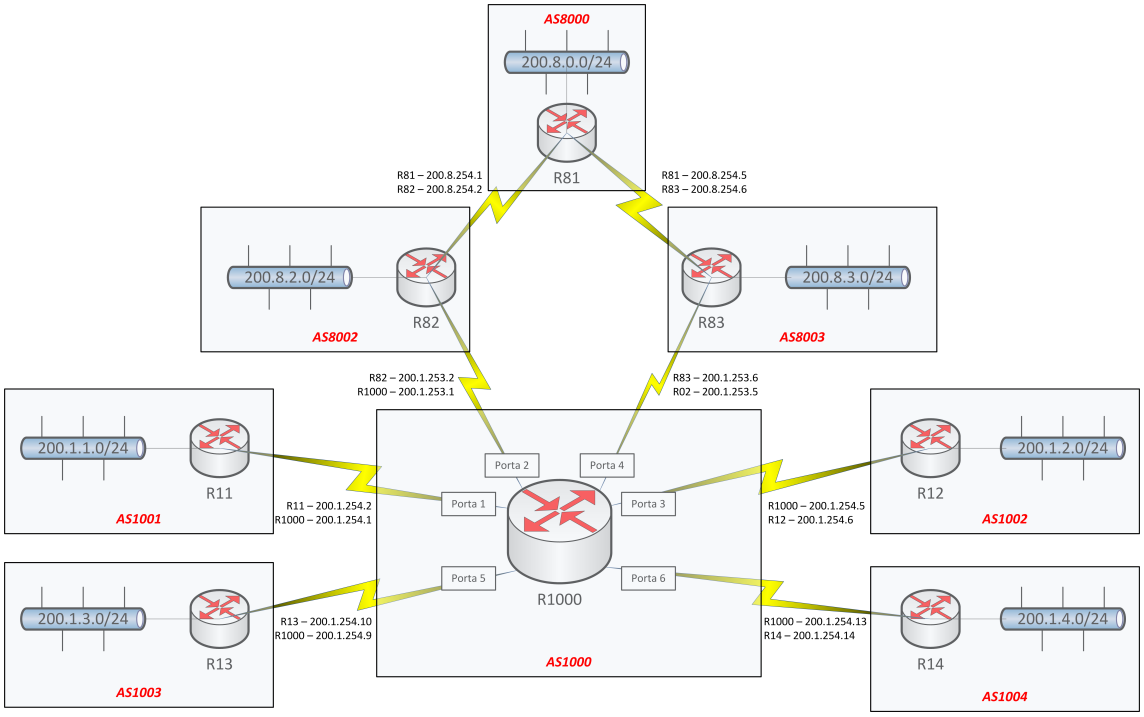


Figura 30: Operação obtida a partir da execução do serviço de agregação.

As informações de alcançabilidade foram corretamente disseminadas através do AS 1000. Foi possível verificá-lo através do comando “route” dos roteadores operando de forma legada (R11, R12, R13, R14, R82, R83 e R81) que estes incorporaram à sua FIB tanto os prefixos de redes diretamente conectadas quanto de redes remotas. Também foram realizadas tentativas de comunicação a partir de todos os roteadores do cenário, rumo a todas as interfaces de rede disponíveis, inclusive aquelas conectadas aos segmentos intra-ASs, como por exempo as redes 200.1.1.0/24 e 200.8.2.0/24. Todas foram bem-

Tabela 4: Entradas da tabela de roteamento da MV de agregação

Destino	*Gateway*	Máscara	Interface
200.1.254.8	0.0.0.0	255.255.255.252	eth5
200.1.254.12	0.0.0.0	255.255.255.252	eth6
200.1.254.0	0.0.0.0	255.255.255.252	eth1
200.1.254.4	0.0.0.0	255.255.255.252	eth3
200.1.253.0	0.0.0.0	255.255.255.252	eth2
200.1.253.4	0.0.0.0	255.255.255.252	eth4
200.8.254.0	200.1.253.2	255.255.255.252	eth2
200.8.254.4	200.1.253.6	255.255.255.252	eth4
200.1.4.0	200.1.254.14	255.255.255.0	eth6
200.1.1.0	200.1.254.2	255.255.255.0	eth1
200.1.2.0	200.1.254.6	255.255.255.0	eth3
200.1.3.0	200.1.254.10	255.255.255.0	eth5
200.8.2.0	200.1.253.2	255.255.255.0	eth2
200.8.3.0	200.1.253.6	255.255.255.0	eth4
200.8.0.0	200.1.253.6	255.255.255.0	eth4

sucedidas.

A Tabela 4 lista as rotas presentes na memória da MV de agregação, também consultadas por meio do comando “route”. Como se pode ver, a conectividade plena da rede se estende a esta abstração que, estabelecendo com sucesso sessões BGP para com todos os seus pares, incorporou todas as informações de seus prefixos remotos à sua própria FIB. As entradas com *gateway* 0.0.0.0, é importante registrar, são representadas desta forma por se tratarem de segmentos diretamente conectados. O pleno funcionamento do serviço de roteamento, comunicando-se com todos os seus pares, permite inferir a correta operação da resolução ARP em nível de enlaces.

Igualmente, a partir da consulta às entradas de fluxos OpenFlow instaladas no plano de dados, foi possível verificar que o processo de coleta de dados da FIB, bem como sua tradução em uma lógica de encaminhamento distribuída, foi bem-sucedido. A Tabelas 5 e 6 listam as entradas OpenFlow existentes nos *datapaths* após a completa inicialização da topologia, sendo representados os dispositivos 0x01 e 0x02 na primeira e 0x03 e 0x04 na segunda. Para facilitar a leitura das mesmas e a análise de suas respectivas informações, um campo “Núm.” foi acrescentado aos dados de encaminhamento. Tal campo permite que este texto se referencie a cada regra em um *datapath* pelo número que a precede.

Tabela 5: Entradas de tabelas de encaminhamento após teste de comunicação entre MVs

Datapath	Núm.	Destino	Ações
0x01	1	200.1.254.10 (r13)	output:2
	2	200.1.254.6 (r12)	output:1
	3	200.1.254.2 (r11)	$ARP_{dst} = MAC_{r11}, ARP_{src} = MAC_{MV}$, output:4
	4	200.1.253.6 (r83)	output:1
	5	200.1.254.14 (r14)	output:3
	6	200.1.253.2 (r82)	$ARP_{dst} = MAC_{r82}, ARP_{src} = MAC_{MV}$, output:5
	7	200.8.0.0/24	output:1
	8	200.1.2.0/24	output:1
	9	200.1.1.0/24	$ARP_{dst} = MAC_{r11}, ARP_{src} = MAC_{MV}$, output:4
	10	200.1.3.0/24	output:2
	11	200.1.4.0/24	output:3
	12	200.8.2.0/24	$ARP_{dst} = MAC_{r82}, ARP_{src} = MAC_{MV}$, output:5
	13	200.8.3.0/24	output:1
	14	200.8.254.4/30	output:1
	15	200.8.254.0/30	$ARP_{dst} = MAC_{r82}, ARP_{src} = MAC_{MV}$, output:5
0x02	1	200.1.254.10 (r13)	output:2
	2	200.1.254.6 (r12)	$ARP_{dst} = MAC_{r12}, ARP_{src} = MAC_{MV}$, output:4
	3	200.1.254.2 (r11)	output:1
	4	200.1.253.6 (r83)	$ARP_{dst} = MAC_{r83}, ARP_{src} = MAC_{MV}$, output:5
	5	200.1.254.14 (r14)	output:3
	6	200.1.253.2 (r82)	output:1
	7	200.8.0.0/24	$ARP_{dst} = MAC_{r83}, ARP_{src} = MAC_{MV}$, output:5
	8	200.1.2.0/24	$ARP_{dst} = MAC_{r12}, ARP_{src} = MAC_{MV}$, output:4
	9	200.1.1.0/24	output:1
	10	200.1.3.0/24	output:2
	11	200.1.4.0/24	output:3
	12	200.8.2.0/24	output:1
	13	200.8.3.0/24	$ARP_{dst} = MAC_{r83}, ARP_{src} = MAC_{MV}$, output:5
	14	200.8.254.4/30	$ARP_{dst} = MAC_{r83}, ARP_{src} = MAC_{MV}$, output:5
	15	200.8.254.0/30	output:1

Tabela 6: Entradas de tabelas de encaminhamento após teste de comunicação entre MVs

Datapath	Núm.	Destino	Ações
0x03	1	200.1.254.10 (r13)	$ARP_{dst} = MAC_{r13}, ARP_{src} = MAC_{MV}$, output:4
	2	200.1.254.6 (r12)	output:2
	3	200.1.254.2 (r11)	output:1
	4	200.1.253.6 (r83)	output:2
	5	200.1.254.14 (r14)	output:3
	6	200.1.253.2 (r82)	output:1
	7	200.8.0.0/24	output:2
	8	200.1.2.0/24	output:2
	9	200.1.1.0/24	output:1
	10	200.1.3.0/24	$ARP_{dst} = MAC_{r13}, ARP_{src} = MAC_{MV}$, output:4
	11	200.1.4.0/24	output:3
	12	200.8.2.0/24	output:1
	13	200.8.3.0/24	output:2
	14	200.8.254.4/30	output:2
	15	200.8.254.0/30	output:1
0x04	1	200.1.254.10 (r13)	output:3
	2	200.1.254.6 (r12)	output:2
	3	200.1.254.2 (r11)	output:1
	4	200.1.253.6 (r83)	output:2
	5	200.1.254.14 (r14)	$ARP_{dst} = MAC_{r14}, ARP_{src} = MAC_{MV}$, output:4
	6	200.1.253.2 (r82)	output:1
	7	200.8.0.0/24	output:2
	8	200.1.2.0/24	output:2
	9	200.1.1.0/24	output:1
	10	200.1.3.0/24	output:3
	11	200.1.4.0/24	$ARP_{dst} = MAC_{r14}, ARP_{src} = MAC_{MV}$, output:4
	12	200.8.2.0/24	output:1
	13	200.8.3.0/24	output:2
	14	200.8.254.4/30	output:2
	15	200.8.254.0/30	output:1

Constata-se que os destinos das entradas de encaminhamento em todos os *datapaths* acompanham a mesma ordem. Tal ordem não se mantém entre diferentes execuções da plataforma, porque mesmo a conectividade entre dispositivos de rede virtuais é assíncrona. Para a mesma execução, porém, todos os equipamentos do AS 1000 apresentam entradas ordenadas igualmente. Tal comportamento é um reflexo natural da lógica de processamento da aplicação RFAgg: a cada vez que uma nova entrada da FIB é coletada, todos os *datapaths* do domínio recebem a instrução correspondente no plano de dados.

A ordenação das entradas também permite inferir que o critério de priorização de regras baseadas em prefixos de rede funciona, com o objetivo de implementar o encaminhamento IP baseado no prefixo mais longo (*Longest-Prefix Matching* - LPM). Entradas baseadas na descoberta de um nó diretamente conectado (as 6 primeiras para cada dispositivo) são posicionadas no início da lista. Os destinos para prefixos que não se referem a nós "descobertos" a partir de uma entrada ARP, mas a redes remotas, são enviados para posições inferiores, com aquelas de máscara mais longa ocupando o fim da listagem.

Auferida a operação da plataforma RouteFlow nos termos estabelecidos em conseqüência da hipótese desta pesquisa, na próxima seção a generalidade e os possíveis desdobramentos dos resultados verificados serão avaliados.

6.4 Discussão de resultados

Aplicar o comportamento apresentado para o serviço RFAgg a uma rede do mundo real envolve uma relação custo-benefício. Esta relação é detalhada a seguir, em dimensões que se relacionam com as de operação de um AS. Conforme aplicável, este comportamento também é comparativamente analisado frente a outras soluções do estado da arte.

```
service rfagg preference: 200.8.0.0/24 0x01:5
```

Figura 31: Exemplo de instrução que poderia traduzir uma necessidade de negócio para um AS.

6.4.1 Gerenciamento

Sob esta dimensão, em (ZHANG-SHEN; WANG; REXFORD, 2008) constata-se o esforço de gerenciamento necessário para manter a operação de múltiplos roteadores em um AS:

> "How can a distributed collection of routers realize a policy? Suppose for a moment that an AS consists of a single router. That router would learn all the routes, and assign routes to all its links according to the ASś policy. The landscape changes in several interesting ways when an AS has multiple routers[...]" (ZHANG-SHEN; WANG; REXFORD, 2008)[1]

A solução aqui apresentada efetivamente reduz a operação de um AS à atividade de manutenção de uma única unidade de encaminhamento.

Por outro lado, ainda a respeito de Zhang-Shen, Wang e Rexford (2008), perseguem-se duas funcionalidades que não são intrínsecas ao BGP, mas deveriam ser implementadas na visão dos autores: a capacidade de um roteador designar diferentes rotas para diferentes enlaces, e o desacoplamento entre a seleção e a disseminação de rotas. O alvo de ambas é ampliar o universo de rotas divulgadas em um AS, porque tradicionalmente um roteador divulga para seus pares apenas a "melhor rota" escolhida, o que de algum modo limita os caminhos disponíveis para aqueles que recebem tais anúncios e pode levá-los a escolhas não-ótimas ou que, ainda que ótimas, estejam em desacordo com a política de negócio vigente.

Para a priorização de enlaces que não violem a política de negócio vigente, um parâmetro adicional poderia ser incorporado ao RFAgg, sendo perfeitamente compatível com a gramática de configuração construída neste trabalho. Um exemplo do tipo é apresentado na Figura 31.

[1] "Como pode uma coleção distribuída de roteadores realizar uma política? Suponha por um momento que um AS consista de um único roteador. Tal roteador aprenderia todas as rotas, e designaria rotas para todos os seus enlaces de acordo com a política do AS. O cenário muda de diversas maneiras quando um AS tem múltiplos roteadores[...]".

Para este exemplo, o serviço RFAgg poderia instruir o plano de controle a transmitir sempre pela porta 5 do *datapath* 0x01 o tráfego destinado a 200.8.0.0/24. Tal comportamento só seria diverso nas ocasiões em que esta rota não estivesse disponível.

É bem verdade que ainda haveria um problema a ser tratado: a escolha de uma única "melhor rota" pelo processo BGP. Uma vez que apenas uma rota é incorporada à FIB pelo protocolo, não haveriam múltiplas opções disponíveis para a aplicação RFAgg, que nunca poderia efetivamente optar por este ou aquele caminho. A alternativa disponível seria substituir o único processo BGP utilizado no plano de controle por uma versão capaz de instalar na FIB múltiplas "melhores rotas" de mesmo custo. Ao menos uma versão com esta capacidade está publicamente disponível (GOOGLE, 2012), e baseia-se na mesma suíte Quagga já utilizada pela topologia virtual RouteFlow (o que é conveniente, mas não chega a ser um pré-requisito, conforme visto na Seção 5.2.2.3).

A escolha de uma rota que satisfaça a política de negócio vigente, já tratada, é um problema distinto da escolha do caminho ótimo para transmissão de tráfego. Esta é trivial para as topologias a que este trabalho se aplica. Uma vez que todos os *datapaths* estão conectados em uma malha completa, então todos os outros dispositivos do AS encontram-se a apenas um salto de distância. Mas, mesmo redes que não possuam uma topologia física em *full mesh* podem implementar uma solução de roteamento baseado em um único salto. O próprio trabalho de Zhang-Shen, Wang e Rexford (2008) considera, por exemplo, o uso de túneis IP-em-IP para garantir que seus objetivos de encaminhamento sejam realizados.

Seria viável implementar um serviço que, a partir da matriz de adjacências de uma topologia arbitrária, calculasse os melhores caminhos entre todos os pares de vértices (*datapaths*) do grafo de conectividade do AS. O algoritmo de Dijkstra, previamente citado, seria uma opção para tal. A escolha da rota para transmitir pacotes de um ponto a outro da rede poderia ser então apoiada pelos caminhos previamente calculados. O instrumento de túneis, por sua vez, poderia ser substituído pelo emprego de *tags* de VLANs ou *labels* MPLS (SHARAFAT et al., 2011), que podem ser ambos programaticamente definidos em um contexto SDN (PFAFF et al., 2011).

O conceito de simular caminhos de um único salto entre os roteadores de um AS também aparece em Caesar et al. (2010). Naquele trabalho sugere-se o emprego de MPLS como instrumento para a comutação de pacotes, que seria orientada por protocolos de sinalização como o RSVP e o LDP. Ambos já foram anteriormente aplicados a cenários de redes SDN (GUDE et al., 2008) (SHARAFAT et al., 2011), o que pode ser uma indicação da viabilidade de se combinar tais implementações à plataforma aqui projetada, na forma de serviços.

Outras considerações realizadas em Caesar et al. (2010) poderiam ser aplicáveis à plataforma RouteFlow construída, mas há um ponto em que ambos os trabalhos se apartam significativamente. Enquanto a presente pesquisa parte de um paradigma em que o plano de controle é responsável por todas as decisões de roteamento, aquela sugere que o plano de dados retenha a função de reagir a eventos de desconexão e reconexão de enlaces. Tal possibilidade não foi considerada para efeito dos estudos aqui realizados.

6.4.2 Engenharia de tráfego

As entradas de encaminhamento reproduzidas nas Tabelas 5 e 6 da Seção 6.3 refletem um comportamento intrínseco ao protocolo BGP, executado na MV de agregação. Como se pode verificar pelas entradas de número "7" em cada *datapath*, tráfego com destino à rede 200.8.0.0/24, por exemplo, é sempre encaminhado para o dispositivo 0x02, ainda que 0x01 tenha um caminho de menor custo disponível a partir de sua porta 5 e através do roteador R82. Ocorre que, por definição, o protocolo escolhe uma *única* "melhor" rota para transmissão, sendo correto do ponto de vista de uma MV que representa o AS 1000 de forma atômica que se faça a opção por um dos caminhos via R82 ou via R83. Assim, apenas um dos dois caminhos pode ser escolhido, incorporado à sua FIB e, por conseguinte, disseminado por toda a rede. Portanto, tal como implementado, o serviço de roteamento agregado RFAgg não reproduz o comportamento *hot-potato* manifestado em execuções de roteamento tradicionais.

A ausência do *hot-potato* não implica em uma deficiência na visão de plataforma de

roteamento como serviço aqui elaborada. Pelo contrário, a descrição da infraestrutura de rede subjacente permite que se incorpore esta característica à aplicação desenvolvida, conforme previsto inclusive neste capítulo, durante a concepção do modelo de tratamento de informações de portas de comunicação. Outro aspecto relevante é a viabilidade de se experimentar outras abordagens, diferentes da *hot-potato*, que poderiam compor a lógica de um novo serviço e cuja política poderia ser sintetizada em parâmetros de configuração. A própria noção de que pode ser vantajoso mover o processo decisório do roteamento para além do *hot-potato* já foi anteriormente apresentada em Teixeira et al. (2004), (CAESAR et al., 2005), Caesar et al. (2010) e Caesar et al. (2005).

Em Caesar et al. (2010) ainda é proposto que se considere evitar completamente o processo de convergência de protocolos, especificamente de IGPs. Relativamente a esta idéia, o serviço RFAgg não depende de um IGP para a realização de sua operação de encaminhamento. Porém, também não é capaz de reagir apropriadamente a uma falha em um dos enlaces de sua malha intra-AS. A queda de um destes enlaces não levaria a uma nova decisão de roteamento, o que por definição impediria que uma ação reativa pudesse ser empregada. Por outro lado, o cálculo de menores caminhos entre vértices previamente discutido poderia embasar a incorporação desta funcionalidade.

Caesar et al. (2010) considera também "separar o roteamento dos roteadores". Propõe-se que o cálculo de rotas não seja delegado aos dispositivos de encaminhamento em si, mas seja realizado em caráter "fora-da-banda" e baseado na topologia de rede projetada, em oposição aos estados ligado/desligado dos enlaces de uma rede (usados por IGPs). A pesquisa aqui realizada fornece um meio de descrição da topologia controlada, e o serviço de roteamento construído realiza encaminhamento baseando-se em uma descrição do tipo. Mais ainda, a separação das funções de encaminhamento e controle é intrínseca ao modelo operacional da plataforma RouteFlow. Por estes pontos, mais o tratamento dado à questão de IGPs, novamente este trabalho e Caesar et al. (2010) coincidem.

6.4.3 Desempenho e escalabilidade

Ao mesmo tempo em que a perfeita coincidência da ordem das entradas instaladas nos *datapaths* demonstra corretude, denota a serialização das ações de resposta às alterações da FIB. Esta observação leva a considerar a possibilidade de que para um AS que conte com centenas de dispositivos, mensagens de atualização de FIB sejam recebidas a uma taxa maior que a velocidade de instalação de regras em *datapaths*. Tal possibilidade não foi investigada para o escopo desta pesquisa. Mas, poderia fazer sentido implementar múltiplos processos paralelos de envio de instruções ao plano de dados, talvez na proporção de uma *thread* para cada *datapath*.

O consumo de uma entrada de encaminhamento para cada prefixo alcançável poderia levar a plataforma a ultrapassar o limite de informações na tabela de fluxos de um dispositivo OpenFlow. A este fator soma-se outro, aparentemente de menor relevância: também se aloca uma entrada para cada dispositivo em uma rede diretamente conectada. Para efeito desta pesquisa, tais aspectos não foram considerados. O trabalho em Sarrar et al. (2010), porém, aborda estas questões e sugere métodos que permitiriam manter nas tabelas de encaminhamento apenas aqueles prefixos com maior chance de serem contatados, ao custo de se ter de instalar entradas de forma reativa, quando uma rede não prevista fosse contatada. Por outro lado, tal solução é orientada a arquiteturas virtuais de roteamento IP tradicionais, e seria preciso contextualizar sua conclusões à luz do trabalho aqui desenvolvido.

6.4.4 Avaliação comparativa

Por fim, é possível estabelecer uma comparação entre os resultados verificados para plataforma RouteFlow e aqueles apresentados para a plataforma RCP (CAESAR et al., 2005), que também preconiza um modelo centralizado de cálculo e disseminação de rotas.

A RCP opera através de processos de roteamento que são integrados às topologias BGP e IGP de um AS. Por isto, é um pré-requisito para sua operação que se ajuste os

custos de enlaces conectados às interfaces locais. De outra forma, seria possível que a própria RCP viesse a integrar um caminho de menor custo dentro do AS – um papel que ela não se propõe a atender. Em contrapartida, a plataforma RouteFlow pode simplificar a operação do domínio. Para o caso específico do serviço de roteamento implementado, não é necessário qualquer IGP, por exemplo.

A dispensa de um IGP também pode representar uma vantagem quando se consideram os problemas relacionados à atrasos de convergência discutidos na literatura, e aos quais a RCP se mantém sujeita (CAESAR et al., 2005).

Uma vez que conta com as informações de alcançabilidade entre nós providas pelo protocolo de roteamento intra-domínio, a RCP é capaz de reagir nativamente a eventos de desconexão e reconexão de enlaces. Os menores caminhos entre os diversos roteadores da infraestrutura controlada também são calculados via IGP, cabendo à plataforma apenas confrontar estes dados com os prefixos obtidos via BGP para derivar escolhas ótimas de encaminhamento para todos os roteadores do AS. Esta seria a maneira encontrada de implementar o roteamento *hot-potato*. Escolhas ótimas e reação a eventos de conectividade não fazem parte do serviço RFAgg no momento. Por outro lado, a plataforma RouteFlow construída oferece suporte a este tipo de construção.

Há que se considerar ainda que a derivação de escolhas ótimas pode ser um processo computacionalmente custoso, e os próprios autores da RCP relatam em seu trabalho que mudanças contínuas na conectividade poderiam fazê-lo representar um desafio significativo. Além disto, sugerem que a plataforma poderia facilitar a transformação da lógica de roteamento vigente, movendo-a justamente para além de comportamentos do tipo *hot-potato*.

Uma implementação da RCP poderia ser integrada ao modelo de operação da ferramenta RouteFlow original, a partir da execução de múltiplas MVs em um mapeamento "1:1", dotadas de múltiplos processos de roteamento, e às quais fosse estabelecida uma relação de adjacência para com o plano de controle. Aquela versão da arquitetura prosse-

guiria coletando entradas de FIBs, que seriam então ajustadas pelo servidor de rotas RCP e transmitindas ao plano de controle para instalação nos dispositivos correspondentes.

Em contrapartida, as inovações incorporadas na plataforma de roteamento como serviço aqui estabelecida poderiam permitir a implementação da própria RCP como um *serviço de roteamento*. E ao passo em que informações de alcançabilidade fossem coletadas e automaticamente submetidas à lógica do serviço, requisitos como uma adjacência IGP entre a plataforma de cálculo e cada roteador do domínio poderiam se tornar dispensáveis. Nesta hipótese, também seria dispensável qualquer ajuste especial dos protocolos de roteamento. Adicionalmente, ainda outras lógicas de processamento poderiam ser combinadas, com vistas à realização da política de negócio vigente no AS controlado.

7 Conclusão

Esta pesquisa partiu do problema de gerenciamento de um sistema autônomo Internet, notadamente a dificuldade em se realizar consistentemente a política de negócio de um AS Internet. Conforme levantado, tal condição deriva da necessidade de se realizá-la a partir da conciliação de parâmetros dos protocolos de roteamento e de roteadores que têm, cada qual, configurações distintas e visões parciais da topologia da rede. Este desafio ainda é tangenciado pelos efeitos secundários do acoplamento entre decisões de roteamento, disseminação das mesmas e divulgação do estado de enlaces, com forte contribuição do uso de IGPs.

Foi proposta uma solução que apóia-se no paradigma de virtualização de redes e no modelo SDN, que preconiza construção de redes a partir de elementos mais flexíveis que os atuais, com funções melhor definidas e isoladas. A abordagem SDN que orienta sua concepção é a OpenFlow, que delega aos comutadores de uma rede uma atuação restrita ao plano de dados, enquanto o plano de controle fica a cargo de um elemento controlador. O referencial para seu modelo de operação veio do grupo de ferramentas classificado como arquiteturas virtuais de roteamento IP, que acompanham a lógica de separação de planos e se baseiam na observação do comportamento de topologias de rede virtuais para a instrução dos componentes do plano de dados.

Segundo o raciocínio apresentado, poderia-se explorar esta função de disseminar decisões de roteamento. Cogitou-se que uma arquitetura virtual de roteamento poderia servir de plataforma de execução a componentes de *software*, capazes de operar sobre as múltiplas decisões recebidas da topologia virtual antes que estas fossem efetivamente tra-

duzidas em instruções de encaminhamento. A atuação de cada componente, aqui nomeado *serviço*, poderia produzir um comportamento conforme à política de negócio vigente, a partir de uma visão global da rede e suas decisões. Assim, esperava-se obter efeitos práticos que antes só poderiam ser obtidos através de cuidadosa parametrização de um protocolo de roteamento.

Uma vez que tal elemento materializaria o objetivo geral deste trabalho, a hipótese de pesquisa estabelecida foi a viabilidade de se obter um artefato capaz de produzir os resultados pretendidos a partir da modificação de uma arquitetura virtual de roteamento IP. Optou-se por testar esta hipótese no contexto da arquitetura RouteFlow, classificando o resultado deste processo como uma *plataforma de roteamento como serviço*.

Foram então elencadas características a serem incorporadas ao sistema que permitiriam concretizar aquela idéia: (i) uma ferramenta de descrição da política de roteamento desejada (na forma dos serviços que a compunham), (ii) uma base de informações da topologia sobre a qual a política deveria operar e (iii) instrumentos do plano de controle capazes de reproduzir com a maior eficiência e eficácia possíveis o conhecimento obtido a partir das duas características anteriores.

A especificação de instrumentos virtuais foi o primeiro resultado apresentado, obtido a partir de um trabalho em duas etapas. A primeira foi baseada na análise das características funcionais de uma seleção de ferramentas do estado da arte, servindo também como uma triagem das opções disponíveis. Em seguida, um plano de experimentação foi detalhado e posto em prática para aferir métricas de desempenho que embasaram a análise quantitativa do desempenho de cada ferramenta aprovada na primeira etapa.

Antes mesmo que o projeto de pesquisa relatado por esta dissertação chegasse ao fim, os instrumentos especificados foram incorporados à arquitetura RouteFlow por seus desenvolvedores.

Em relação às outras duas características estabelecidas, uma gramática que as atende simultâneamente foi construída. Para tal, foi necessário elaborar modelos de informação

concernentes aos elementos de uma rede SDN dedicada a roteamento. Após a conclusão de que tais modelos reproduziam o conhecimento a respeito destas redes, foi criado um conjunto de símbolos e regras gramaticais para a expressão dos mesmos. Em seguida, uma arquitetura que permite armazenar tal expressão em uma memória sistêmica foi apresentada.

Por fim, buscou-se construir, apoiado sobre a arquitetura modificada, um serviço que implementasse uma lógica de encaminhamento particular e capaz de usar o roteamento BGP como uma entrada ao invés de tê-lo como um fim. Assim, sua operação atestaria a validade das intervenções realizadas e, por conseguinte, da hipótese de pesquisa.

O comportamento do serviço foi averiguado através de sua aplicação a um caso hipotético, mas que reflete o cenário encontrado em *backbones* de produção do mundo real. Constatou-se que ele desempenhava seu papel conforme esperado. Frente a esta constatação, uma cuidadosa análise dos resultados obtidos foi realizada.

As seções a seguir tratam justamente a respeito da formalização, tanto das contribuições e limitações observadas, quanto das oportunidades de pesquisa abertas a partir desta dissertação.

7.1 Contribuições

Além do conhecimento absorvido durante a pesquisa, alguns dos benefícios obtidos com esta dissertação foram:

- Um método para o estabelecimento de uma plataforma de roteamento como serviço, que permite a livre implementação da política de negócio de um AS sobre a arquitetura de roteamento já existente na Internet.
- A gramática de configuração que permite descrever os elementos da infraestrutura subjacente a uma rede virtual no contexto SDN, e que é consumida ao longo da execução da plataforma elaborada no item anterior.

- A concepção de um novo modelo de roteamento, o serviço RFAgg, que possibilita a operação unificada de diferentes dispositivos de encaminhamento organizados em uma rede de malha completa.

- Um estudo aprofundado sobre a aderência de diferentes ferramentas de virtualização às funções próprias a elas no contexto das arquiteturas virtuais de roteamento IP.

- O desenvolvimento das propostas e conceitos apresentados no código da arquitetura RouteFlow, grande parte dos quais foram incorporados ao código oficial da ferramenta.

Na Figura 32 também pode-se visualizar as publicações derivadas desta pesquisa. As linhas tracejadas indicam uma relação entre as idéias apresentadas em cada capítulo e o foco dos artigos produzidos. Como se pode ver, os Capítulos 5 e 6 foram reservados para submissões futuras.

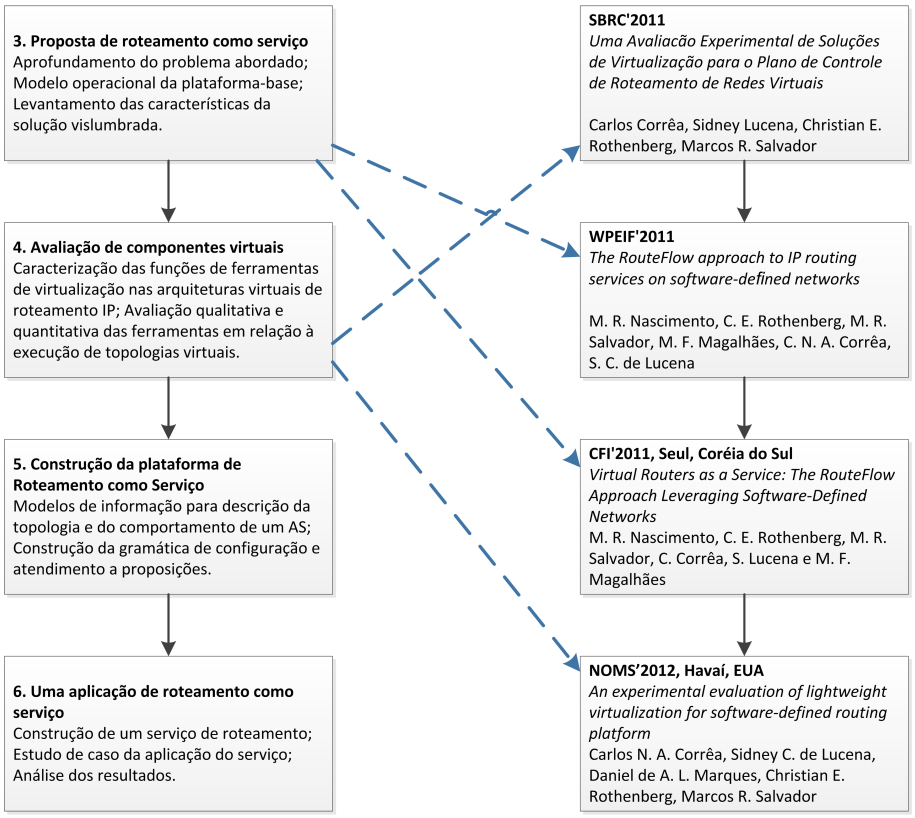


Figura 32: Publicações realizadas a partir dos resultados desta pesquisa.

7.2 Limitações

As limitações identificadas no âmbito do trabalho de pesquisa são apresentadas e discutidas a seguir.

- Não foi estabelecida uma estrutura formal de disparo e tratamento de eventos, tanto do ponto de vista das decisões de roteamento nas MVs, quanto oriundos do plano de dados. Isto faz com que o desenvolvimento de um novo serviço para a plataforma RouteFlow tenha que ser integrado ao código da própria ferramenta, que certamente se beneficiaria de uma solução de melhor engenharia.

- A instância de serviço de roteamento desenvolvida exige a disposição dos elementos de encaminhamento do domínio controlado em uma malha completa em nível de enlaces. Tal configuração pode não ser possível em um *backbone* de produção do mundo real.

- A estratégia de "pós-processamento" de decisões, no caso específico do protocolo BGP, pode limitar as opções de caminhos disponíveis para avaliação do serviço de roteamento. É possível encontrar na literatura implementações do BGP que suportam a escolha de múltiplas rotas de mesmo custo, mas estas não foram consideradas para efeito desta pesquisa.

- A avaliação de ferramentas de virtualização realizada no Capítulo 4 não se aprofunda na investigação das razões que levam ao desempenho observado. Uma investigação do tipo permitiria caracterizar melhor a aderência de cada mecanismo à construção de redes virtuais de uso geral.

7.3 Sugestões para futuras pesquisas

Enquanto as contribuições apresentadas foram alcançadas, outras frentes de pesquisa se abrem após a conclusão deste trabalho. Dentre algumas destas possibilidades, sugerimos:

- A construção de uma estrutura de código plugável, que permitiria que cada serviço de roteamento incorporado à plataforma RouteFlow registrasse tratadores de eventos referentes a cada possível evento dos planos de dados e controle.

- A incorporação do comportamento *hot-potato* ao serviço RFAgg, que poderia ser realizada tanto por meio de um cálculo preliminar dos menores caminhos entre vértices da rede, quanto pela combinação das decisões BGP da MV agregadora a um plano de controle secundário, baseado em um IGP.

- Um estudo da viabilidade de se atender, simultâneamente, à política de negócio estipulada para um AS, à construção de caminhos ótimos entre seus elementos e ao comportamento *hot-potato*.

- A implementação de novos serviços para a plataforma RouteFlow, do que poderia resultar a aplicação de lógicas inteiramente novas ao encaminhamento de tráfego de dados na Internet.

7.4 Considerações finais

Conserva-se alguma semelhança entre o roteamento como serviço aqui proposto e o conceito de Plataforma como Serviço nas redes de computadores apresentado em Keller e Rexford (2010). As idéias só se diferenciam pelo fato de que aquele trabalho procura caracterizar as ofertas que seriam inerentes a um sistema de roteamento operado da perspectiva dos *clientes de um AS* e não pelo próprio Sistema Autônomo. Desta forma, uma comparação entre alguns dos requisitos lá estabelecidos, e dos resultados aqui produzidos, também pode contribuir para o caráter conclusivo desta seção.

As ofertas estipuladas por Keller e Rexford (2010) iniciam-se pelo *roteamento controlado pelo cliente*, na forma de uma abstração única do serviço de roteamento com a qual o usuário de uma infraestrutura de redes possa interagir. Considera-se que isto foi aqui materializado, ao se construir a gramática de configuração RouteFlow e abrir-se a

possibilidade de se desenvolver e especificar conjuntos arbitrários de serviços com seus respectivos parâmetros.

Também propõe-se que seja necessária uma relação interativa entre o desempenho da política de roteamento de um cliente e a plataforma de controle de encaminhamento. Os serviços da plataforma RouteFlow são uma forma de especificação de lógica de transporte. Enquanto se pode argumentar que estes dependem de uma implementação em linguagem C++, mesmo o próprio trabalho de Keller e Rexford (2010) sugere que a interação pretendida seja implementada pelo fornecimento de um arquivo executável, por parte do cliente, que implemente o comportamento pretendido. Aquela pesquisa chega a sugerir que clientes devam poder utilizar lógicas de processamento geral, o que se aproxima ainda mais do que aqui foi produzido. Outrossim, o estudo de caso aqui realizado baseia-se exclusivamente em uma infraestrutura virtual, inerente às operações hospedadas. Pode-se conceber múltiplas execuções da RouteFlow, cada qual controlando um universo distinto de elementos de encaminhamento virtuais. Nesta extrapolação, permitir que os vários clientes operando cada uma destas topologias gerencie sua execução é algo factível. Mesmo na eventualidade de o instrumento de uma linguagem de configuração seja demonstrado como incapaz de alcançar tal escala, os modelos fornecidos são diretamente conversíveis em estruturas de bancos de dados.

Finalmente, o serviço desenvolvido e aplicado no estudo de caso do Capítulo 6 vai além de mera prova de conceito, podendo ser útil em cenários reais, similares ao avaliado. Mas, naturalmente, não cobre todo o espaço de soluções e desafios persistentes no campo do encaminhamento de tráfego na Internet.

É a plataforma sobre a qual o serviço foi implementado, constituída a partir da validação da hipótese de pesquisa, que pode tornar a execução de protocolos de roteamento mais gerenciável. Neste sentido, este trabalho é bem-sucedido em oferecer um instrumento que se aplica ao tratamento dos problemas estudados. Ainda que outras abordagens permaneçam válidas (e novas ainda possam ser tentadas), um serviço apoiado sobre as bases constituídas pode implementar uma ou várias das características existentes em soluções

relacionadas e aqui discutidas, as quais, individualmente, também não oferecem um conjunto absoluto de respostas para os problemas atacados.

ANEXO A – Definição dos símbolos terminais da gramática de configuração

```
\%\{
\#include <string.h>
\#include "rfconf.hh"
\%\}

\%\%
(host)                                          return HOST;
(service)                                       return SERVICE;
(datapath)                                      return DATAPATH;
(lxc)                                           return VIRT_LXC;
(dpid)                                          return DPID;
(ports)                                         return PORTS;
(access)                                        return PORT_ACCESS;
(trunk)                                         return PORT_TRUNK;
(port)                                          return PORT;
(speed)                                         return SPEED;
(((0|128|192|224|240|248|252|254|255)\.
[0]\{1,3\}\\.[0]\{1,3\}\\.[0]\{1,3\})|(255\\.
(0|128|192|224|240|248|252|254|255)\\.
[0]\{1,3\}\\.[0]\{1,3\})|(255\.255\\.
(0|128|192|224|240|248|252|254|255)\\.
[0]\{1,3\})|(255\\.255\\.255\\.
(0|128|192|224|240|248|252|254|255)))        yylval=strdup(yytext); return NETMASK;
(0x[a-fA-F0-9]*)                                yylval=strdup(yytext); return DPATH_ID;
([0-9]+)                                        yylval=strdup(yytext); return NUMBER;
(([0-9]|[1-9][0-9]|1[0-9]\{2\}|2[0-4]
[0-9]|25[0-5])\\.)\{3\}([0-9]|[1-9][0-9]|
1[0-9]\{2\}|2[0-4][0-9]|25[0-5])               yylval=strdup(yytext); return IP;
(([a-zA-Z]|[a-zA-Z][a-zA-Z0-9\\-]*
[a-zA-Z0-9])\\.)*([A-Za-z]|[A-Za-z]
[A-Za-z0-9\\-]*[A-Za-z0-9])                     yylval=strdup(yytext); return HOSTNAME;
(\\/)                                           return SLASH;
(:)                                             return COLON;
(\\")                                           return QUOTE;
\\n                                             /* ignore newlines */
[\\t]+                                          /* ignore blanks */
\%\%
```

Referências

AHO, A. V.; SETHI, R.; ULLMAN, J. D. *Compilers: principles, techniques, and tools*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1986. ISBN 0-201-10088-6.

ALDERSON, D. et al. Understanding internet topology: principles, models, and validation. *IEEE/ACM Trans. Netw.*, IEEE Press, Piscataway, NJ, USA, v. 13, p. 1205–1218, December 2005. ISSN 1063-6692.

ALKMIM, G. P.; BATISTA, D. M.; FONSECA, N. L. S. da. Mapeamento de redes virtuais em substratos de rede. In: *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.: s.n.], 2011.

ALVES, N. et al. *Topologia e Modelagem Relacional da Internet Brasileira*. [S.l.], Setembro 2010.

ARMBRUST, M. et al. A view of cloud computing. *Commun. ACM*, ACM, New York, NY, USA, v. 53, p. 50–58, abr. 2010. ISSN 0001-0782.

BALLARD, J. R.; RAE, I.; AKELLA, A. Extensible and scalable network monitoring using opensafe. In: *Proceedings of the 2010 internet network management conference on Research on enterprise networking*. Berkeley, CA, USA: USENIX Association, 2010. (INM/WREN'10), p. 8–8.

BARHAM, P. et al. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, ACM, New York, NY, USA, v. 37, p. 164–177, October 2003. ISSN 0163-5980.

BENVENUTI, C. *Understanding Linux Network Internals*. [S.l.]: O'Reilly Media, Inc., 2005. ISBN 0596002556.

BHATIA, S. et al. *Hosting Virtual Networks on Commodity Hardware*. [S.l.], 2008.

BOLLA, R. et al. Drop: An open-source project towards distributed sw router architectures. In: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. [S.l.: s.n.], 2009. p. 1 –6. ISSN 1930-529X.

BOLTE, M. et al. Non-intrusive virtualization management using libvirt. In: *DATE '10*. [S.l.: s.n.], 2010. ISBN 978-3-9810801-6-2.

BOZAKOV, Z. An open router virtualization framework using a programmable forwarding plane. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 40, p. 439–440, August 2010. ISSN 0146-4833.

BRAGA, R.; MOTA, E.; PASSITO, A. Lightweight ddos flooding attack detection using nox/openflow. In: *Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2010. (LCN '10), p. 408–415. ISBN 978-1-4244-8387-7.

BUTLER, K. et al. *A Survey of BGP Security Issues and Solutions*. [S.l.], fev. 2004.

CAESAR, M. et al. Design and implementation of a routing control platform. In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*. Berkeley, CA, USA: USENIX Association, 2005. (NSDI'05), p. 15–28.

CAESAR, M. et al. Dynamic route recomputation considered harmful. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 40, p. 66–71, April 2010. ISSN 0146-4833.

CHOWDHURY, N. M. K.; BOUTABA, R. A survey of network virtualization. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 54, p. 862–876, April 2010. ISSN 1389-1286.

CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: McGraw-Hill Higher Education, 2001. ISBN 0070131511.

DORIA, A. et al. *General Switch Management Protocol (GSMP) V3*. [S.l.]: IETF, jun. 2002. RFC 3292 (Proposed Standard). (Request for Comments, 3292).

DRAGON TEAM. *Virtual Label Switching Router Implementation Guide*. Abril 2008. Http://dragon.east.isi.edu/twiki/pub/DRAGON/VLSR/dragon-vlsr-implement-v2.1b.pdf.

DUBE, R. A comparison of scaling techniques for bgp. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 29, p. 44–46, July 1999. ISSN 0146-4833.

DUBE, R.; SCUDDER, J. G. *Route Reflection Considered Harmful*. May 1999. Internet-Draft (work in progress), draft-dube-route-reflection-harmful-00.

EGI, N. et al. Evaluating xen for router virtualization. In: *16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007.* [S.l.: s.n.], 2007. p. 1256 – 1261.

EGI, N. et al. Towards high performance virtual routers on commodity hardware. In: *Proceedings of the 2008 ACM CoNEXT Conference*. New York, NY, USA: ACM, 2008. (CoNEXT '08), p. 20:1–20:12. ISBN 978-1-60558-210-8.

FERNANDES, N. et al. Virtual networks: isolation, performance, and trends. *Annals of Telecommunications*, Springer Paris, p. 1–17, October 2010. ISSN 0003-4347.

FILIP, O. et al. *BIRD Internet Routing Daemon*. Janeiro 2012. Http://bird.network.cz/.

FOSTER, N. et al. Frenetic: a network programming language. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 46, p. 279–291, set. 2011. ISSN 0362-1340.

FUNDAÇÃO CPQD. *Repositório público do projeto RouteFlow*. Janeiro 2012. Https://github.com/CPqD/RouteFlow.

GOOGLE. *Open source changes by Google to the Quagga routing suite*. Janeiro 2012. Https://code.google.com/p/google-quagga/.

GRIFFIN, T. G.; WILFONG, G. On the correctness of ibgp configuration. In: *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2002. (SIGCOMM '02), p. 17–29. ISBN 1-58113-570-X.

GUDE, N. et al. Nox: towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, p. 105–110, July 2008. ISSN 0146-4833.

HABIB, I. Virtualization with kvm. *Linux J.*, Specialized Systems Consultants, Inc., Seattle, WA, USA, v. 2008, February 2008. ISSN 1075-3583.

HANDIGOL, N. et al. Aster*x: Load-balancing as a network primitive. In: *ACLD '10: Architectural Concerns in Large Datacenters*. [S.l.: s.n.], 2010.

HANDLEY, M.; HODSON, O.; KOHLER, E. Xorp: an open platform for network research. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 33, p. 53–57, January 2003. ISSN 0146-4833.

INTERNET2 CONSORTIUM. *Map of Internet2 Network Connectors*. Janeiro 2012. Http://www.internet2.edu/pubs/networkmap-connectors-participants.pdf.

JUNG, C. F. *Metodologia científica e tecnológica*. 2003. Http://www.jung.pro.br/.

KELLER, E.; REXFORD, J. The "platform as a service"model for networking. In: *Proceedings of the 2010 internet network management conference on Research on enterprise networking*. Berkeley, CA, USA: USENIX Association, 2010. (INM/WREN'10), p. 4–4.

KHOSRAVI, H.; ANDERSON, T. *Requirements for Separation of IP Control and Forwarding*. [S.l.]: IETF, nov. 2003. RFC 3654 (Informational). (Request for Comments, 3654).

KRASNYANSKY, M. *Universal TUN/TAP driver*. 2005. `http://vtun.sourceforge.net/tun/index.html`.

KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach*. 5th. ed. USA: Addison-Wesley Publishing Company, 2009. ISBN 0136079679, 9780136079675.

LAFORE, R. *Object-oriented programming in C++*. [S.l.]: Sams, 2002. (Kaleidoscope Series). ISBN 9780672323089.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: rapid prototyping for software-defined networks. In: *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*. New York, NY, USA: ACM, 2010. (Hotnets '10), p. 19:1–19:6. ISBN 978-1-4503-0409-2.

LAVERICK, M. *VMware vSphere 4 Implementation*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 2010. ISBN 0071664521, 9780071664523.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, p. 69–74, March 2008. ISSN 0146-4833.

NASCIMENTO, M. R. et al. Routeflow: Roteamento commodity sobre redes programáveis. In: *Anais do XXIX Simpósio Brasileiro de Redes de Computadores*. [S.l.: s.n.], 2011a.

NASCIMENTO, M. R. et al. Quagflow: partnering quagga with openflow. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 40, p. 441–442, August 2010. ISSN 0146-4833.

NASCIMENTO, M. R. et al. Virtual routers as a service: the routeflow approach leveraging software-defined networks. In: *Proceedings of the 6th International Conference on Future Internet Technologies*. New York, NY, USA: ACM, 2011b. (CFI '11), p. 34–37. ISBN 978-1-4503-0821-2.

NORTON, W. B. Internet service providers and peering. In: *Proceedings of NANOG 19*. Albuquerque, New Mexico: [s.n.], 2000.

PAPAZOGLOU, M. P. et al. Service-oriented computing: State of the art and research challenges. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 40, p. 38–45, November 2007. ISSN 0018-9162.

PARK, J. H. et al. *BGP Route Reflection Revisited*. [S.l.], June 2010.

PFAFF, B. et al. *OpenFlow Switch Specification*. Fevereiro 2011. Http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf.

PFAFF, B. et al. Extending networking into the virtualization layer. In: *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*. [S.l.: s.n.], 2009.

PISA, P. et al. Openflow and xen-based virtual network migration. In: PONT, A.; PUJOLLE, G.; RAGHAVAN, S. (Ed.). *Communications: Wireless in Developing Countries and Networks of the Future*. [S.l.]: Springer Boston, 2010, (IFIP Advances in Information and Communication Technology, v. 327). p. 170–181.

PROJETO QEMU. *QEMU Documentation*. Janeiro 2012. Http://en.wikibooks.org/wiki/QEMU.

RNP. *Rede RNP - Mapa do backbone*. Janeiro 2012. Http://www.rnp.br/backbone/index.php.

SALIM, J. et al. *Linux Netlink as an IP Services Protocol*. United States: RFC Editor, 2003.

SARRAR, N. et al. Fibium – towards hardware accelerated software routers. In: *EuroView 2010 (poster session)*. [S.l.: s.n.], 2010. Poster.

SCUDDER, J. G.; DUBE, R. Bgp scaling techniques revisited. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 29, p. 22–23, October 1999. ISSN 0146-4833.

SHARAFAT, A. R. et al. Mpls-te and mpls vpns with openflow. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 41, p. 452–453, ago. 2011. ISSN 0146-4833.

SHERWOOD, R. et al. Carving research slices out of your production networks with openflow. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 40, p. 129–130, January 2010. ISSN 0146-4833.

STANFORD OPENFLOW TEAM. *OpenFlow 1.2 proposals*. Janeiro 2012. Http://www.openflow.org/wk/index.php/OpenFlow_1_2_proposal.

STANFORD, U. of. *Beacon: a Java-based OpenFlow Control Platform*. 2010. `http://www.beaconcontroller.net/`.

STEVENS, R. W.; RAGO, S. A. *Advanced Programming in the UNIX(R) Environment (2nd Edition)*. [S.l.]: Addison-Wesley Professional, 2005. ISBN 0201433079.

TEIXEIRA, R.; AGARWAL, S.; REXFORD, J. Bgp routing changes: merging views from two isps. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 35, p. 79–82, October 2005. ISSN 0146-4833.

TEIXEIRA, R. et al. Dynamics of hot-potato routing in ip networks. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 32, p. 307–319, June 2004. ISSN 0163-5999.

UDUGAMA, A. Manipulating the network environment using rtnetlink. *Linux J.*, Belltown Media, Houston, TX, v. 2006, p. 7–, maio 2006. ISSN 1075-3583.

URIARTE, Y. *Zebra Hacking How-To*. Fevereiro 2001. Http://www.quagga.net/zhh.html.

W3C CONSORTIUM. *XQuery 1.0: An XML Query Language (Second Edition)*. Janeiro 2012. Http://www.w3.org/TR/xquery/#EBNFNotation.

WANG, R.; BUTNARIU, D.; REXFORD, J. Openflow-based server load balancing gone wild. In: *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*. Berkeley, CA, USA: USENIX Association, 2011. (Hot-ICE'11), p. 12–12.

WANG, Y. et al. Virtual routers on the move: live router migration as a network-management primitive. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, p. 231–242, August 2008. ISSN 0146-4833.

XIAO, L.; WANG, J.; NAHRSTED, K. *Optimizing iBGP Route Reflection Network*. May 2003. IEEE International Conference on Communication.

XIAO, L.; WANG, J.; NAHRSTEDT, K. Reliability-aware ibgp route reflection topology design. In: *Proceedings of the 11th IEEE International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2003. (ICNP '03), p. 180–. ISBN 0-7695-2024-3.

ZHANG-SHEN, R.; WANG, Y.; REXFORD, J. *Atomic Routing Theory: Making an AS Route Like a Single Node*. [S.l.], July 2008.