Redes Virtuais de Data Center Mapeadas como Serviço em Redes Definidas por Software

Raphael Vicente Rosa, Christian Esteve Rothenberg, Edmundo R. M. Madeira

[1] Universidade Estadual de Campinas (UNICAMP) – Campinas, SP – Brazil

raphaelvrosa@lrc.ic.unicamp.br, chesteve@dca.fee.unicamp.br, edmundo@ic.unicamp.br

Abstract. *Based on Software Defined Network paradigm we apply the Network-as-a-Service model to build a data center network architecture well-suited to the problem of virtual networks embedding. This task is performed by the algorithm proposed in this paper, which is based on aggregated information from a virtual routing plane using the BGP protocol and a physical network topology based on OpenFlow 1.3 devices. The experimental evaluation shows that the proposed algorithm performs efficient load balancing on the data center network and altogether yields better utilization of the physical resources.*

Resumo. *Com base nos conceitos de Redes Definidas por Software, aplicamos o modelo de Rede como Serviço na construção de uma arquitetura para dar suporte a alocação de redes virtuais de data center. Esta tarefa é realizada pelo algoritmo proposto neste trabalho, o qual tem como base a agregação de informações de roteamento em um plano virtual definido pelo protocolo BGP e de uma topologia física de rede com suporte a OpenFlow 1.3. Com base em experimentos realizados, mostramos que o algoritmo proposto realiza com eficácia o balanceamento de carga na rede permitindo melhor aproveitamento de recursos da topologia física de rede de data centers.*

1. Introdução

Provedores de infraestrutura (Infrastructure Providers - InPs) têm virtualizado os data centers que possuem criando uma diversidade de recursos (ex: máquinas virtuais, switches/roteadores virtuais) que são utilizados pelos próprios InPs e por diversos provedores de serviços (Service Providers - SP). Um data center virtualizado pode fornecer recursos compartilhados para diferentes SPs e seus respectivos data centers virtuais (Virtual Data Centers - VDCs) [Bari et al. 2013]. Tal modelo de negócio requer a virtualização de servidores e de redes de data center (Data Center Networks - DCNs). A repartição e uso eficiente de recursos de rede, como largura de banda, ainda é um desafio que diversas propostas de arquiteturas de DCNs vem tentando solucionar [Verdi et al. 2010].

Um SP compartilhando uma infraestrutura de DCN pode possuir em suas redes virtuais requisitos como isolamento de recursos, gerenciamento flexível de alocações de tráfego, tolerância a falhas, balanceamento de carga e implantação de novas aplicações. Estes, muitas vezes necessitam ser alterados dinamicamente conforme planos de serviço oferecidos por SPs. Nesse sentido, soluções tradicionais, atreladas às pilhas de protocolos TCP/IP (ex: VLANs), possuem diversas limitações tais como: não isolamento de desempenho; riscos de segurança; difícil implementação de novas aplicações; flexibilidade limitada de gerenciamento; e ausência de suporte a inovações de rede [Bari et al. 2013].

Seguindo o paradigma de Redes Definidas por Software (Software Defined Networking - SDN), neste trabalho propomos utilizar o conceito de Rede como Serviço (Network-as-a-Service - NaaS), como em [Keller and Rexford 2010], aplicado à construção de redes de data center virtualizadas. Propomos uma arquitetura que possibilita o tratamento de alocações dinâmicas de redes virtualizadas de data center seguindo critérios de largura de banda e resiliência. Consideramos um cenário onde um InP possua a necessidade de compartilhar sua infraestrutura física de data center para diversos SPs os quais já possuam suas demandas de alocações de máquinas virtuais (Virtual Machines - VMs) bem definidas.

Utilizando a plataforma RouteFlow [Nascimento et al. 2011], definimos um plano virtual utilizando o protocolo de roteamento BGP com suporte a múltiplos caminhos, considerando as premissas de [Lapukhov et al. 2013] para operar em DCNs com topologia de rede folded-Clos. No plano de dados, utilizamos uma infraestrutura física com suporte a versão 1.3 do padrão OpenFlow [ONF 2012]. E no plano de controle da plataforma, construímos serviços que agregam informações dos planos virtual e de dados, e logo, possibilitam que estas sejam utilizadas em conjunto para tratar o mapeamento de redes virtuais. O algoritmo proposto neste trabalho tem como tarefa principal alocar largura de banda em enlaces da topologia física do InP por meio do mapeamento de rotas da topologia virtual do InP para a criação de topologias virtualizadas para cada SP. Políticas, representando demandas de SPs, definem requisitos de endereçamento, largura de banda e resiliência, os quais são utilizadas pelo algoritmo de mapeamento para construir redes virtuais por meio da configuração de rotas na topologia física de rede do data center. Tais rotas são definidas para prover isolamento de recursos e suporte às funcionalidades que o padrão OpenFlow 1.3 oferece, como *group tables* e *meter tables* [ONF 2012].

A abordagem de alocação de redes virtuais proposta neste artigo avalia o algoritmo proposto para prover balanceamento de carga a DCNs e o compara a propostas existentes na literatura. Neste sentido, este trabalho se distingue dos demais pelos seguintes aspectos: (*i*) propõe a configuração de uma topologia virtual a qual utiliza o protocolo BGP de forma eficiente para o roteamento em data centers com topologia de rede folded-Clos; (*ii*) estende a plataforma RouteFlow para utilizar o padrão OpenFlow 1.3 e prover suporte a diferentes aplicações de políticas de rede, tais como reserva de largura de banda e rotas por múltiplos caminhos; (*iii*) e define um algoritmo eficaz de mapeamento de redes virtuais que propicia garantias de resiliência e largura de banda.

A estrutura deste artigo segue da seguinte forma, na seção 2 são apresentados os conceitos básicos relacionados a este trabalho. A seção 3 mostra a definição da arquitetura montada assim como a descrição de seus principais elementos. Na seção 4 apresentamos o algoritmo proposto neste trabalho. A seção 5 apresenta os experimentos realizados, os dados obtidos e uma discussão sobre a arquitetura e o algoritmo proposto. E a seção 6 conclui o artigo abordando trabalhos futuros a serem feitos.

2. Conceitos Básicos

Nesta seção descrevemos os principais componentes em uma DCN virtualizada, as propostas existentes na literatura sobre este tópico, assim como os elementos principais utilizados na construção da arquitetura proposta neste trabalho.

2.1. Virtualização de Redes de Data Center

Um data center é formado por servidores, equipamento de rede (ex: switches, roteadores, cabos) e sistemas de distribuição de energia e de resfriamento. A rede de um data center é definida pela topologia (ex: BCube, Fat-tree, Clos) em que seus switches/roteadores encontram-se conectados e pelos protocolos de rede utilizados na comunicação entre seus componentes (ex: IP, Ethernet). Uma DCN geralmente possui a seguinte configuração topológica. Servidores são agregados em racks e se conectam a um switch Top-of-Rack (ToR). Este se conecta a switches End-of-Row (EoR), que não se interconectam, e são utilizados como intermediários em conexões de ToRs com switches Core [Verdi et al. 2010]. Um VDC se define como um conjunto de recursos virtualizados (ex: VMs, switches/roteadores virtuais) conectados por enlaces virtuais. Neste cenário, uma rede virtual é definida como um conjunto de recursos de rede virtualizados conectados por enlaces virtuais. Ou seja, a virtualização de redes, similar a virtualização de servidores, permite a criação de múltiplas redes virtuais sobre um mesmo substrato de rede físico as quais são implementadas e gerenciadas de forma independente [Bari et al. 2013].

Em trabalhos recentes sobre virtualização de redes de data center, os seguintes desafios são abordados: tecnologias de virtualização; topologias de rede; isolamento de desempenho; escalabilidade; tolerância a falhas; e técnicas de encaminhamento de pacotes. Nessas pesquisas, recursos de DCNs são utilizados de duas formas, por competição ou compartilhamento. No primeiro caso, destacam-se as propostas Seawall [Shieh et al. 2010], Netshare [Lam et al. 2012] e FairCloud [Popa et al. 2011], as quais propõem disputa justa de recursos de rede por multiplexação estatística e requisitos mínimos de largura de banda a VMs, mas não fornecem garantias determinísticas de recursos de rede (ex: latência, largura de banda). Já no segundo caso, as principais propostas são Gatekeeper [Rodrigues et al. 2011], SecondNet [Guo et al. 2010], Oktopus [Ballani et al. 2011] e Proteus [Xie et al. 2012]. Estas realizam a alocação de garantias mínimas de largura de banda a conjuntos de VMs, os quais são definidos de diferentes formas, tais como heurísticas e análises temporais de padrões de comunicação entre VMs.

2.2. RouteFlow

RouteFlow [Nascimento et al. 2011] é uma arquitetura de roteamento que segue o paradigma de SDNs ao centralizar logicamente o controle da rede, unificar o estado de informação e desacoplar as lógicas de encaminhamento e configuração de equipamentos de rede. A plataforma RouteFlow permite oferecer serviços de roteamento IP definidos em um plano virtual de controle e mapeados aos recursos de uma infraestrutura física de rede com suporte ao padrão OpenFlow permitindo a oferta da plataforma como serviço à rede pelo mapeamento flexível de recursos de uma topologia virtual sobre uma infraestrutura física de rede a qual pode ser distribuída e compartilhada. Constituída por três planos, virtual, físico e de controle, RouteFlow possui respectivamente em cada um destes planos seus seguintes componentes principais: rfclient, rfproxy e rfserver.

O plano virtual executa roteadores virtuais, interconectados por um switch com suporte a OpenFlow, definidos em nível de virtualização do sistema operacional Linux (Contêineres Linux - LXCs). Neles, a aplicação rfclient captura as rotas computadas por uma suíte de roteamento (ex: Quagga, XORP, BIRD) e as envia ao plano de controle. O plano físico é formado por switches com suporte a OpenFlow interligado ao(s) controlador(es) de rede OpenFlow, que executa(m) a aplicação rfproxy, responsável por realizar

a interface entre o plano de controle e o controlador de rede. No plano de controle, a aplicação rfserver armazena todo o estado de configuração e define o mapeamento entre topologias física e virtual. Ela também gerencia o estado do mapeamento físico/virtual contido em um banco de dados (ex: MongoDB) e realiza a troca e formatação de mensagens entre as aplicações rfclient e rfproxy, tais como rotas computadas em LXCs no plano virtual. Desde sua concepção, há mais de três anos [Nascimento et al. 2011], RouteFlow tem evoluído significativamente até o ponto de atualmente estar operando em pontos de troca de tráfego [Stringer et al. 2013].

3. Arquitetura Proposta

Com base na necessidade de uma arquitetura que desse suporte à criação e instanciação de redes virtuais em DCNs, vimos na plataforma RouteFlow as características que permitem o controle direto da rede, a unificação do estado das informações dos planos virtual e de dados, e o desacoplamento das lógicas de encaminhamento e configuração de equipamentos de rede. Também encontramos na proposta [Lapukhov et al. 2013] as características de configuração do protocolo BGP para o roteamento intra-AS em DCNs que foram ao encontro das funcionalidades da plataforma RouteFlow. Essa proposta define o uso do protocolo BGP com suporte a múltiplos caminhos sobre uma topologia de rede folded-Clos. Dentre suas vantagens podemos citar: design prático de roteamento para grandes data centers; protocolo de simples implementação com baixa complexidade de código e fácil suporte operacional; minimização do domínio de falha de equipamentos e protocolo de roteamento; diminuição de custos operacional e de capital.

Logo, utilizamos a plataforma RouteFlow para propor uma arquitetura para DCNs (vide Figura 2) com topologia virtual definida por roteamento com protocolo BGP, plano de dados com suporte a OpenFlow 1.3, e plano de controle com total gerenciamento sobre as topologia física e virtual do data center para dar suporte ao algoritmo de mapeamento de redes virtuais proposto neste trabalho. A arquitetura permite que *group tables* e *meter tables* sejam definidas pelo plano de controle para serem programadas pela aplicação rfproxy, de modo que fique transparente para ambos os planos quais os detalhes de implementação das funcionalidades de cada um. Nesta seção, descrevemos as modificações na plataforma RouteFlow feitas para sua adequação aos requisitos de mapeamento de elementos da infraestrutura física de um InP em uma topologia virtual para operação de uma DCN. Estas descrições estão separadas de acordo com os planos da arquitetura nas seguintes subseções: plano físico, plano virtual e plano de controle.

3.1. Plano Físico

Neste plano utilizamos uma topologia de rede folded-Clos contendo switches com suporte a OpenFlow 1.3. Em um controlador de rede com suporte a esta mesma versão de OpenFlow construímos a aplicação rfproxy. Esta, auxiliada pela aplicação de descoberta de topologia, captura eventos de adição e remoção de switches e enlaces na topologia física e os encaminha ao plano de controle para a definição da topologia física disponível aos mapeamentos. Funcionalidades do padrão OpenFlow 1.3 são utilizadas para dar suporte aos mapeamentos de redes virtuais, tais como a utilização de *group tables* para encaminhamento de dados semelhante ao mecanismo de roteamento ECMP (Equal-Cost Multipath), e reservas de largura de banda definidas por *meter tables*.

A programação de rotas de topologias virtuais de SPs na topologia física ocorre pela identificação única de uma rede de um SP por tags MPLS. Em switches Core e EoR, o tráfego é encaminhado somente por *matches* destas tags, definidas em rotas programadas na tabela 0. Já em ToRs, quatro tabelas são programadas. A tabela 0 trata *matches* em tags MPLS com tráfego destinado ao interior do rack e realiza a ação de retirada desta camada MPLS e o encaminhamento para a tabela 1. Nesta, o *match* ocorre por endereços de rede, e tem como ações a reescrita de endereços MAC nos pacotes e o encaminhamento para a próxima tabela. Na tabela 2, o *match* com os enredeços MAC de servidores gera a ação de encaminhamento de pacotes para as devidas portas em que eles estão conectados. Por fim, a tabela 3 trata o tráfego a ser encaminhado para fora do rack. Logo, esta tem como ação a marcação da tag MPLS relativa à rota de uma rede virtual de um SP, definida no plano de controle. Todas as rotas de uma política são programadas em switches com um tempo de *hard timeout* o qual define o tempo de permanência de uma topologia virtual no plano de dados. Este tempo é configurado pelo plano de controle, e nos switches do plano de dados, quando expirado, define a exclusão das rotas por ele definidas. Limitadores de largura de banda são programados em ToRs delimitando em racks o tráfego de entrada e saída por *meter tables* associadas respectivamente às regras das tabelas 2 e 3.

3.2. Plano Virtual

No plano virtual foi adicionada à aplicação rfclient a característica de leitura de rotas *multipath*. O roteamento em LXCs foi definido pela suíte de roteamento Quagga executando o protocolo BGP, configurado com as vantagens de uso intra-AS para DCNs em topologia folded-Clos [Lapukhov et al. 2013]. Como visto na Figura 1, realizamos o mapeamento das topologias base, física e virtual, as quais representam os planos de dados e virtual de um InP da seguinte forma: agregamos elementos que contém o mesmo ASN em uma camada da topologia física folded-Clos. Isto foi possível, pois elementos de uma mesma camada de switches da topologia folded-Clos não estão interconectados entre si, possuem rotas equidistantes a quaisquer outros elementos da topologia, e computam rotas com mesmo AS-PATH para todos os destinos da rede. Neste plano, também definimos que inicialmente todas as mensagens de controle são transferidas entre os planos virtual e de dados através do controlador de rede. Após mapeadas as topologias base, física e virtual, rotas são definidas no switch do plano virtual para que as mensagens de controle permaneçam na topologia virtual, não sobrecarregando o controlador de rede.

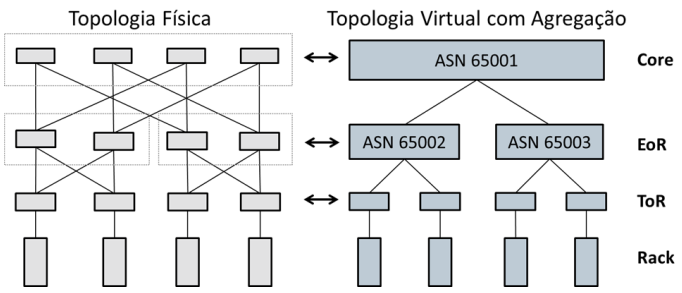


Figura 1. Mapeamento de topologias física e virtual com agregação

3.3. Plano de Controle

Neste plano é executada a aplicação rfserver, o banco de dados que mantém o estado do mapeamento entre os planos físico e virtual, e os componentes desenvolvidos neste trabalho (vide Figura 3): **Recurso**; **Politica**; **Configuração**; **Alocador**; e **Escalonador**.

Recurso realiza o armazenamento e gerenciamento de informações dos planos virtual e de dados. Para isso faz a criação de classes as quais representam os recursos de cada plano. As classes *TopologiaFísica* e *TopologiaVirtual* tem como atributos componentes que representam respectivamente os planos físico (ex: switches, enlaces) e virtual (ex: LXCs, interfaces, rotas). Estas duas classes são herdeiras da classe *Topologia* a qual constrói grafos utilizados para representá-las. Neste componente também define-se a classe *Topologias* a qual possui funções de instanciação de topologias físicas e virtuais, gerenciamento do mapeamento entre elas, preenchimento de suas configurações tais como rotas, enlaces e switches ou LXCs, além de funções de análise das configurações e estados das topologias física e virtuais. Além disso, a classe *Servidores* trata toda a representação de VMs em racks de servidores.

Política é responsável pelo gerenciamento de demandas de alocação de redes virtuais com requisitos de largura de banda, resiliência e esquemas de endereçamento entre VMs e servidores que irão constituir a topologia virtual do VDC estabelecido a um SP. Ou seja, a partir da determinação do mapeamento de VMs em servidores interconectados a ToRs, previamente estipulada por SP e InP, uma política representa este mapeamento. Ela contém os endereços das VMs, servidores e ToRs desta demanda, assim como a matriz de tráfego entre eles. A topologia de rede virtual de uma política, criada pelo mapeamento desta no plano de dados pelo componente Configuração e representada pela instância de uma *TopologiaVirtual* do componente Recurso, é armazenada e associada a um identificador MPLS único desta política, utilizado na programação de rotas no plano de dados. Dentro deste componente a classe denominada Políticas realiza o cadastramento, gerenciamento e armazenamento das políticas criadas bem como de suas respectivas topologias virtuais construídas e mapeadas.

Configuração contém algoritmos de mapeamento que realizam tanto a associação entre topologias quanto o mapeamento de rotas da topologia base virtual para a criação de topologias virtuais. Além disso, dentro deste componente existem algoritmos que auxiliam nas funções principais de mapeamento, os quais lidam com modificações de recursos das topologias base para melhor atender as demandas de mapeamento.

Alocador realiza toda a interface de configuração de topologias virtuais no plano físico. Por meio dele são configuradas topologias virtuais, mapeamentos são feitos utilizando o componente Configuração e topologias virtuais são mapeadas na topologia física do InP com o envio de mensagens à aplicação rfproxy. Além disso, ele também realiza a configuração de rotas no switch virtual para o gerenciamento do tráfego de controle do plano virtual do InP.

Escalonador é o componente responsável por gerir todo o funcionamento dos componentes anteriores. Por meio dele, são criadas políticas, constituídas e configuradas topologias virtuais e físicas e onde demandas de mapeamento de topologias são definidas em políticas para serem alocadas e desalocadas dinamicamente. Ou seja, este componente realiza toda a interface de comunicação com a aplicação rfserver.

3.4. Execução da Arquitetura

O funcionamento da arquitetura proposta ocorre da seguinte forma. Os elementos da plataforma são inicializados nesta ordem: LXCs e aplicações rfclient; switch do plano virtual; banco de dados; aplicação rfserver; controlador de rede e aplicação rfproxy; switches da topologia física. A aplicação rfserver realiza a inicialização de todos os componentes descritos na subseção anterior, tratando-os como serviços da plataforma RouteFlow.

O componente Escalonador (vide Figura 3), após aguardar a convergência de rotas do plano virtual e a descoberta de topologia do plano físico, cria uma política padrão a qual é definida pelo mapeamento das topologias base, física e virtual. Neste primeiro momento, os objetos que representam estas topologias são instanciados para serem utilizados nos mapeamentos posteriores. Além disso, o objeto que representa servidores é criado para definir a instanciação de políticas, ou seja, o mapeamento de VMs em servidores e, consequentemente, em ToRs. Em seguida, três *threads* são iniciadas para realizarem as tarefas de: (*i*) criação de demandas de mapeamento (ex: geradas pelo OpenStack) e definição de políticas; (*ii*) alocação de políticas para criação de topologias virtuais; e (*iii*) desalocação de políticas conforme tempo de mapeamento atingido.

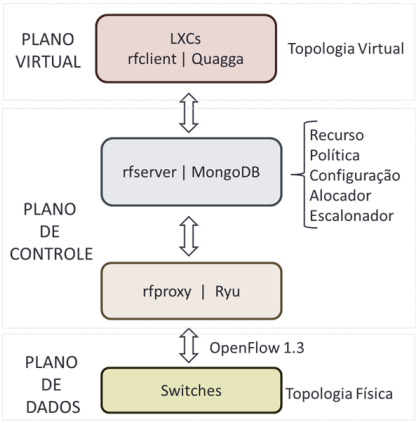


Figura 2. Arquitetura Proposta

Figura 3. Componentes do Plano de Controle

A primeira *thread*, com base em uma matriz de tráfego entre VMs, aloca estas em racks de servidores utilizando o objeto *Servidores*, criando uma matriz de tráfego entre ToRs. Estas informações são agregadas juntamente com requisitos de largura de banda, resiliência e endereçamento, definindo uma política que é colocada em uma fila para ser alocada. A *thread* de alocação verifica esta fila e realiza a alocação de políticas na topologia física base por meio do algoritmo de mapeamento proposto neste trabalho, e logo, cria a topologia virtual associada a esta política. Em função do tempo de alocação definido para cada política, a *thread* de desalocação retira o mapeamento de topologias virtuais da topologia física base.

4. Algoritmos do Plano de Controle

Existem dois algoritmos principais no plano de controle: (*i*) o *algoritmo de suporte* auxilia o gerenciamento de recursos da topologia física para que (*ii*) o *algoritmo de mapeamento* proposto neste trabalho realize a alocação de topologias virtuais de forma consistente.

4.1. Algoritmo de Suporte

Em cada um dos switches da topologia física base do plano de controle, dois atributos se destacam. O primeiro define uma tabela (*bw_port_table*) de porcentagem de largura de banda em cada porta de um switch. Cada vez que um enlace adjacente a um switch é adicionado a topologia física base ou uma rota é mapeada a este enlace, o percentual de largura de banda disponível neste enlace é calculado e armazenado na *bw_port_table* dos switches adjacentes, na forma {*porta adjacente a enlace : porcentagem de largura de banda disponível*}. O outro atributo diz respeito à tabela (*bw_table*) de porcentagem de largura de banda disponível para endereços de destino das rotas de um switch. Cada vez que uma rota é mapeada a um enlace, os switches adjacentes atualizam suas tabelas de largura de banda conforme o endereço de destino da rota mapeada, constituindo entradas na forma {*endereço de destino : [porta de destino de rota a endereço : porcentagem de largura de banda disponível para a rota]* }.

Algoritmo 1 : Executado para atualizar *bw_tables* de switches da topologia física base

Entrada: topologia física base (topo_phy_base)
Saída: *bw_tables* atualizadas de todos switches de topologia física base (topo_phy_base)

```
1:  para todo switch ToR em topo_phy_base faça
2:      filaSwitches.adicionaItem(ToR, PrefixoDeRedeDeToR, ToR.bw_table, porta=1)
3:  fim para
4:  enquanto filaSwitches não vazia faça
5:      (switch, PrefixoDeRede, bw_tableDeSwitch, porta) = filaSwitches.retiraItem()
6:      se (switch, PrefixoDeRede, porta) presente em SwitchesEmFila então
7:          bw_tableDeSwitch ← SwitchesEmFila(switch, PrefixoDeRede, porta)
8:          SwitchesEmFila.removeItem(switch, PrefixoDeRede, porta)
9:      fim se
10:     SwitchesVisitados.adicionaItem(switch, PrefixoDeRede, porta)
11:     bw_usage_list ← lista de todos os valores de largura de banda em switch.bw_table com destino a PrefixoDeRede
12:     bw_usage_mean ← Maior valor a ser alocado igualmente em todas entradas de bw_usage_list para PrefixoDeRede
13:     se porta contida em enlaces a outros switches então
14:         switch_bw_table[PrefixoDeRede][porta] ← min(bw_usage_mean,switch._bw_port_table[porta])
15:     fim se
16:     para todo enlace adjacente a switch não contendo porta faça
17:         se (enlace.switchDestino,PrefixoDeRede,enlace.portaDestino) não presente em SwitchesEmFila e SwitchesVisitados
            então
18:             filaSwitches.adicionaItem(enlace.switchDestino,PrefixoDeRede,switch_bw_table,enlace.portaDestino)
19:         fim se
20:         se (enlace.switchDestino,PrefixoDeRede,enlace.portaDestino) presente em SwitchesEmFila então
21:             SwitchesEmFila(switch,PrefixoDeRede,porta) ← switch_bw_table
22:         fim se
23:     fim para
24: fim enquanto
```

É importante ressaltar que uma *bw_table* compreende a largura de banda fim-a-fim disponível para uma rota, enquanto que a *bw_port_table* representa a largura de banda disponível local, ou seja, somente em enlaces adjacentes a switches. Além disso, switches ToR tem em suas *bw_table* os valores de largura disponíveis para cada um dos servidores que possuem, definindo assim a largura de banda disponível em um rack.

Tendo como base estes atributos, o algoritmo 1 atualiza as estruturas *bw_table* de todos os switches da topologia física base tornando-as consistentes com o estado das políticas mapeadas, e logo, propicias à utilização no algoritmo 1. Ou seja, este algoritmo, realizando uma busca em largura no grafo da topologia física base, faz com que todos os switches da topologia física base tenham informações atualizadas sobre a largura de banda disponível em cada rack da rede. Esta atualização ocorre somente na topologia física base, toda vez que uma política é criada, alocada e desalocada.

4.2. Algoritmo de Mapeamento Proposto

O algoritmo 2 abaixo, define uma busca em largura no grafo da topologia física base. Os nós de entrada são os switches ToR que uma política possui definidos pela matriz de tráfego de VMs mapeadas em racks de servidores. A saída do algoritmo é uma topologia virtual construída com base em informações de rotas da topologia virtual base e disponibilidade de largura de banda da topologia física base. O mapeamento ocorre por anotações de largura de banda, definidas pelo identificador único de uma política, feitas em enlaces da topologia física base. Estas marcações são realizadas de acordo com rotas selecionadas da topologia virtual pelo algoritmo 3 perante requisitos de largura de banda e resiliência definidos nas políticas de mapeamento.

Algoritmo 2 : Executado para mapear topologias virtuais na topologia física base

Entrada: topologia física base (topo_phy_base), topologia virtual base (topo_virt_base) e política
Saída: topologia virtual

para todo switch ToR em matriz de tráfego de política **faça**
    para todo PrefixoDeRede em matriz de tráfego de política $\neq$ faixa de endereços de rede de switch ToR **faça**
        lxc $\leftarrow$ lxc de topo_virt_base mapeado a ToR
        filaSwitches.adicionaItem(lxc, ToR, PrefixoDeRede)
        PropriedadesSwitches(lxc, ToR, PrefixoDeRede) $\leftarrow$ Largura de banda de ToR a PrefixoDeRede em matriz de tráfego de política
    fim para
fim para
enquanto filaSwitches não vazia **faça**
    (lxc, switch, PrefixoDeRede) $\leftarrow$ filaSwitches.retiraItem()
    se PrefixoDeRede não presente em entradas de SwitchesEmFila **então**
        SwitchesVisitados.adicionaItem(lxc, switch, PrefixoDeRede)
    fim se
    larguraDeBandaRequisitada = PropriedadesSwitches(lxc, switch, PrefixoDeRede)
    switch_rotas_selecionadas $\leftarrow$ SelecionaRotas(switch,lxc,PrefixoDeRede,larguraDeBandaRequisitada) # Algoritmo 3
    larguraDeBandaRotas$\leftarrow$larguraDeBandaRequisitada dividida pela quantidade de switch_rotas_selecionadas
    para todo rota em switch_rotas_selecionadas **faça**
        se larguraDeBandaRotas[rota] alocada em enlace de topo_phy_base definido por rota **então**
            Adicione switch, rota em switch e enlace em TopologiaVirtual
            Defina como lxcDestino e switchDestino os respectivos lxc e switch de destino do enlace definido por rota
            filaSwitches.adicionaItem(lxcDestino, switchDestino, PrefixoDeRede)
            PropriedadesSwitches(lxcDestino, switchDestino, PrefixoDeRede) $\leftarrow$ larguraDeBandaRotas[rota]
        else
            Mapeamento $\leftarrow$ Falso
            Pare os laços de execução
        fim se
    fim para
fim enquanto
se Mapeamento $\neq$ Verdadeiro **então**
    Desfaça mapeamentos até então feitos de politica em topo_phy_base
fim se

Algoritmo 3 SelecionaRotas: Executado para selecionar rotas para mapeamento

Entrada: (switch, lxc, PrefixoDeRede, larguraDeBandaRequisitada)
Saída: rotas selecionadas para mapeamento

switch_rotas $\leftarrow$ lxc.get_rotas(PrefixoDeRede)
Seleciona switch_rotas com maior capacidade em switch.bw_port_table que satisfaçam critério de resiliência de política
se Opção SelecRotas Agreg Proposto **então**
    Calcula média, desvio padrão, maior e menor valor de entradas de switch.bw_port_table com portas definidas por switch_rotas
    Seleciona combinações de rotas de switch_rotas que satisfazem larguraDeBandaRequisitada dividida entre elas **e** que definam valores de switch._bw_port_table maiores ou iguais a média subtraída do dobro do desvio padrão de switch.bw_port_table
    Para as combinações de rotas selecionadas na etapa anterior selecione aquela que possui a menor diferença entre os valores máximo e mínimo caso seus valores de largura de banda sejam selecionados e aplicados a switch.bw_port_table
    Retorne conjunto de rotas selecionadas na etapa anterior
fim se
se Opção SelecRotas Tradicional **então**
    Retorna rota de switch_rotas que satisfaz larguraDeBandaRequisitada em entrada PrefixoDeRede de switch.bw_table
fim se

Na opção "SelecRotas Agreg Proposto" do algoritmo 3, os critérios de resiliência de políticas são definidos por porcentagens, as quais expressam a quantidade de rotas que serão selecionadas para a avaliação de disponibilidade de caminhos. Além disso, nesta opção, a utilização do parâmetro de seleção de rotas, média subtraída do dobro do desvio padrão, permite que as rotas selecionadas estejam dentro de dois percentis da média dos valores de *bw_port_tables*. Isto garante que haja balanceamento de carga nas portas de um switch, ao contrário da opção "SelecRotas Tradicional", onde somente uma rota com menor disponibilidade de largura de banda é selecionada e nenhum critério de balanceamento de carga é realizado.

5. Avaliação Experimental

O testbed utilizado para a avaliação da arquitetura proposta neste trabalho foi montado utilizando a plataforma RouteFlow com as respectivas modificações mencionadas anteriormente. No plano de dados construímos a aplicação rfproxy sobre o controlador de rede Ryu[1], com as devidas modificações para suporte a OpenFlow 1.3. Utilizamos o switch ofsoftswitches13[2] tanto no plano de dados quanto no plano virtual. Realizamos a construção de duas topologias folded-Clos utilizando o emulador Mininet[3], com 12 e 48 switches. No plano virtual realizamos a construção do mapeamento com agregação para as topologias físicas criadas. Consideramos na topologia física e no plano de controle que cada enlace entre swiches possui 10.000 unidades de largura de banda, e entre switches e servidores 1.000 unidades de largura de banda. Além disso, consideramos que nas topologia físicas com 12 e 48 switches existam respectivamente 20 e 40 servidores em cada rack, podendo cada servidor possuir no máximo 20 VMs alocadas.

5.1. Experimentos e Resultados

Inicialmente fizemos um estudo analítico sobre a abordagem de construção da arquitetura com a utilização do protocolo BGP em roteadores virtuais definidos pelo mapeamento com agregação de switches da topologia física. Nesse aspecto, buscamos observar uma comparação entre as topologias folded-Clos comum e com agregação, referenciadas respectivamente por topoClos e topoClosAgreg. Na Tabela 1 vemos que a topologia virtual com agregação determina um número menor de conexões entre roteadores virtuais, menor quantidade de conexões TCP para as sessões BGP e, consequentemente, menor número de mensagens trocadas em cada rodada de atualizações do protocolo BGP.

Tabela 1. Comparação de topologias

Topologia	Switches Plano de Dados	Roteadores Virtuais	Conexões entre Roteadores	Mensagens por rodada de atualização
topoClos	12	12	16	32
topoClosAgreg	12	7	6	12
topoClos	48	48	64	128
topoClosAgreg	48	9	8	16
topoClos	96	96	256	512
topoClosAgreg	96	11	34	68
topoClos	128	128	1024	2048
topoClosAgreg	128	35	68	136

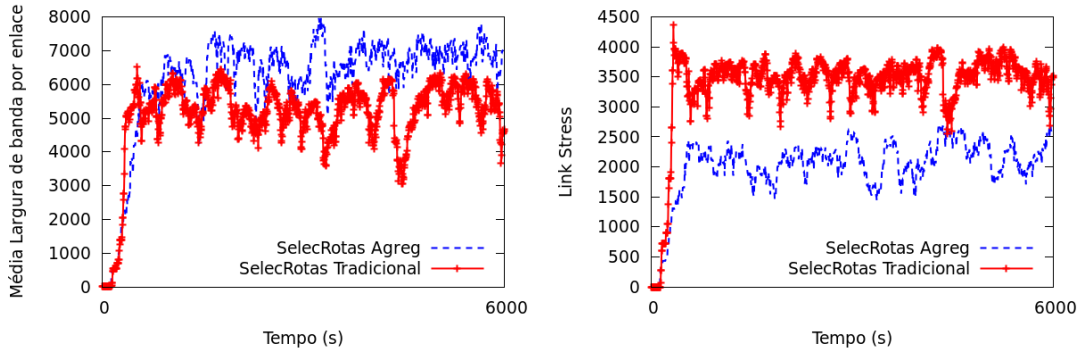
[1]https://github.com/osrg/ryu

[2]https://github.com/CPqD/ofsoftswitch13

[3]https://github.com/mininet/mininet

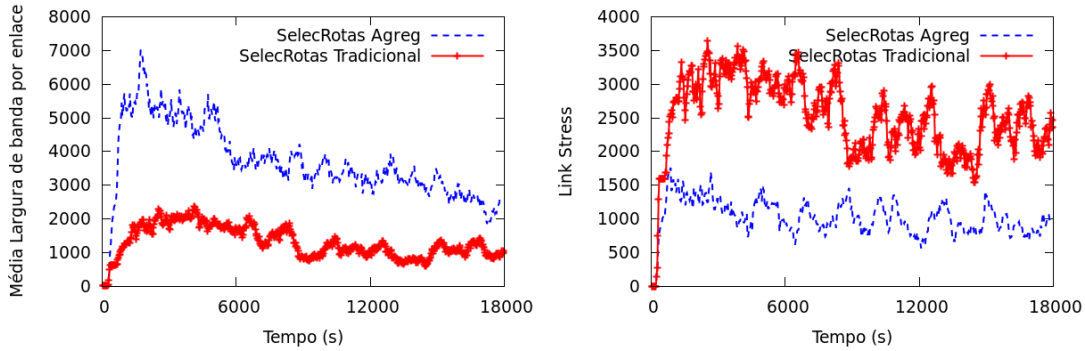
Adiante tratamos de avaliar o algoritmo “SelecRotas Agreg” proposto neste trabalho e compará-lo ao algoritmo “SelecRotas Tradicional”. Para isso, realizamos a criação de demandas de mapeamento em um processo de poisson com VMs sendo alocadas de forma ordenada em servidores de ToRs conforme a disponibilidade de recursos em servidores. Consideramos os seguintes parâmetros para a realização deste experimento: chegada de demandas por processo de Poisson com média de 30 requisições por minuto. Cada requisição possui, respectivamente para as topologias com 12 e 48 switches: quantidade de máquinas virtuais uniformemente distribuída entre 10 e 30 e entre 30 e 70; demanda de tráfego entre VMs uniformemente distribuída entre 1 e 10 unidades de largura de banda; redes virtuais mapeadas com tempo de permanência na rede uniformemente distribuído entre 180 e 300 e entre 540 e 660 segundos; e tempo total de experimento definido em 6000 e 18000 segundos.

Neste sentido, avaliamos a largura de banda e sua variação (*link stress*) nos enlaces da rede, para entendermos o balanceamento de carga na topologia física base. Vemos em todos os experimentos que para as topologias com 12 (Figura 4) e 48 (Figura 5) switches, que o algoritmo “SelecRotas Agreg” obteve maior utilização de largura de banda na rede e menor link stress do que o algoritmo “SelecRotas Tradicional”, contribuindo portanto, para o uso eficiente dos recursos da rede. É importante ressaltar, que em todos os experimentos obtivemos 100% de taxa de aceitação das requisições de mapeamento.



(a) Média de largura de banda utilizada por enlace (b) Link stress

Figura 4. Dados de topologia física folded-Clos com 12 switches



(a) Média de largura de banda utilizada por enlace (b) Link stress

Figura 5. Dados de topologia física folded-Clos com 48 switches

Buscando obter critérios de desempenho da arquitetura criada e do algoritmo proposto, realizamos experimentos para avaliar o tempo de mapeamento de demandas conforme o tamanho de requisições de redes virtuais. Discriminamos os tempos de operações de: pré configuração e checagem de topologias; mapeamento da rede virtual; e configuração das rotas na topologia física. Realizamos experimentos variando o tamanho das políticas de mapeamento, com matrizes de tráfego entre 1 e 16 racks, na topologia com 48 switches. Observamos na Figura 6 que os tempos de pré-configuração tem definição praticamente constante por não dependerem do tamanho das políticas de mapeamento. Já os tempos de mapeamento e configuração crescem conforme o tamanho das políticas devido a quantidade de rotas a serem analisadas, selecionadas, mapeadas e configuradas nos racks. No caso da topologia analisada, notamos que existem pequenos saltos de tempo conforme uma política utiliza racks não interconectados aos mesmos switches EoR, o que se deve à necessidade de mapeamento de rotas em switches Core.

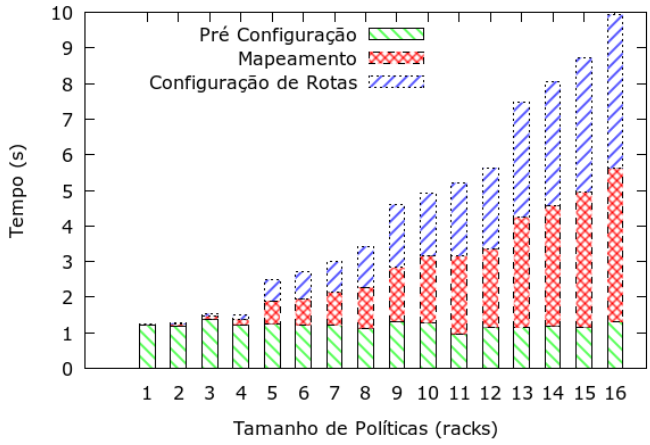


Figura 6. Tempos de Mapeamento

5.2. Discussão

Notamos nos resultados obtidos, com experimentos nas topologias folded-Clos com 12 e 48 switches (Figuras 4 e 5), que o algoritmo proposto no nosso trabalho realiza o balanceamento de carga na rede de forma eficiente quando comparado ao algoritmo "SelecRotas Tradicional", comumente visto na literatura (ex: [Ballani et al. 2011] e [Xie et al. 2012]). Isto pois, determina maior utilização e melhor espalhamento de alocações de largura de banda nos enlaces da rede. Além disso, vimos na Tabela 1 que o protocolo BGP, além de possuir as vantagens citadas no início da seção 3, quando implementado na topologia virtual folded-Clos com agregação, propicia as vantagens de menores utilização de roteadores virtuais e overhead do protocolo BGP. Por fim, vemos que a complexidade dos algoritmos apresentados condiz com os resultados apresentados na Figura 6, onde os tempos de mapeamento e de configuração de rotas crescem linearmente com o tamanho da política de requisição.

Diferente das propostas existentes na literatura, até onde temos conhecimento, a arquitetura construída neste trabalho é a única que faz utilização do padrão OpenFlow 1.3 e aborda explicitamente requisitos de balanceamento de carga na rede. Além disso, não só a utilização de múltiplos caminhos, como também de critérios de resiliência de políticas,

é uma abordagem até então desconhecida na literatura de DCNs. Agregar informações em grafos de topologias física e virtual em um plano de controle logicamente centralizado, permite uma diversidade de oportunidades de controle da rede e mapeamento de redes virtuais. Logo, tanto questões relacionadas a formas de endereçamento, flexibilidade da rede, desenvolvimento e criação de novas aplicações de rede, entre outras, podem ser feitas com total liberdade sobre a arquitetura proposta. Tal fato permite a abordagem de diversos problemas de DCNs pela extensão deste trabalho, considerando para isso diferentes requisitos de operação destas redes.

A arquitetura construída também permite um alto grau de liberdade na criação de redes virtuais. Toda a agregação de estado dos planos físico e virtual da arquitetura propicia ao plano de controle uma visão não conhecida nos trabalhos relacionados a esta proposta. Primeiro, pois objetivos de alto nível podem ser facilmente implementados tanto em políticas criadas conforme demandas de alocação de redes virtuais quanto na rede interna do próprio InP, seja pela programação de regras na seleção de rotas ou pela definição de configurações na própria execução do protocolo BGP no plano virtual. Do mesmo modo, a permissividade de características do padrão OpenFlow 1.3 dá ao mapeamento de redes virtuais todos os parâmetros necessários para programá-las de diferentes formas, como por exemplo, definindo a seletividade de tráfego de uma rede virtual somente por portas TCP ou diferentes protocolos de rede (ex: VXLAN, NVGRE).

6. Conclusões e Trabalhos Futuros

Neste artigo buscamos avaliar a alocação de redes virtuais em data centers utilizando o modelo de Network-as-a-Service com a aplicação de conceitos de Redes Definidas por Software. Construímos uma arquitetura utilizando a plataforma RouteFlow a fim de moldar todo o ambiente de rede, assim como os requisitos de operação de uma arquitetura de data center. Utilizamos para isso o mapeamento da infraestrutura física da rede de data center em uma topologia virtual onde rotas provenientes do protocolo BGP, configurado precisamente para este ambiente, foram obtidas para a alocação de topologias de data centers virtuais. Por meio de algoritmos implementados sobre o plano de controle da plataforma RouteFlow, conseguimos mostrar eficácia na alocação de redes virtuais considerando os fatores de utilização de largura de banda e balanceamento de carga da rede, e o overhead do protocolo de roteamento.

Vemos neste trabalho que muitos desafios surgem da aplicação de conceitos de SDN em DCNs, principalmente pelo uso do conceito de Network-as-a-Service. A avaliação de critérios de desempenho e escalabilidade em DCNs, utilizando novas formas de endereçamento, mapeamento de VMs em servidores, outras funcionalidades do OpenFlow 1.3, são tópicos de pesquisa que já estão em andamento. Futuramente iremos estender o funcionamento da plataforma para suportar tolerância a falhas em todos os seus planos. Buscaremos também elaborar novos algoritmos que levem em consideração requisitos de gerenciamento de energia no que diz respeito à utilização de recursos de rede e como estes interesses podem ir de encontro às abordagens resilientes de mapeamento de redes virtuais em data centers. Por exemplo, a seleção de rotas poderia ser estabelecida de diversas formas, considerando pesos de enlaces, políticas de roteamento interno, ou mesmo critérios de utilização de energia em enlaces e resiliência de caminhos. E por fim, vemos que os nós de entrada do algoritmo proposto podem ser estendidos a servidores, tornando esta solução flexível à programação de switches virtuais.

7. Referências Bibliográficas

Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. (2011). Towards predictable datacenter networks. In *Proceedings of the ACM SIGCOMM 2011*, pages 242–253, New York, NY, USA. ACM.

Bari, M., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q., and Zhani, M. (2013). Data center network virtualization: A survey. *Communications Surveys Tutorials, IEEE*, 15(2):909–928.

Guo, C., Lu, G., Wang, H. J., Yang, S., Kong, C., Sun, P., Wu, W., and Zhang, Y. (2010). Secondnet: A data center network virtualization architecture with bandwidth guarantees. In *Proceedings of Co-NEXT '10*, pages 15:1–15:12, New York, NY, USA. ACM.

Keller, E. and Rexford, J. (2010). The "platform as a service" model for networking. In *Proceedings of INM/WREN'10*, pages 4–4, Berkeley, CA, USA. USENIX Association.

Lam, V. T., Radhakrishnan, S., Pan, R., Vahdat, A., and Varghese, G. (2012). Netshare and stochastic netshare: Predictable bandwidth allocation for data centers. *SIGCOMM Comput. Commun. Rev.*, 42(3):5–11.

Lapukhov, P., Premji, A., and J. Mitchell, E. (2013). Use of bgp for routing in large-scale data centers. Internet-Draft draft-lapukhov-bgp-routing-large-dc-06.txt, IETF Secretariat.

Nascimento, M. R., Rothenberg, C. E., Denicol, R. R., Salvador, M. R., and Magalhaes, M. F. (2011). Routeflow: Roteamento commodity sobre redes programáveis. *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos-SBRC*.

ONF (2012). Openflow switch specification version 1.3.0 (wire protocol 0x04). Technical report, ONF.

Popa, L., Krishnamurthy, A., Ratnasamy, S., and Stoica, I. (2011). Faircloud: Sharing the network in cloud computing. In *Proceedings of the 10th ACM HotNets*, pages 22:1–22:6, New York, NY, USA. ACM.

Rodrigues, H., Santos, J. R., Turner, Y., Soares, P., and Guedes, D. (2011). Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *Proceedings of WIOV'11*, pages 6–6, Berkeley, CA, USA. USENIX Association.

Shieh, A., Kandula, S., Greenberg, A., and Kim, C. (2010). Seawall: Performance isolation for cloud datacenter networks. In *Proceedings of the 2Nd USENIX HotCloud*, pages 1–1, Berkeley, CA, USA. USENIX Association.

Stringer, J. P., Fu, Q., Lorier, C., Nelson, R., and Rothenberg, C. E. (2013). Cardigan: Deploying a distributed routing fabric. In *Proceedings of the Second ACM SIGCOMM HotSDN '13*, pages 169–170, New York, NY, USA. ACM.

Verdi, F. L., Rothenberg, C. E., Pasquini, R., and Magalhaes, M. (2010). Novas arquiteturas de data center para cloud computing. In *28th Brazilian SBRC 2010 - Minicursos*.

Xie, D., Ding, N., Hu, Y. C., and Kompella, R. (2012). The only constant is change: Incorporating time-varying network reservations in data centers. In *Proceedings of SIGCOMM '12*, pages 199–210, New York, NY, USA. ACM.