Um Mecanismo de Comutação Multicaminhos de Pacotes Baseado em OpenFlow para Redes de Centros de Dados*

Diogo Menezes Ferrazani Mattos , Lucas Henrique Mauricio , Lyno Henrique Gonçalvez Ferraz e Otto Carlos Muniz Bandeira Duarte

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{menezes,lucas,lyno,otto}@gta.ufrj.br

Resumo. *O crescente número de servidores e comutadores nos Centros de Dados, que hoje chega a dezenas de milhares, fez com que a rede de comunicação se tornasse um grande desafio. O algoritmo de árvore de cobertura é convencionalmente usado na interconexão de pontes Ethernet para evitar caminhos com laços. No entanto, a construção de uma árvore, livre de redundâncias, restringe a comunicação a um caminho único e, consequentemente, provoca sobrecarga e congestionamento em enlaces compartilhados por muitos caminhos. Comutadores baseados em OpenFlow centralizam o controle das tabelas de comutação permitindo de forma trivial calcular vários caminhos e evitar a formação de laços. Este trabalho propõe um mecanismo de interconexão multicaminho para uma maior eficiência na taxa de transferência da comunicação em Centro de Dados. A ideia básica é usar comutadores OpenFlow que centralizam o controle das tabelas de comutação, permitindo de forma fácil evitar laços e calcular múltiplos caminhos entre a origem e o destino. O desempenho do mecanismo proposto é verificado através da emulação de uma rede OpenFlow com topologias comumente empregadas em Centros de Dados. Os resultados mostram que o mecanismo proposto obteve sucesso ao espalhar as cargas de tráfego em Centro de Dados com topologias de redes redundantes e aumentou a máxima vazão alcançada em até 50% quando comparado com o algoritmo clássico de árvore de cobertura.*

Abstract. *The increasing number of switches and servers in Data Center introduces new challenges to communication and to Data Center network organization. Spanning Tree Algorithm is currently used to avoid paths with loops. The construction of a redundancy-free spanning tree restricts communication to a unique path and also causes bottleneck due to a high share of certain network links. OpenFlow based switches have a centralized controller that allows calculate multiple paths and avoid loops. In this paper, we propose an algorithm that provides multiple different paths in a Data Center communication, which improves the data transfer speed and achieved maximum bandwidth. We emulate an OpenFlow Data Center network and experimented our algorithm. The results show that our algorithm successfully spread traffic loads in high redundant Data Center topologies and incresaed the maximum achived throuput up to 50%, when compared to the classical Spanning Tree Algorithm.*

1. Introdução

As topologias de Centros de Dados *Data Centers* tornaram-se complexas devido ao enorme crescimento do número de servidores que estão sendo instalados para

*Este trabalho foi realizado com recursos da FINEP, FUNTTEL, CNPq, CAPES, FAPERJ e UOL.

atender as demandas de largura de banda, confiabilidade e novos usuários. Uma rede de Centro de Dados atualmente tenta conciliar redundância, escalabilidade e eficiência de custo [Couto et al. 2012a, Couto et al. 2012b]. Nesse sentido, a tecnologia de comunicação principal usada em redes de Centro de Dados é a Ethernet por causa da sua capacidade de autoconfiguração e o seu baixo custo para proporcionar elevadas taxas de transmissão. A comutação Ethernet, no entanto, é um desafio porque protocolo Ethernet não controla a formação laços (*loops*) na rede. Assim, o algoritmo árvore de cobertura (*Spanning Tree*) foi proposto para o encaminhamento de pacotes na camada de enlace em uma topologia arbitrária [Touch and Perlman 2009]. Esse algoritmo garante o estabelecimento de caminhos sem qualquer redundância ou laços. A árvore de cobertura constrói uma árvore para conectar todos os nós da rede, desativando os demais enlaces que não são eleitos para compor essa árvore. Como o grafo de interconexão dos nós resultante é uma árvore, qualquer par de nós só tem um único caminho entre origem e destino.

A ausência de enlaces redundantes resolve questões como o encaminhamento de pacotes em um laço infinito que impede o funcionamento de protocolos de difusão (*broadcast*) e de encaminhamento para múltiplos destinatários (*multicast*) [Nakagawa et al. 2012]. Contudo, os elementos de encaminhamento são limitados ao uso de um caminho único pertencente à árvore de cobertura para todo o tráfego *unicast* e *multicast*. Logo, há uns enlaces que são desativados enquanto outros ficam sobrecarregados, causando congestionamento nas redes de Centro de Dados [Al-Fares et al. 2010]. Assim, para uma maior eficiência da taxa de transferência de pacotes nas redes de Centro de Dados, todos os enlaces da rede de interconexão deveriam ser considerados no cálculo do melhor caminho entre quaisquer dois nós, e não se restringir aos enlaces da árvore de cobertura. Além disso, deve ser levada em consideração a possibilidade de se transferir pacotes da origem para o destino por múltiplos caminhos, aumentando assim a taxa de transferência e também a eficiência da banda passante disponível. Os laços, evidentemente, continuam tendo que ser evitados. A tecnologia de comutação OpenFlow permite resolver facilmente este problema uma vez que centraliza o controle das tabelas de comutação.

Este trabalho propõe: i) uma extensão do algoritmo de Dijkstra para calcular vários caminhos; e ii) um mecanismo baseado na API (*Application Programming Interface*) OpenFlow para implantar o algoritmo de melhores caminhos em uma rede comutada Ethernet de Centro de Dados. A avaliação da proposta foi realizada através da emulação de redes OpenFlow [McKeown et al. 2008, Mattos et al. 2011] através do Mininet [Lantz et al. 2010]. Por se tratar de uma emulação, o controlador OpenFlow usado implementa de fato o mecanismo proposto através de uma aplicação para o controlador NOX [Gude et al. 2008] que é a base da implementação do algoritmo Dijkstra aplicado ao conceito de múltiplos caminhos. A ideia central do algoritmo proposto é pré-calcular um determinado número de caminhos diferentes e de menor custo entre pares de nós e, quando for definir um novo fluxo na rede, escolher um dos caminhos calculados *a priori*. A avaliação da proposta considerou entre 10% e 100% de uso da rede e mediu a largura de banda do tráfego total alcançada. Os resultados mostram que o mecanismo proposto tem a vantagem de usar caminhos redundantes, melhorando a taxa de transferência total nas topologias de malha completa, BCube [Guo et al. 2009], e Fat-Tree. O mecanismo proposto apresenta o mesmo desempenho para a topologia em estrela que não tem redundâncias. Os resultados mostram ainda que a proposta, quando comparada com o algoritmo de árvore de cobertura, melhora em até 50% a largura de banda alcançada para a topologia BCube, enquanto melhora 22% e 13% a largura de banda obtida pelas topologias Fat-Tree e de malha completa respectivamente.

O mecanismo de múltiplos caminhos proposto depende da tecnologia OpenFlow. O OpenFlow se baseia no paradigma de Redes Definidas por *Software* (SDN – *Software Defined Networking*) [Casado et al. 2012], que separa a comutação em dois planos: o encaminhamento e o controle de dados. O OpenFlow também centraliza o controle de dados. A proposta aproveita a visão global da rede do controlador centralizado para executar o algoritmo de Dijkstra estendido. Esta abordagem difere de trabalhos anteriores, pois as propostas atuais para múltiplos caminhos em Centros de Dados se baseiam na criação de VLANs para identificar e distribuir o tráfego [Mudigonda et al. 2010, Raiciu et al. 2011]. Na proposta deste artigo não há necessidade de criação de qualquer VLAN, pois a classificação de todos os pacotes entre os caminhos é feita pelo OpenFlow. Outra importante vantagem do mecanismo proposto é que ele é leve, uma vez que não apresenta qualquer sobrecarga de cabeçalho para o pacote enviado, ao contrário do que acontece em outras propostas [Touch and Perlman 2009]. O restante do artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta o modelo de comutação em redes com OpenFlow. O mecanismo proposto é detalhado na Seção 4. A avaliação do mecanismo e os seus resultados são apresentados na Seção 5. A Seção 6 conclui o artigo.

2. Trabalhos Relacionados

As principais abordagens atuais para prover o encaminhamento de pacotes em redes Ethernet por múltiplos caminhos envolve a troca dos equipamentos de encaminhamento de prateleira dos Centros de Dados por novos equipamentos com essa nova funcionalidade [Duarte et al. 2012]. Contudo, propostas de Redes Definidas por *Software* abordam a criação de redes para atender características "sob medida" (*Software Defined Networking*), baseadas em equipamentos de prateleira. Arquiteturas de rede para Centros de Dados, topologias dessas redes e aplicação do conceito de redes definidas por *software* nesse cenário representam desafios para o desenvolvimento de soluções de encaminhamento em múltiplos caminhos [Duarte et al. 2012, Couto et al. 2012b].

Hao *et al.* apresentam a infraestrutura VICTOR (*Virtually Clustered Open Router*) [Hao et al. 2009] que se baseia em criar um agrupamento de Centros de Dados, através de uma rede definida por *software*. A ideia central dessa proposta é usar o OpenFlow como a infraestrutura básica da rede de Centro de Dados, de modo a permitir o uso da virtualização de máquinas e que, ao migrar uma máquina virtual de uma localização física para outra, seja possível reconfigurar os caminhos na rede. Essa proposta se serve do OpenFlow para otimizar o uso da rede realizando migrações de servidores, porém não aborda a comunicação entre servidores virtualizados através de múltiplos caminhos aproveitando a redundância da topologia da rede do Centro de Dados.

O TRILL (*Transparent Interconnection of Lots of Links*) é uma proposta para realizar o encaminhamento de pacotes na camada de enlace em redes de grande porte como as de Centros de Dados, com suporte a múltiplos caminhos e sem a necessidade de se definir uma árvore de cobertura [Touch and Perlman 2009]. O TRILL é um protocolo padronizado pelo IETF (*Internet Engineering Task Force*) e aplica técnicas de roteamento da camada de rede para interconectar comutadores Ethernets na camada de enlace [Touch and Perlman 2009]. A interconexão dos comutadores é feita de forma transparente para o protocolo de camada de rede como o IP. Os enlaces interconectados pelo TRILL são vistos pela camada de rede como uma única subrede IP, por exemplo. O TRILL encapsula, com um cabeçalho próprio, os quadros Ethernet a serem encaminhados. Como o quadro Ethernet é mantido inalterado, o TRILL garante a provisão das funcionalidades da camada de enlace, como VLAN (*Virtual Local Area Network*), e permite

o uso de *multicast* e *broadcast*. A ideia central do TRILL é evitar a formação de grandes Árvores de Cobertura nas redes dos Centros de Dados e aproveitar a possibilidade de uso de múltiplos caminhos usando técnicas de roteamento em uma rede sobrecamada entre as camadas de enlace e rede. Já a arquitetura de redes para Centros de Dados DOVE (*Distributed Overlay Virtual Ethernet*) é uma proposta de virtualização de redes que provê isolamento entre redes virtuais, capacidade de reconfiguração dinâmica da rede e de gerenciamento da rede virtual, independentemente da infraestrutura física preexistente no Centro de Dados [Barabash et al. 2011]. A arquitetura DOVE permite a criação de redes virtuais com espaços de endereçamentos isolados e dinâmicas sobre uma infraestrutura física comum. O funcionamento do DOVE baseia-se no encapsulamento de pacotes e na consequente criação de uma rede de sobrecamada para permitir a separação entre rede virtual e a rede física subjacente.

Al-Fares *et al.* argumentam que um dos principais desafios na organização de uma rede de Centro de Dados é maximizar a banda agregada disponibilizada às aplicações ao mesmo tempo em que se minimiza o custo com de implantação da topologia da da rede [Al-Fares et al. 2010]. Assim, topologia de redes proposta é constituída de uma árvore com múltiplas raízes, com diversos caminhos com custos iguais entre pares de nós. Al-Fares *et al.* propõem o sistema Hedera [Al-Fares et al. 2010], um sistema dinâmico de escalonamento de fluxos para topologias de comutadores em múltiplas camadas em Centro de Dados, cuja ideia básica é otimizar a banda agregada fornecida aos serviços e aproveitar os caminhos de mesmo custos fornecidos pela topologia de árvores com múltiplas raízes. O sistema Hedera coleta as informações de fluxos dos comutadores constituintes da rede, computa caminhos não conflitantes para os maiores fluxos e roteia novamente o tráfego de acordo com os novos caminhos calculados. O principal objetivo do sistema é maximizar a utilização da rede e minimizar a sobrecarga de escalonamento e o impacto nos fluxos ativos. O sistema Hedera foi implementado com base no OpenFlow.

Outra proposta para realizar a comutação por múltiplos caminhos em uma rede de Centro de Dados é a abordagem do TCP de múltiplos caminhos (MPTCP – *Multipath TCP*) [Raiciu et al. 2011]. A ideia principal dessa proposta é dividir grandes fluxos TCP em vários sub-fluxos menores e enviar a cada um deles por um caminho diferente na rede do Centro de Dados, evitando gargalos e usando melhor a capacidade da rede. Para garantir que os pacotes dos sub-fluxos sejam transmitidos realmente em caminhos diferentes, cada sub-fluxo é encaminhado por uma VLAN (*Virtual Local Area Network*). Contudo, o mecanismo para o cálculo dos melhores caminhos de cada VLAN e o escalonamento dos sub-fluxos em VLANs é ainda um desafio. Além disso, a proposta restringe-se ao protocolo de transporte TCP, que devido aos seus controles de fluxo e de congestionamento, ameaça a aplicação da proposta em casos nos quais o atraso de entrega de pacotes seja um fator importante, , já que o controle de fluxo reduz a banda usada por uma conexão TCP em caso de perdas de segmentos, e apresenta um custo de memória, pois e mantém na os segmentos em memória até que todos os segmentos em uma janela sejam entregues.

Outra abordagem de múltiplos caminhos baseada em VLANs é o SPAIN (*Smart Path Assignment In Networks*) [Mudigonda et al. 2010]. A ideia central do SPAIN é utilizar um controlador de rede centralizado para calcular *a priori* os múltiplos caminhos disponíveis na rede. Os objetivos principais do SPAIN são aumentar a banda de bisseção, banda de interconexão entre duas metades da rede,

Quanto às topologias de rede redundantes, existem diferentes topologias que consideram confiabilidade e custo. Quanto ao desempenho, Al-Fares *et al.* propõem o uso de uma topologia em múltiplas camadas de comutadores, formando árvores de comutadores

com múltiplas raízes [Al-Fares et al. 2010]. A topologia de árvores com múltiplas raízes é chamada de *Fat-Tree*. Popa *et al.* comparam tanto o consumo de energia quanto o custo de topologias de redes e concluem que a BCube, uma topologia híbrida de comutadores e roteadores, apresenta um custo muito baixo aliado à redundância e, por isso, é uma topologia adequada a Centro de Dados [Popa et al. 2010]. Contudo, dada a queda contínua do custo de comutadores, em um futuro próximo, topologia baseadas em comutadores irão superar a vantagem de custo da topologia híbrida. Couto *et al.* consideram a confiabilidade das topologias [Couto et al. 2012a]. Apesar do aumento do comprimento do caminho, eles concluem em favor de topologias híbridas, já que degradam de forma mais suave quando comparada a topologias como as *Fat-Trees* em caso de falha.

As principais características do mecanismo de encaminhamento em múltiplos caminhos proposto neste artigo são o uso de comutadores OpenFlow [McKeown et al. 2008, Mattos et al. 2011] para o encaminhamento dos pacotes, o fato de não usar encapsulamento de pacotes e a sua flexibilidade em relação ao uso de VLANs. As propostas relacionadas baseiam-se, em grande parte, na marcação de pacotes, seja por etiquetas de VLAN ou pelo encapsulamento dos pacotes encaminhados. A proposta deste artigo baseia-se exclusivamente nos endereços de origem e destino do pacote encaminhado e na definição de todo o caminho do fluxo no momento em que o pacote chega à rede OpenFlow do Centro de Dados. A maioria dos trabalhos relacionados citados baseiam-se na marcação de pacotes, seja por etiquetas de VLAN ou pelo encapsulamento dos pacotes encaminhados. Assim, a proposta é mais leve do que as demais e não requer sobrecarga de cabeçalho.

O mecanismo proposto neste artigo se baseia no paradigma de Redes Definidas por *Software* e é implementado como uma aplicação do NOX [Gude et al. 2008], controlador OpenFlow que apresenta uma visão global da rede. O mecanisno é avaliado em uma rede OpenFlow emulada através da ferramenta Mininet [Lantz et al. 2010].

3. O Paradigma Redes Definidas por *Software*

O paradigma de Redes Definidas por Software (*Software Defined Networking* - SDN) [Casado et al. 2012] se baseia nesse conceito de um *hardware* especializado em encaminhar pacotes associado a uma interface de programação que permite a redefinição, pelo administrador da rede, da lógica de encaminhamento de pacotes de um determinado tipo [Guedes et al. 2012]. A abordagem SDN estende o *hardware* de encaminhamento de pacotes de modo que a tarefa de encaminhamento seja pouco alterada, em relação à atual, e o desempenho do *hardware* estendido seja o mesmo do atual, enquanto novas primitivas de controle são oferecidas ao controlador da rede.

Uma Rede Definida por *Software* caracteriza-se pela existência de um sistema em *software* e centralizado que controle o mecanismo de encaminhamento dos elementos da rede através de uma interface de programação de aplicação bem definida [Guedes et al. 2012], conforme mostrado na Figura 1. O controle da rede pode ser executado por uma única aplicação, mas, na prática, o controle da rede é executado por um *software* de propósito geral sobre o qual se desenvolvem aplicações com propósitos específicos para o controle da rede. Esse software é o chamado controlador de rede. O controlador de rede pode assumir uma configuração centralizada, em que os elementos de rede programáveis se comunicam com ele e, então, o controlador possui uma visão unificada de todo o estado da rede. Assim, uma das principais vantagens da abordagem SDN é a formação de uma visão global, unificada, do controle da rede facilitando a tomada de decisões sobre a operação da rede. A visão global centralizada torna a programação da rede mais fácil e simplifica a representação dos problemas de rede [Guedes et al. 2012].

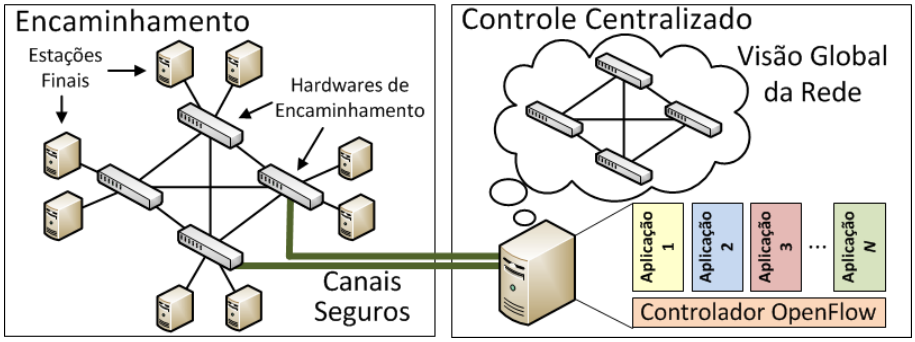


Figura 1. Rede Definida por *Software* separa as funções de encaminhamento e controle. O *hardware* de encaminhamento é controlado por aplicações centralizadas que têm uma visão global da rede.

As Redes Definidas por *Software*, ou simplesmente SDN, representam um novo paradigma para o desenvolvimento de pesquisas e aplicações na área de redes de computadores. O conceito de SDN deriva da arquitetura de rede proposta Ethane [Casado et al. 2007], na qual é proposto um mecanismo de controle de acesso distribuído para a rede, porém com o controle centralizado em um nó, responsável pela supervisão. Nessa arquitetura, a cada novo fluxo, o nó central era consultado para a verificação das políticas de acesso. Em caso de o novo fluxo ser permitido, o nó controlador central instalava o novo fluxo nos comutadores da rede através de uma interface de programação. Em caso negativo, caso as políticas de acesso bloqueassem a entrada do novo fluxo, os pacotes desse fluxo eram descartados. O modelo de programação de rede seguido pelo Ethane foi então estendido e formalizado, dando origem ao OpenFlow [McKeown et al. 2008]. O OpenFlow define que os elementos de encaminhamento ofereçam uma interface de programação de aplicação (API - *Application Programming Interface*) que permita um nó controlador centralizado estender as ações de controle e de acesso sobre a tabela utilizada pelo *hardware* de encaminhamento para determinar o próximo destino de cada pacote encaminhado.

Nesse contexto, o mecanismo de encaminhamento por múltiplos caminhos proposto baseia-se na visão global da rede centralizada no controlador OpenFlow para executar o algoritmo de Dijkstra estendido para múltiplos caminhos. A visão global da rede permite ao controlador conhecer todos os seus enlaces, mesmos os desativados pelo algoritmo de árvore de cobertura. Assim, o paradigma de SDN permite executar o controle centralizado da rede, prover múltiplos caminhos combinados com o uso de *hardware* de prateleira que implemente a API OpenFlow.

4. O Mecanismo Proposto

Uma maneira simples de encaminhar pacotes na camada de enlace é a comutação transparente. Os nós encaminhadores aprendem por qual porta um determinado nó final é acessível e passam a utilizar essa porta para encaminharem tráfego para esse nó. No entanto, esse método funciona somente quando há apenas um caminho entre quaisquer pares de nós. Assim, para permitir o funcionamento do encaminhamento em camada de enlace para topologias arbitrárias, foi criado o algoritmo de árvore de cobertura (*Spanning Tree Algorithm*) [Touch and Perlman 2009]. O algoritmo de árvore de cobertura garante a formação de caminhos na rede sem laços, mas desativa enlaces da rede física para evitar os laços. O algoritmo de árvore de cobertura faz com que todo o tráfego da rede fique restrito a um único conjunto de caminhos, pertencente à árvore de cobertura. Uma con-

sequência desse algoritmo é a sua instabilidade. A perda de conectividade em um enlace pertencente à árvore de cobertura pode significar o recálculo de toda a árvore e consequentes mudanças em toda a sua organização [Touch and Perlman 2009]. Dessa forma, o algoritmo de árvore de cobertura apresenta comportamento instável em redes extensas interligadas por comutadores, como as de Centros de Dados. Outra consequência do algoritmo de árvore de cobertura é a criação enlaces congestionados ao mesmo tempo que há enlaces desativados.

O mecanismo proposto, ao invés de usar apenas o caminho calculado pelo protocolo de arvore de cobertura (*Spanning Tree*), explora o uso dos vários caminhos possíveis na rede para melhorar o desempenho global da rede. A ideia básica do mecanismo proposto é que todos os pacotes (*unicast* e *multicast*) são encaminhados por múltiplos caminhos calculados pelo algoritmo de Dijkstra e apenas os pacotes em difusão são encaminhados pela árvore definida pela árvore de cobertura, para evitar que os pacotes enviados em difusão sejam enviados em um laço. No entanto, apenas o cálculo de melhores caminhos por Dijkstra não garante a distribuição dos fluxos na rede, pois não considera que pode haver vários caminhos entre dois nós, como é comum em topologias de Centros de Dados. O algoritmo de Dijkstra estendido que seleciona os N melhores caminhos, ao invés de apenas um único caminho, de um nó a todos os demais. A principal diferença entre o algoritmo estendido e o algoritmo clássico é que o clássico recebe o nó inicial e retorna o caminho mais curto para cada outro nó na rede, enquanto o estendido só converge após o cálculo de N melhores caminhos ou quando se esgotam as possibilidades de novos caminhos. O algoritmo de múltiplos caminhos recebe o nó inicial e o número de caminhos diferentes a serem calculados e retorna uma lista dos N caminhos mais curtos a partir do nó inicial para cada outro nó da rede. Assim, o controlador é capaz de decidir qual o caminho, entre até N caminhos selecionados, que vai ser usado. O controlador, ao instanciar um novo fluxo, escolhe de forma aleatória, seguindo uma distribuição uniforme, qual dos N caminhos calculados *a priori* entre origem e destino que deve ser usado.

Redes baseadas na API OpenFlow são uma possível infraestrutura para implementar o paradigma de Redes Definidas por *Software* (SDN - *Software Defined Networking*). O algoritmo de Dijkstra estendido executa como uma aplicação do controlador OpenFlow. No entanto, como o algoritmo de Dijkstra tem um alto custo computacional, não é viável realizar o cálculo dos melhores caminhos sob demanda de instanciação de novos fluxos. Dessa forma, o controlador calcula *a priori* os melhores caminhos. O cálculo dos caminhos entre todos os nós da rede é disparado sempre que há uma mudança na topologia da rede que afete os caminhos previamente calculados. Em uma rede de Centro de Dados a mudança na topologia dos elementos encaminhadores, no caso comutadores OpenFlow, não é um evento frequente e, no geral, são eventos programados [Couto et al. 2012b]. No momento da instanciação de um novo fluxo, o controlador escolhe aleatoriamente um entre os N caminhos possíveis e instancia o fluxo pelo caminho selecionado. Um novo cálculo de caminhos só é disparado quando as alterações da topologia, ou seja, quando a rede é inicializada pela primeira vez, ou cada vez que um comutador é adicionado ou removido.

O Algoritmo 1 mostra o funcionamento do encaminhamento com múltiplos caminhos. As entradas desse algoritmo são a base de conhecimento de caminhos calculados *a priori* pelo algoritmo de Dijkstra estendido e o pacote a ser encaminhado. O primeiro passo do algoritmo verifica se o pacote deve ser enviado em difusão ou não. Em caso positivo, o pacote é difundido na árvore de cobertura que interconecta todos os nós da rede. Em caso negativo, o pacote é encaminhado por um dos múltiplos caminhos. Para realizar

o encaminhamento, verifica-se qual o endereço MAC de origem e o endereço MAC de destino do pacote. Esses endereços são usados para identificar em qual comutador o nó de origem do pacote está conectado, MAC de origem, e em qual comutador o nó de destino do pacote está conectado, MAC de destino. A função $Escutou\_Primeiro()$ retorna o comutador ao qual um determinado endereço MAC está conectado diretamente. De posse dos comutadores de origem e de destino do pacote, a base de conhecimento de caminhos é consultada, resultando na lista de caminhos possíveis entre o comutador de origem e o de destino. A lista de caminhos é ordenada de acordo com o custo dos caminhos, em que o custo representa o número de saltos de um dado caminho. A escolha do caminho ocorre da seguinte forma. Um valor x é escolhido de forma aleatória, seguindo um probabilidade uniforme, entre os valores máximo e mínimo dos custos de todos caminhos. O valor x então é mapeado para o índice da lista de caminhos de acordo com a equação

$$i_{lista} = (N-1) * \left\lfloor \frac{x\text{–}Min_{custo}}{Max_{custo}\text{–}Min_{custo}} \right\rfloor, \quad (1)$$

em que N é o número de caminhos calculados *a priori* e Min_{custo} e Max_{custo} são respectivamente os custos do menor e maior caminho calculado. Após a seleção do caminho, o fluxo é definido na rede e o custo do caminho selecionado é acrescido de uma penalidade K.

Algoritmo 1: Mecanismo de encaminhamento de pacotes baseado em múltiplos caminhos.

Entrada: $Caminhos\{\}, pacote$
se $(Difusao(pacote) == Verdadeiro)$ **então**
  $Inundacao\_na\_Arvore\_de\_Cobertura(pacote)$
fim
senão
  $comutador_{origem} = Escutou\_Primeiro(pacote.mac_{origem})$
  $comutador_{destino} = Escutou\_Primeiro(pacote.mac_{destino})$
  $lista\_de\_Caminhos = Caminhos[comutador_{origem}][comutador_{destino}]$
  $x = Valor_Aleatorio(min(Custo), max(Custo))$
  $N = Cardinalidade(lista\_de\_Caminhos)$
  $indice_{caminho} = (N-1) * \lfloor \frac{x\text{–}Min_{custo}}{Max_{custo}\text{–}Min_{custo}} \rfloor$
  $caminho_{final} = lista\_de\_Caminhos[indice_{caminho}]$
  $Definir\_Fluxo\_no\_Caminho(caminho_{final})$
  $Custo(caminho_{final}) += K$ %Penalidade de Uso do Caminho
fim

5. Resultados

O protótipo desenvolvido se serve do controlador NOX para implementar a aplicação de comutação de pacotes por múltiplos caminhos. A aplicação de encaminhamento é escrita em Python e é avaliada em uma rede OpenFlow emulada pelo ambiente de experimentação Mininet [Lantz et al. 2010]. A ideia chave ao usar o Mininet é prover um ambiente de experimentação que execute exatamente o protocolo OpenFlow, ao contrário de simular o funcionamento idealizado da aplicação. O protótipo conta ainda

com geradores de topologias que replicam topologias comuns de Centro de Dados no ambiente de testes Mininet. A avaliação da proposta consiste em comparar o comportamento do encaminhamento de pacotes original do OpenFlow, usando o encaminhamento sobre os caminhos pertencentes à árvore de cobertura, com o encaminhamento provido pela comutação por múltiplos caminhos proposta. Na avaliação da proposta são considerados quatro casos, em que são calculados *a priori* 1, 2, 4 ou 8 melhores caminhos. Vale notar que o caso em que apenas um melhor caminho é calculado diferencia-se do caso clássico de encaminhamento no OpenFlow, pois o caso de um melhor caminho corresponde à aplicação do algoritmo de Dijkstra para o cálculo do melhor caminho entre dois nós na rede, não importando a árvore de cobertura definida e podendo usar qualquer enlace da rede, enquanto o caso clássico de encaminhamento só encaminha pacotes pelos caminhos aprendidos sobre a árvore de cobertura, ou seja, sujeito à desativação dos enlaces que não pertençam à árvore de cobertura.

O Mininet executa em uma máquina virtual Linux Ubuntu sobre o virtualizador VirtualBox[1]. A máquina virtual foi configurada com 64 GB de memória RAM e acessando 16 núcleos de processamento. A máquina física que hospeda a máquina virtual é um Intel Xeon X5570 de 2.93 GHz , com 96 GB de memória. Todos os comutadores, servidores e o controlador de rede NOX executam na máquina virtual. O Mininet instancia um controlador real, executando a aplicação desenvolvida, comutadores em software e cria ambientes virtuais que exercem a função de servidores.

Quatro diferentes topologias modelam as redes usadas na avaliação da proposta. Duas dessas topologias, Fat-Tree e BCube, são as topologias mais usadas para prover redes de Centros de Dados [Couto et al. 2012b]. A Fat-Tree [Al-Fares et al. 2010] é tradicionalmente usada em Centros de Dados, pois permite que qualquer servidor se comunique com qualquer outro na banda máxima de transmissão de dados, usando somente comutadores de prateleira (*comodity*), ou seja, com custo reduzido. A topologia Fat-Tree é ainda compatível com padrões e protocolos bastante difundidos, como é o caso do Ethernet e IP, permitindo que esses protocolos sejam executados em uma rede de Centro de Dados. A topologia BCube [Guo et al. 2009], por sua vez, propõe a construção de um Centro de Dados modular com baixo custo. A BCube apresenta tolerância a falhas, o que é importante, pois uma das propostas ao usar essa topologia é criar módulos de Centros de Dados selados em *containers* e, então, de difícil acesso para troca ou reparos. As outras duas topologias consideradas são a malha completa e a topologia em estrela. Essas duas últimas topologias foram consideradas por se tratarem de casos extremos, a malha completa representa o cenário em que todos se conectam diretamente a todos e a estrela, o cenário em que há um ponto único de conexão de todos os nós.

O experimento realizado consiste em definir um tráfego TCP entre pares de servidores. Cada experimento varia a taxa de utilização da rede entre 10 e 100%. O percentual de pares de nós se comunicando. Cada experimento é composto 10 rodadas e os resultados são apresentados com um intervalo de confiança de 95%. A ferramenta usada para gerar o tráfego nos ambientes virtuais é o Iperf[2]. A largura de banda total foi considerada como sendo o somatório de todas taxas alcançadas entre comunicações em uma rodada de experimentos

É importante ressaltar que Mininet emula uma rede OpenFlow, em vez de simular o funcionamento de rede [Li and Kwok 2005]. A simulação permite testar cenários maiores, mas usa modelos de rede simplificada, enquanto a emulação do Mininet permite

[1] https://www.virtualbox.org/.

[2]http://iperf.sourceforge.com/.

executar uma aplicação real em um ambiente emulado.

5.1. Topologia Fat-Tree

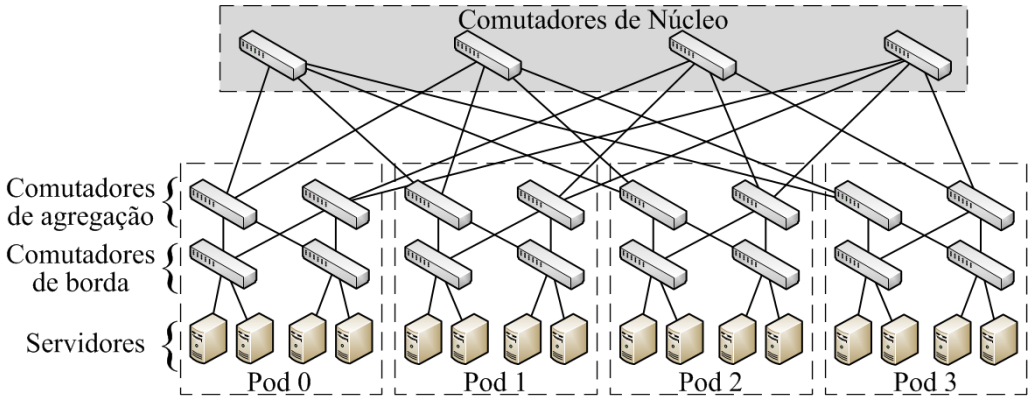


Figura 2. Topologia Fat-tree usada nos experimentos, com k=4. Topologia com 20 comutadores e 16 servidores.

A topologia Fat-Tree conta com dois grupos distintos de elementos, o núcleo (*core*) e os *pods*, mostrados na Figura 2. Cada comutador está conectado a cada *pod* por uma camada de comutadores de agregação. Os *pods* podem conter também comutadores de borda e servidores. Cada comutador de borda conecta um grupo de servidores. Todos os comutadores são idênticos e possuem k portas, conectados a uma árvore k-nária. O núcleo e os k *pods* possuem k comutadores. Cada $k/2$ comutadores de borda está conectado a $k/2$ comutadores de agregação e $k/2$ servidores. O número total de servidores é, então, $k * (k/2)^2$. No cenário de testes, a topologia Fat-Tree construída considera $k = 4$, resultando em vinte comutadores e dezesseis servidores.

Esta topologia apresenta diversos caminhos de mesmo custo para um dado par de servidores, portanto é um cenário favorável para a comutação com múltiplos caminhos. Dijkstra, $Multi_2$, $Multi_4$, e $Multi_8$ representam, respectivamente, 1, 2, 4 ou 8 melhores caminhos entre cada par de servidores. SPT representa o desempenho do método original de comutação OpenFlow com o algoritmo de árvore de cobertura. A Figura 4(a) mostra que a proposta $Multi_4$ apresentou desempenho até 22% superior quando comparado com SPT e, também, com $Multi_2$ ou Dijkstra. Tal comportamento ocorre porque, quando os quatro melhores caminhos são calculados, quaisquer deles podem ser usado para encaminhar sem perda de desempenho, já que os quatro caminhos têm custo igual, quando o par de servidores considerados estão em *pods* diferentes. $Multi_8$ é o pior caso, pois calcula muitos caminhos, o que ocasiona uma possibilidade de aproximadamente 50% de escolher um caminho com mais saltos do que o necessário. Esse comportamento implica a escolha de enlaces que já estejam sobrecarregados desnecessariamente.

5.2. Topologia BCube

A topologia BCube baseia-se comutadores, com baixo número de portas (*mini-switches*) e baixo custo, os chamados COTS (*commodity off-the-shelf*), e servidores, que também desempenham o papel de encaminhamento de pacotes. O *pod* chamado $BCube_0$ consiste de n servidores conectados a um n comutadores. A camada seguinte é a $BCube_1$ que está conectada a n *pods* $BCube_0$. Cada comutador em $BCube_1$ está ligado a um único servidor de cada $BCube_0$. A topologia pode ser vista na Figura 3. A figura evidencia que certos pares de servidores só podem se comunicar através de outros servidores atuando

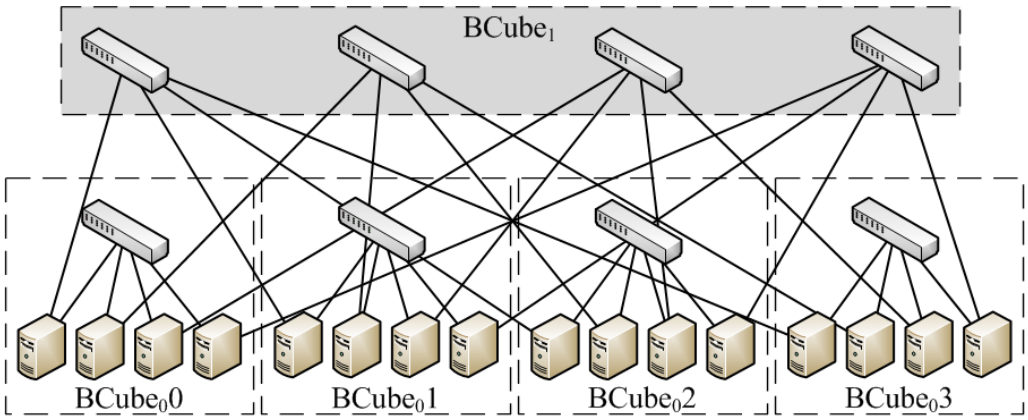
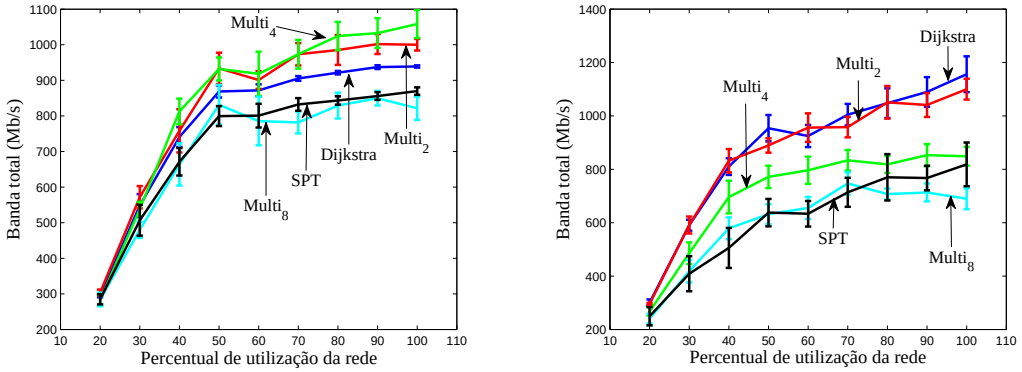


Figura 3. Topologia BCube usada nos experimentos, com n=4. Topologia com 8 comutadores e 16 servidores.

como elementos de encaminhamento. O experimento realizado implanta uma topologia BCube com $n = 4$, com oito comutadores e dezesseis servidores.



(a) Banda agregada na topologia Fat-Tree

(b) Banda agregada na topologia BCube

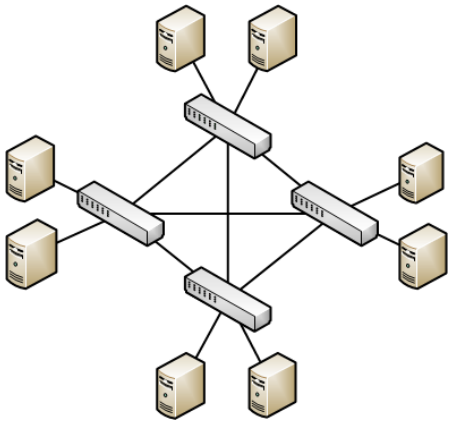
Figura 4. Resultados dos experimentos com Fat-Tree e BCube.

A topologia BCube apresenta pouca redundância entre caminhos e, por isso, a Figura 4(b) apresenta que a banda agregada ao usar Dijkstra ou $Multi_2$ é até 50% superior à alcançada com a comutação clássica pela árvore de cobertura e, por sua vez, os ganhos ao usar $Multi_4$ não são significativos. Na topologia BCube com n = 4, verifica-se que há dois caminhos diferentes, porém com o mesmo custo, entre dois servidores que estão em $BCube_0$ diferentes e em diferentes posições. Por exemplo, o servidor mais a direita em $BCube_0 3$ apresenta dois caminhos de custos iguais com segundo servidor, da direita para a esquerda, em $BCube_0 2$. A abordagem de encaminhamento clássica com árvore de cobertura, representada por SPT, apresenta desempenho semelhante ao $Multi_8$. Em $Multi_8$, o controlador calcula caminhos mais longos que utilizam enlaces desnecessários causando descongestionamentos.

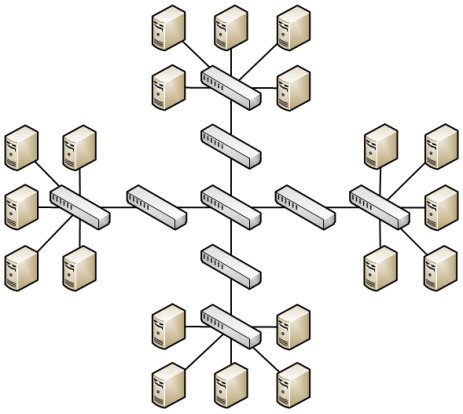
5.3. Topologia em Malha Completa

A topologia em malha completa caracteriza-se por cada comutador se conectar a todos os outros. A topologia com k comutadores, cada um com tem $k - 1$ portas conectadas aos demais $k - 1$ comutadores. Nos experimentos, cada comutador conecta

dois servidores, totalizando $2k$ servidores. A Figura 5(a) mostra uma topologia em malha completa com $k = 4$. Devido a limitações de *hardware*, o experimento considera $k = 8$, resultando em oito chaves e dezesseis servidores.



(a) Topologia em malha completa com $k = 4$. A topologia usada nos experimentos usa $k = 8$, resultando em 8 comutadores e 16 servidores.



(b) Topologia em estrela com nove comutadores. Os experimentos empregam a topologia com vinte servidores.

Figura 5. Topologia em malha completa e em estrela.

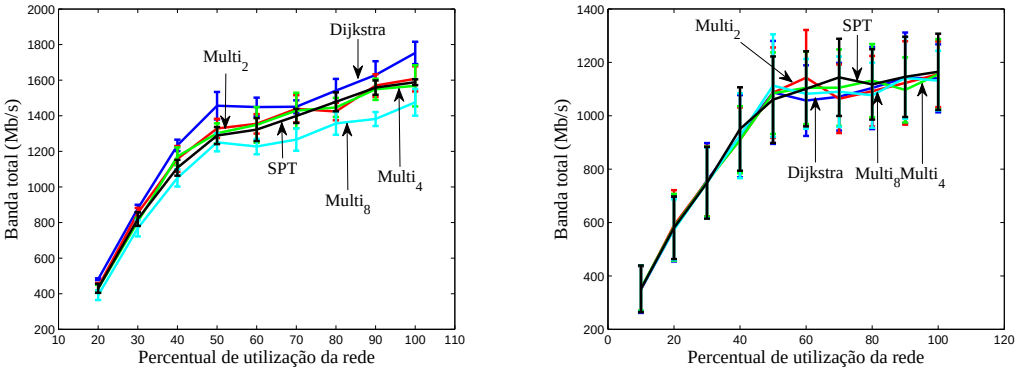
Figura 6(a) mostra que, quanto maior o percentual de uso da rede, maior a diferença entre largura de banda obtida entre as propostas testadas. A proposta de apenas um melhor caminho apresentou uma banda agregada superior em 13% em relação à abordagem clássica, pois a topologia em malha completa apresenta uma ligação direta entre quaisquer dois servidores. Os resultados das propostas de comutação múltiplos caminhos 'utilizando dois ou quatro melhores caminhos apresentam desempenho semelhante à comutação clássica com árvore de cobertura (STP). Esse resultado explica-se pelo fato de que as abordagens de dois ou quatro melhores caminhos utilizam mais enlaces na rede em relação à abordagem de um melhor caminho único, e esta é, em média, semelhante à abordagem clássica da árvore de cobertura. A abordagem com menor uso de banda agregada foi a $Multi_8$. Quando caminhos oito são computados utilizando uma topologia em malha completa, a probabilidade de escolha de caminhos maiores do que o melhor é de 87,5%, o que indica que pode haver um uso desnecessário de enlaces na rede, causando uma ocupação desnecessária dos recursos da rede.

5.4. Topologia em Estrela

A topologia em estrela é composta por um comutador central e camadas de comutadores, como mostrado na Figura 5(b). A topologia avaliada conta com quatro comutadores na camada mais externa, cada comutador conecta a cinco servidores. O número total de servidores é vinte. O tamanho dessa topologia foi escolhido para manter aproximadamente o mesmo número de servidores que nas topologias anteriores. A topologia em estrela não apresenta redundâncias de caminhos, assim, o único caminho possível entre dois servidores é através do caminho que pertence à arvore de cobertura, já que a topologia, por si só, já forma uma árvore. A Figura 6(b) apresenta o desempenho das propostas comparadas.

Os resultados mostram que o mecanismo de comutação de múltiplos caminhos proposto calcula, de forma eficiente, caminhos aumentam o espalhamento do fluxo na

rede de Centros de Dados, o que aumenta o desempenho da rede quando comparado com a abordagem clássica de comutação com árvore de cobertura. O único caso em que o mecanismo proposto não é superior é ao executar sobre a topologia em estrela, em que não há redundâncias e há apenas um caminho possível entre qualquer par de servidores. Os experimentos com a topologia em malha completa apresentou um ganho máximo de 13% em relação ao clássico, enquanto os experimentos com Fat-Tree e BCube apresentaram ganhos máximos de 22% e 50%, respectivamente.



(a) Banda agregada na topologia BCube

(b) Banda agregada na topologia em estrela

Figura 6. Resultados dos experimentos com as topologias em malha completa e em estrela.

6. Conclusão

A comutação por múltiplos caminhos substitui com vantagens o mecanismo clássico de comutação com base na árvore de cobertura. A árvore de cobertura define um conjunto de enlaces na rede que atinge todos os nós da rede, mas desativa enlaces para evitar laços e elimina a redundância de caminhos. Por outro lado, o algoritmo proposto de múltiplos melhores caminhos permite decidir entre todos os enlaces da rede quais são os mais adequados para executar a comunicação entre qualquer par de nós. Esse artigo propôs uma abordagem de comutação por múltiplos caminhos, apoiada no paradigma de Redes Definidas por *Software* (SDN). A proposta aproveita a visão global do controlador OpenFlow, responsável pelo cenário de SDN, para calcular os melhores caminhos entre os pares de nó. A proposta define um mecanismo leve, que executa funções de baixa complexidade no momento da definição de novos fluxos. Para avaliar a proposta foi desenvolvida uma aplicação OpenFlow e avaliada em um ambiente com nós emulando uma rede OpenFlow em Mininet. Os resultados mostram que a proposta melhorou a taxa de transferência total agregada da rede, quando comparado com o algoritmo de árvore de cobertura clássico, em até 50% para a topologia BCube, 22% para topologia Fat-Tree e 13% para a topologia em malha completa. Verificou-se, também, que a proposta apresenta o mesmo desempenho para a topologia em estrela que não provê nenhuma redundância.

Como trabalhos futuros, serão avaliados novos algoritmos para a computação de melhores caminhos no controlador. Outra abordagem a ser seguida é adaptação dos fluxos na rede de acordo com a carga de cada comutador, a exemplo de um balanceador de cargas para comutadores. Por fim, outra proposta é combinar a comutação por múltiplos caminhos proposta com a abordagem de TCP por múltiplos caminhos, permitindo fracionar fluxos TCP em fluxos menores e balancear tais fluxos na rede OpenFlow.

7. Referências

[Al-Fares et al. 2010] Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., and Vahdat, A. (2010). Hedera: Dynamic flow scheduling for data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pages 19–19. USENIX Association.

[Barabash et al. 2011] Barabash, K., Cohen, R., Hadas, D., Jain, V., Recio, R., and Rochwerger, B. (2011). A case for overlays in DCN virtualization. In *Proceedings of the 3rd Workshop on Data Center - Converged and Virtual Ethernet Switching*, DC-CaVES '11, pages 30–37. ITCP.

[Casado et al. 2007] Casado, M., Freedman, M., Pettit, J., Luo, J., McKeown, N., and Shenker, S. (2007). Ethane: Taking control of the enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4):1–12.

[Casado et al. 2012] Casado, M., Koponen, T., Shenker, S., and Tootoonchian, A. (2012). Fabric: a retrospective on evolving sdn. In *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, pages 85–90, New York, NY, USA. ACM.

[Couto et al. 2012a] Couto, R., Campista, M., and Costa, L. (2012a). A reliability analysis of datacenter topologies. IEEE Global Communications Conference - GLOBECOM.

[Couto et al. 2012b] Couto, R. d. S., Campista, M. E. M., and Costa, L. H. M. K. (2012b). Uma avaliação da robustez intra data centers baseada na topologia da rede. In *SBRC 2012*, Ouro Preto, MG.

[Duarte et al. 2012] Duarte, O. C. M. B., de Amorim, M. D., Costa, L. H., Rubinstein, M., Campista, M. E. M., and Florissi, P. (2012). Grandes massas de dados na nuvem: Desafios e técnicas para inovação. In *SBRC 2012 - Minicursos*.

[Gude et al. 2008] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). NOX: towards an operating system for networks. In *SIGCOMM Comput. Commun. Rev., 2008*, pages 105–110. ACM.

[Guedes et al. 2012] Guedes, D., Vieira, L., Vieira, M., Rodrigues, H., and Nunes, R. (2012). Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, pages 160–210.

[Guo et al. 2009] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., and Lu, S. (2009). BCube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM Comput. Commun. Rev., 2009*, pages 63–74. ACM.

[Hao et al. 2009] Hao, F., Lakshman, T., Mukherjee, S., and Song, H. (2009). Enhancing dynamic cloud-based services using network virtualization. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 37–44. ACM.

[Lantz et al. 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA. ACM.

[Li and Kwok 2005] Li, Z. and Kwok, Y.-K. (2005). A new multipath routing approach to enhancing tcp security in ad hoc wireless networks. In *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pages 372 – 379.

[Mattos et al. 2011] Mattos, D., Fernandes, N., da Costa, V., Cardoso, L., Campista, M., Costa, L., and Duarte, O. (2011). Omni: Openflow management infrastructure. In *Network of the Future (NOF), 2011 International Conference on the*, pages 52–56. IEEE.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev., 2008*.

[Mudigonda et al. 2010] Mudigonda, J., Yalagandula, P., Al-Fares, M., and Mogul, J. C. (2010). SPAIN: COTS data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10. USENIX Association.

[Nakagawa et al. 2012] Nakagawa, Y., Hyoudou, K., and Shimizu, T. (2012). A management method of ip multicast in overlay networks using openflow. In *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, pages 91–96, New York, NY, USA. ACM.

[Popa et al. 2010] Popa, L., Ratnasamy, S., Iannaccone, G., Krishnamurthy, A., and Stoica, I. (2010). A cost comparison of datacenter network architectures. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 16:1–16:12. ACM.

[Raiciu et al. 2011] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. (2011). Improving datacenter performance and robustness with multipath tcp. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, pages 266–277. ACM.

[Touch and Perlman 2009] Touch, J. and Perlman, R. (2009). Transparent interconnection of lots of links (TRILL): Problem and applicability statement.