REUNI: Um algoritmo para REduzir tabelas de encaminhamento e UNIformizar a distribuição dos fluxos em redes com topologia hipercubo

Dione S. A. Lima[1], Rafael S. Guimarães[12], Alextian B. Liberato[12]
Eros Spalla[2], Gilmar L. Vassoler[2], Magnos Martinello[1], Rodolfo S. Villaça[1] *

[1] Núcleo de Estudos em Redes Definidas por Software (NERDS)
Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES) – Vitória/ES

[2]Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo (IFES)
Campus Cachoeiro/ES, Colatina/ES, Aracruz/ES, Serra/ES – Brazil

dione.sousa@gmail.com, {rafaelg,alextian,spalla}@ifes.edu.br
gilmar@ifes.edu.br, magnos@inf.ufes.br, rodolfo.villaca@ufes.br

Abstract. *The hypercube is an interconnection topology of servers in a Data Center. In this topology, packet forwarding are normally based on XOR operation, which is highly efficient when the hypercupe is complete, but doesn't work when there are node or link failures. The use of SDN and OpenFlow can be viewed as an alternative to ensure traffic forwarding in these situations. However, the adoption of forwarding tables in a hypercube has a high cost associated to the huge number of entries in these tables, which increases as an exponential function of the number of nodes. In this context, this paper presents REUNI, an algorithm to reduce the number of entries in the forwarding table of a hypercube. REUNI achieves a logarithmic compression rate, which turns the number of entries equal to the number of dimensions of the hypercube. In addition to the compression rate, the distribution of the number of flows in the links was also evaluated and it was observed that the number of flows in a communication among all nodes are equally distributed between the links in the hypercube.*

Resumo. *Uma possível topologia de interligação dos servidores em um Data Center é o hipercubo. Nele, o encaminhamento de pacotes é normalmente baseado em operações de XOR e é bastante eficiente, porém só funciona na ausência de falhas de nós ou enlaces no hipercubo. O uso de SDN e do protocolo OpenFlow pode ser uma alternativa para garantir o encaminhamento de tráfego nessas situações. Entretanto, a adoção de tabelas de encaminhamento em um hipercubo possui um alto custo, relacionado ao grande número de entradas nessas tabelas, que crescem em escala exponencial. Nesse contexto, este artigo apresenta o REUNI, um algoritmo para redução do número de entradas dessas tabelas com taxa de compactação logarítmica, proporcional ao número*

*Este trabalho tem recebido financiamento do projeto Horizon 2020 da União Européia para pesquisa, desenvolvimento tecnológico e demonstração sob no. 688941 (FUTEBOL), assim como do Ministério Brasileiro da Ciência, Tecnologia e Inovação (MCTI) por meio da RNP e do CTIC. Além disso, gostaríamos de agradecer o financiamento do CNPq sob no. 456143/2014-9 e 449369/20145, e da FAPES sob no. 524/2015.

de dimensões do hipercubo. Além da taxa de compactação, avaliou-se também a distribuição dos fluxos nos enlaces após a compactação e observou-se que após a compactação das tabelas ocorre um balanceamento natural das rotas nos enlaces do hipercubo em uma comunicação de todos para todos.

1. Introdução

Data Centers são bastante utilizados no processamento e armazenamento de grandes volumes de dados. Em função disso, existem diversas arquiteturas para a montagem da sua infraestrutura tais como: Dcell [Guo et al. 2008], BCube [Guo et al. 2009] e Fat-Tree [Al-Fares et al. 2008]. Em particular, nas arquiteturas de *Data Centers* centrados em servidores (*server-centric*), a característica mais importante é que os elementos de rede (*switches*, por exemplo) não estão envolvidos diretamente nas decisões de encaminhamento de tráfego de rede, eles simplesmente fornecem conexões entre os servidores, e estes sim, são os responsáveis por todas as decisões de encaminhamento no interior do *Data Center*. Nesse contexto, equipamentos de rede de baixo custo (COTS, ou *Commodity Off-the-Shelf*) podem ser utilizados na montagem da infraestrutura, até mesmo ligações diretas entre os servidores podem ser estabelecidas nestes *Data Centers*.

Uma possível topologia de interligação dos servidores nas arquiteturas de *Data Center* centrados em servidores é o hipercubo, e neste contexto podemos destacar o TRI-IIAD, proposto em [Vassoler 2015]. Em um hipercubo cada servidor (que também pode ser chamado de um nó da rede do *Data Center*) possui um identificador único de n-bits e n interfaces de rede, que o interliga a outros n servidores, que serão os seus vizinhos no *Data Center*. A restrição na escolha desses vizinhos é tal que cada servidor só poderá ter como vizinhos outros servidores cuja distância de Hamming (D_h) entre seus identificadores é igual a 1. Essa restrição facilita o processo de encaminhamento de tráfego no interior do *Data Center*, onde basta uma simples operação de ou-exclusivo (XOR) para determinação da interface de saída para a qual os pacotes deverão ser encaminhados a fim de alcançarem o seu destino.

Embora o encaminhamento baseado em operações XOR seja bastante eficiente [Dally 1990, Vassoler 2015], o mesmo só funciona em situações onde os hipercubos virtuais estão completos, ou seja, na ausência de falhas de nós ou de enlaces no *Data Center*. Na ocorrência de uma falha como essas o resultado da operação XOR poderá indicar uma interface de saída que leve o pacote a um nó inoperante ou a um enlace indisponível, o que irá ocasionar perda de pacotes e quebra da completa conectividade entre os nós do *Data Center* enquanto a falha estiver presente. Em um trabalho anterior [de Lima et al. 2015] apresentamos uma abordagem baseada em Redes Definidas por Software (SDN, ou *Software Defined Networking*), especificamente no protocolo OpenFlow [McKeown et al. 2008], para garantir o encaminhamento de tráfego em *Data Centers* centrados nos servidores ligados na topologia de hipercubo, mesmo em situações de falha.

Entretanto, a adoção de tabelas de encaminhamento em *Data Centers* com topologia baseada em hipercubo possui um alto custo relacionado ao grande número de entradas nessas tabelas, que tende a crescer exponencialmente em função de seu grau (n). Em um *Data Center* com topologia em hipercubo existe uma enorme multiplicidade de caminhos entre quaisquer pares origem-destino [Saad and Schultz 1989], mas que o uso de encaminhamento sem estado (XOR) traz problemas relacionados com a tolerância à

falhas [de Lima et al. 2015]. Além disso, a adoção do XOR como algoritmo de encaminhamento também provoca um forte desbalanceamento na distribuição de tráfego entre os enlaces disponíveis no *Data Center*. Estes resultados serão apresentados na Seção 4 deste artigo.

Desta forma, baseando-se nos trabalhos propostos em [Vassoler 2015, de Lima et al. 2015], este artigo apresenta o REUNI, um algoritmo para redução do número de entradas nas tabelas de encaminhamento dos nós de um *Data Center* centrado em servidores com topologia hipercubo. A proposta apresentada neste artigo faz uso de máscaras de bit (*bitmaks*), característica disponível no OpenFlow a partir de sua versão 1.1, e de técnicas de compactação de tabelas de encaminhamento baseadas na correspondência pelo prefixo mais longo. O REUNI alcança uma taxa de compactação de escala logarítmica (igual a $log_2(2^n) = n$), independentemente da existência de virtualização de servidores, além de propiciar uma melhor distribuição dos fluxos nos enlaces do hipercubo com a garantia de encaminhamento sempre pelo caminho mais curto (SPF, ou *Shortest Path First*).

Para validar a proposta foram feitos comparativos entre o número de entradas nas tabelas de encaminhamento de *Data Centers* em topologia hipercubo, compatíveis com as propostas de [Vassoler 2015, de Lima et al. 2015], com $3 \leq n \leq 10$, com e sem compactação nas tabelas, com e sem virtualização de servidores. Para comprovar a distribuição de carga desbalanceada entre os enlaces quando se utiliza encaminhamento sem estado baseado em XOR, avaliou-se a distribuição do número de fluxos em cada enlace do hipercubo em cenários de comunicação de todos para todos os servidores. Em seguida, a avaliação foi rodada novamente considerando-se as tabelas compactadas com o algoritmo proposto. Por fim, avaliou-se o tempo gasto para compactação de todas as tabelas para todos os servidores.

A principal contribuição deste artigo é o algoritmo de compactação das tabelas de encaminhamento (REUNI), que garante a redução do número de entradas em escala logarítmica com garantia de entrega sempre pelo menor caminho. Na Seção 2 são apresentados importantes trabalhos relacionados e suas contribuições. O algoritmo proposto e detalhes de sua implementação estão apresentados na Seção 3. A Seção 4 apresenta os resultados das avaliações feitas no protótipo desenvolvido com hipercubos com até 1024 nós. Por fim, na Seção 5 são tecidas as conclusões e trabalhos futuros para aprimoramento da proposta.

2. Trabalhos Relacionados

O maior problema relacionado com o uso da topologia hipercubo em *Data Centers* centrados em servidores é a sua incapacidade no tratamento de falhas usando a operação XOR como método de encaminhamento, apesar de sua baixíssima latência [Vencioneck et al. 2014]. A alternativa de uso de tabelas de encaminhamento garante a conectividade nesses casos, porém tem um alto custo associado ao seu crescimento exponencial em função do número de nós do hipercubo.

TRIIIAD [Vassoler 2015] é um trabalho relacionado do grupo NERDS que, dentre outras coisas, demonstra a viabilidade e as vantagens do uso de *Data Centers* programáveis com topologia hipercubo. Na arquitetura TRIIIAD os nós do hipercubo são servidores de grande porte que hospedam máquinas virtuais (VMs) e suas aplicações. Todo o controle de migrações de VMs e orquestração dos fluxos rede é baseado em

controladores SDN em um conceito denominado A-SDN *(Augmented SDN)*, inclusive permitindo a troca do método de encaminhamento de cada nó "*on the fly*" usando o FlexForward [Vencioneck et al. 2014]. Toda essa flexibilidade permite que *Data Centers* com poucos nós físicos possam aproveitar-se das vantagens da topologia hipercubo, que são: baixíssima latência de encaminhamento, alto grau de redundância nos caminhos entre os servidores. O REUNI contribui nesse contexto reduzindo, em escala logarítmica, o tamanho das tabelas de encaminhamento dos nós OpenFlow do *Data Center* TRIIIAD, viabilizando sua implementação em ambientes reais.

O BCube [Guo et al. 2009] e o DCell [Guo et al. 2008] são propostas de *Data Center* centrados em servidores modulares mais citada na literatura, cuja topologia é bastante similar ao hipercubo. Uma vez que a topologia não é um hipercubo original, a ocorrência de falhas não ocasiona interrupção de conexão entre os servidores, porém aumenta-se o tamanho médio dos caminhos entre eles e reduz-se a quantidade de opções de caminhos entre os servidores.

Em [Luiz Fernando T. de Farias 2014] é apresentado um algoritmo para compactação de tabelas de roteamento sob a ótica da interface de saída em roteadores IP. Embora também usem a estratégia do prefixo mais longo para agrupar rotas, essa agregação ocorre nos endereços IP. Em [Braun and Menth 2014] é apresentado um trabalho com proposta similar ao de [Luiz Fernando T. de Farias 2014], também com aplicação em roteadores BGP e com o objetivo de reduzir o número de entradas BGP nos roteadores das redes de acesso. Ambos se diferenciam por considerarem a agregação no nível de rede (IP), enquanto que o REUNI aplica a redução em endereços MAC, usados pela arquitetura TRIIIAD para endereçar nós e VMs, conforme descrição na Seção 3.

Considerando os trabalhos relacionados previamente citados, o REUNI avança o estado da arte no contexto dos *Data Centers* centrados em servidores ao viabilizar o uso da topologia hipercubo em *Data Centers* de pequeno e médio porte em redes reais. Essa viabilização dar-se-á pelo alto grau de compactação das tabelas de encaminhamento dos nós e por, naturalmente, colaborar para uma melhor distribuição dos fluxos nessas redes.

3. Implementação

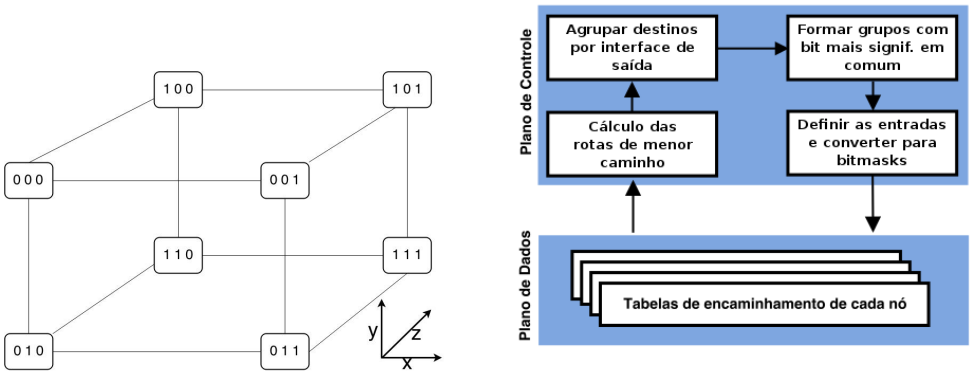
A solução proposta neste artigo compreende os dois aspectos fundamentais das SDNs: plano de controle e plano de dados. No plano de controle serão descritas as características da aplicação no controlador, abordando detalhadamente o REUNI. Dentre as ações realizadas por essa aplicação, destacam-se:

1. Representação topológica do hipercubo;
2. Execução do cálculo das rotas identificando todos os possíveis menores caminhos através do algoritmo (*Shortest Path First*);
3. Implementação do REUNI;
4. Instalação das regras OpenFlow utilizando *bitmasks* no plano de dados.

No plano de dados serão tratadas as questões referentes a instalação das regras de encaminhamento, utilizando-se dos *bitmasks*, característica implementada no OpenFlow 1.3 que permite a utilização de regras curinga (*).

3.1. Plano de Controle

No Plano de Controle a aplicação faz uso de uma representação topológica do hipercubo por meio de uma estrutura de dados do tipo grafo. Para tal, foi utilizada a biblioteca *NetworkX*[1]. A Fig. 1(a) representa um hipercubo grau 3, que será usado para exemplificar o funcionamento do REUNI, onde o bit menos significativo representa o eixo *x*, o segundo bit representa o eixo *y* e o bit mais significativo representa o eixo *z*. Como podemos observar todos os nós são identificados obedecendo o critério da Distância de Hamming (D_h) igual a 1 entre quaisquer pares de nós vizinhos.



(a) (b)

Figura 1. Visão geral: (a) Hipercubo com três dimensões e (b) Construção da aplicação.

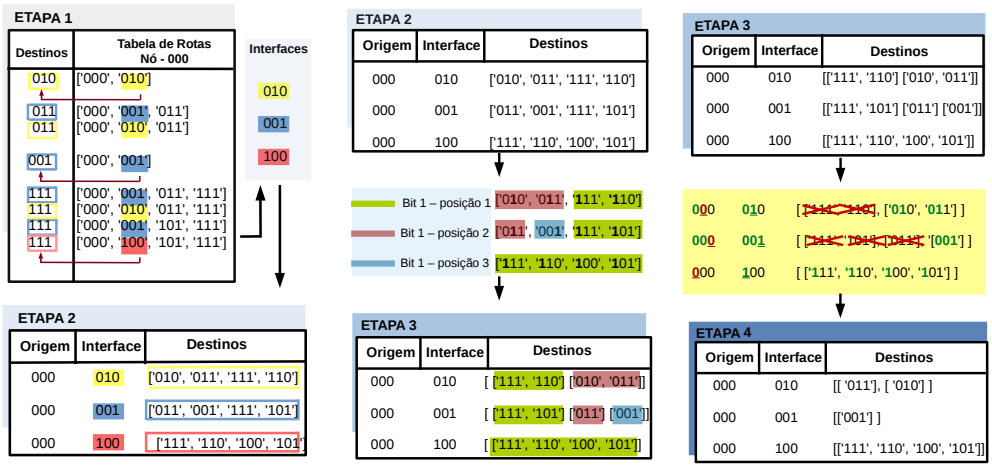
A aplicação no plano de controle será responsável pelo cálculo das rota entre quaisquer pares de origem e destino, identificando, inicialmente, todos os possíveis menores caminhos entre eles. Em seguida as tabelas de encaminhamento de cada nó do hipercubo serão submetidas ao REUNI para compactação. A partir das tabelas compactadas são definidas as novas regras de encaminhamento que serão instaladas em cada nó do hipercubo de acordo com as características e restrições do protocolo OpenFlow. A Figura 1(b) mostra um resumo desse processo, onde o resultado de uma etapa torna-se a entrada para a etapa seguinte. Cada etapa será detalhada a seguir.

A Figura 2 exemplifica todo o processo a partir do nó de origem 000. Na Etapa 1 são geradas as tabelas de encaminhamento de cada nó do hipercubo com todos os caminhos para todos os possíveis destinos do hipercubo. De acordo com a Fig. 2(a), tem-se na primeira coluna da tabela de rotas do nó 000 todos os possíveis destinos no hipercubo a partir desta origem. Por uma questão de limitação de espaço, a tabela da Etapa 1 foi reduzida. Para cada destino, na segunda coluna, são mostradas todas as menores rotas, representadas pelos nós que compõem esse caminho. Por exemplo, a partir nó 000 e com destino ao nó 010 temos a rota $[000, 010]$, ou seja, o destino está diretamente ligado a origem 000 e consequentemente ambos são vizinhos no hipercubo (observe que a D_H entre os identificadores de origem e destino é igual a 1). A partir do mesmo nó 000, porém com destino a 011, por exemplo, observa-se dois caminhos de mesmo custo, que são $[000, 001, 011]$ e $[000, 010, 011]$ (observe que ambos destinos possuem $D_h = 2$ com

[1] http://networkx.github.io

a origem). Isso mostra que esses nós podem se comunicar usando tanto o vizinho 001, quanto o vizinho 010 como próximo salto. A escolha do vizinho para qual o pacote deverá ser encaminhado representa uma interface de saída no hipercubo da Figura 1(a). É usado o termo interface de saída para representar o próximo salto a partir de um nó. A geração desta tabela com todas as possíveis rotas entre origens e destinos corresponde à Etapa 1 do REUNI.

O objetivo da Etapa 2 é: para cada nó, agrupar todos os seus possíveis destinos de acordo com a interface de saída utilizada para alcançá-los, ou seja, agrupa-se os destinos de acordo com o vizinho para o qual o pacote será encaminhado. Para o nó 000, cujas "interfaces" de saída são 010, 001 e 100, são verificados quais os destinos que utilizarão cada um desses vizinhos no encaminhamento dos pacotes a partir do nó 000. Podemos observar na tabela "ETAPA 1" da Fig. 2(a) a identificação das interfaces realçadas na coluna do meio e os destinos sendo associados às mesmas. Em seguida, para cada interface, o REUNI vai agrupar todos os destinos que serão alcançados através de uma mesma interface. Como resultado para o nó 000, temos todas as suas interfaces de saída com suas respectivas listas de destinos. É importante ressaltar aqui que esta é uma grande vantagem da topologia hipercubo: a multiplicidade de caminhos de menor custo entre quaisquer pares de nós da rede.



(a) REUNI - Etapas 1 e 2 (b) REUNI - Etapa 3 (c) REUNI - Etapa 4

Figura 2. REUNI - Etapas 1 a 4

Na Etapa 3, uma lista de destinos é associada a cada interface de saída. Para cada lista é feita uma verificação da posição do bit mais significativo com valor 1 de cada endereço de destino na lista, como pode ser observado na Figura 2(b). Os endereços são agrupados de acordo com essa posição. Por exemplo, observando a lista de destinos pertencentes a interface 010$[[010, 011, 111, 110]]$, o REUNI verifica primeiramente a posição mais significativa do bit 1 de cada endereço da lista e ao encontrar endereços com as mesmas características, adiciona-os em um lista específica que será composta pelos endereços $[111, 110]$. A lista seguinte, $[010, 011]$, é composta com os endereços onde o primeiro bit 1 ocupa a segunda posição mais significativa, e assim sucessivamente, até atingir a quantidade de bits que compõem o endereço. A Função `terceiraEtapa` no Algoritmo 1 representa esse processo. É possível verificar que no exemplo escolhido (nó 000, interface 010) o REUNI não apresenta uma lista cujos bits ocupam a terceira posição

mais significativa, pois a lista de destinos da interface 010 não contém endereços com essa característica.

Algoritmo 1: Terceira e Quarta Etapas

```
Função terceiraEtapa (L_D)
início
    Entrada: L_D ← segundaEtapa()
    Saída: Lista Final com listas de destinos agregados pelo bit mais significativo
    L_1, L_2, ..., L_N
    para cada D_L ∈ L_D faça
        se Operação de bit ∧ D_l na posição N == VERDADEIRO então
            adiciona D_L à lista correspondente da posição L_N
        fim se
    fim para cada
    retorna L_1, L_2, ..., L_N
fim
Função quartaEtapa (L_D)
início
    Entrada: L_D ← terceiraEtapa()
    Saída: Lista Final com lista destinos por interface
    Li_1, Li_2, ..., Li_N
    para cada D_L ∈ L_D faça
        se D_L com maior número de endereços então
            LMaiorTamanho ← D_L
            adiciona LMaiorTamanho a lista de destinos por interface Li_N
        senão
            LAdicional ← (D_L − LMaiorTamanho)
            LFinal ← bit mais significativo da interface Li_N ∧ LAdicional
            adiciona LFinal a lista de destinos por interface Li_N
        fim se
    fim para cada
    retorna Li_1, Li_2, ..., Li_N
fim
```

Na Etapa 4, o REUNI faz uma verificação nas listas de todas as interfaces do seu respectivo nó, no caso do nosso exemplo, do nó 000, para identificar os endereços de destinos que aparecem em listas de mais de uma interface. É possível notar na tabela "ETAPA 1" da Fig. 2(a) que alguns destinos podem ser alcançados por meio de várias interfaces de saída por rotas distintas, ou seja, através de diferentes interfaces. Com isso, torna-se necessário escolher por qual interface aquele destino será alcançado, ou seja, em qual lista o mesmo deverá permanecer. Esse é o objetivo da Etapa 4, representado na Função `quartaEtapa` do Algoritmo 1.

Primeiramente o REUNI verifica qual a interface contem a lista com maior quantidade de endereços de destinos com o bit mais significativo, com valor 1, na mesma posição. No nosso exemplo, a interface 100 contém uma lista com essa característica, onde todos os seus destinos apresentam o valor 1 no bit mais significativo. Esses destinos formam uma nova lista (`LMaiorTamanho` no algoritmo). Em seguida, as listas de destino das outras interfaces são verificadas a fim de eliminar os endereços de destino em comum com essa nova lista, conforme é possíveis visualizar na Figura 2(c).

Por último é feita uma comparação bit a bit entre o endereço do nó de origem, no caso o 000 com o endereço de cada interface. Essa comparação resulta na composição de um prefixo que corresponde aos bits em comum entre o endereço do nó de origem e o endereço da interface mais o primeiro bit diferente do endereço da interface. Por exemplo, comparando o endereço de origem 000, da esquerda para a direita, com o endereço da interface 010, percebe-se que tanto o primeiro bit da origem quanto o primeiro bit do endereço da interface são iguais, então o mesmo será o primeiro bit utilizado para compor

o prefixo. Verifíca-se agora o segundo bit dos dois endereços, onde os mesmos são diferentes. Dessa forma o prefixo será composto pelo bit em comum entre os dois endereços mais o bit diferente do endereço da interface, resultando no prefixo 01. Na comparação, ao encontrar o primeiro bit diferente no endereço da interface, o mesmo será o último bit a compor o prefixo e indicará o fim da comparação dos bits dos endereços em questão. Com isso, todos os endereços pertencentes a lista da interface 010 que iniciam com esse prefixo permanecerão na lista. Os outros endereços serão excluídos. Dessa forma o REUNI provê a tabela compactada para cada nó do hipercubo.

3.2. Plano de Dados

Como pôde ser observado na Figura 1(a), cada nó no hipercubo possui um endereço binário que representa seu identificador e localizador no hipercubo. Na prática, na arquitetura TRIIIAD o endereço de enlace (*MAC Address*) é usado para identificação e localização de cada nó. Como cada nó representa servidores físicos que abrigam máquinas virtuais, uma parte do endereço MAC é utilizado para sua identificação e localização do nó no hipercubo (servidor físico) e outra para a identificação das VMs no hipercubo, conforme visualizado na Fig. 3. Os 32 bits mais significativos são utilizados para caracterizar o servidor físico e outros 16 bits identificam as máquinas virtuais hospedadas neste servidor.

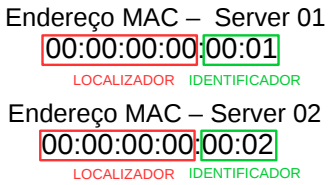


Figura 3. Endereçamento no hipercubo.

Para realizar a instalação de regras OpenFlow nas tabelas de encaminhamento dos nós utiliza-se a característica dos *bitmasks*, onde apenas parte dos endereços MAC é utilizado durante a busca por uma entrada na tabela que faça *matching* com um endereço de destino. Por exemplo, regras de encaminhamento para os endereços MAC 00:00:00:07:00:01 e 00:00:00:07:00:02, podem ser criadas com apenas 1 (uma) entrada, que será representado por 00:00:00:07:00:00/ff:ff:ff:ff:00:00. O funcionamento da máscara determina quais bits deverão ser utilizados durante o processo de verificação *matching*. Neste exemplo apenas os 32 primeiros bits da entrada na tabela deverão coincidir (*match*) com o endereço de destino para o qual se deseja encaminhar um pacote. Para o nó 000 da Figura 1(a) as entradas OpenFlow compactadas com REUNI e instaladas no plano de dados usando *bitmasks*, são apresentadas na Tab. 1.

Tabela 1. Exemplo das Regras com Bitmask no OF 1.3.

Seq.	Descrição da Regra	Ação	Interface
Entrada 1	prioridade=3, dl dst=00:00:00:04:00:00/ff:ff:ff:ff:00:00	output 4	100
Entrada 2	prioridade=2, dl dst=00:00:00:02:00:00/ff:ff:ff:ff:00:00	output 3	010
Entrada 3	prioridade=1, dl dst=00:00:00:01:00:00/ff:ff:ff:ff:00:00	output 2	001

4. Resultados e Discussões

Nesta seção o algoritmo de compactação das tabelas de encaminhamento é avaliado e os resultados são apresentados e discutidos. Para melhor organização esta seção foi subdividida em 4 (quatro) subseções, descritas a seguir.

A Seção 4.1 descreve as características do ambiente de experimentação e os procedimentos metodológicos de avaliação. Em seguida, na Seção 4.2, o algoritmo é avaliado em função da quantidade de entradas na tabela de encaminhamento de um nó em condições normais de funcionamento. Nesta etapa, surgiu então um questionamento relevante ao mecanismo proposto: dada a característica do algoritmo proposto de agregar vários destinos em um mesmo vizinho, será que a compactação das tabelas acarretaria em uma distribuição irregular da quantidade de fluxos que passam por cada um dos enlaces do hipercubo em uma comunicação de todos para todos?

Portanto, na Seção 4.3 avaliou-se a distribuição da quantidade de fluxos por enlace em um hipercubo completo com encaminhamento baseado em XOR. A avaliação foi efetuada levando-se em conta diferentes tamanhos de hipercubo a fim de se fazer um comparativo do comportamento da distribuição dos fluxos sem compactação (com encaminhamento XOR) e com compactação de tabelas (com encaminhamento OpenFlow). Por último, na Seção 4.4 foi efetuada uma avaliação da vazão em um hipercubo com 8 nós sem compactação de tabelas e com compactação de tabelas com o objetivo de avaliar o efeito da compactação da tabela e da redistribuição dos fluxos entre os enlaces no comportamento da vazão média dos fluxos no hipercubo.

4.1. Ambiente de Experimentação

Todo o protótipo foi emulado no Mininet, versão 2.2.1, tendo como máquina de encaminhamento o Open vSwitch [Pfaff et al. 2015] versão 2.3 e OpenFlow 1.3 [McKeown et al. 2008] como protocolo de comunicação entre os planos de dados e de controle. Para instalação das regras de encaminhamento no plano de dados utilizou-se o controlador Ryu, versão 3.23. Como ambiente de experimentação utilizou-se um *host* físico com sistema operacional Linux Ubuntu 14.04 64 bits, processador Core i5 3.2GHz, 6GB RAM. O algoritmo de compactação foi implementando utilizando-se o Python 2.7.

Para avaliar a vazão usou-se a ferramenta Iperf[2] versão 2.05. A configuração topológica ilustrada na Figura 1(a) representa um hipercubo de grau 3 (com 8 nós), usado como referência na maior parte dos experimentos.

4.2. Compactação das Tabelas de Encaminhamento

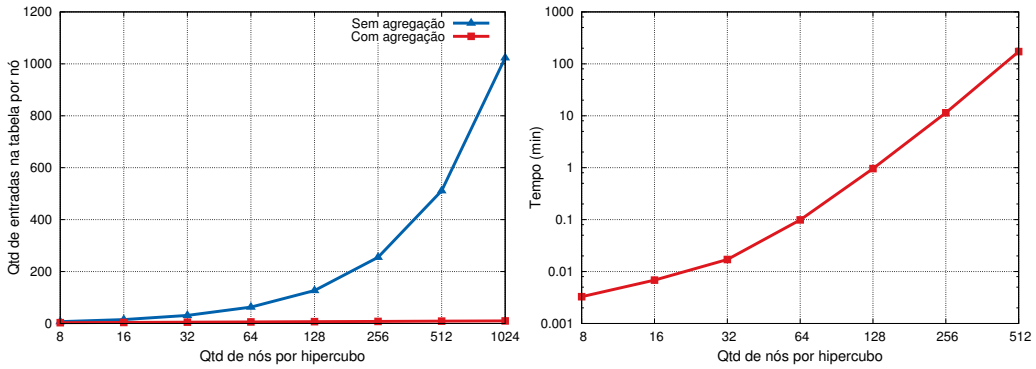
Neste experimento a quantidade de entradas nas tabelas de encaminhamento dos nós do hipercubo foi avaliada. O objetivo deste experimento é avaliar o impacto do algoritmo de compactação no tamanho das tabelas em cada nó. Desta forma avaliou-se a quantidade de entradas sem compactação e com compactação, usando o algoritmo proposto, em hipercubos completos de grau $\{3, 4, 5, 6, 7, 8, 9$ e $10\}$.

A Figura 4(a) apresenta os resultados obtidos neste experimento. Observe que à medida em que aumenta-se o tamanho do hipercubo há um incremento exponencial na quantidade de entradas na tabela de encaminhamento dos nós. Esse fato se deve à característica topológica, onde a quantidade de entradas ($QtdEntradas$) na tabela de um

[2]https://iperf.fr/

nó é igual a 2^n-1, onde n representa o grau do hipercubo. Ou seja, para um hipercubo de grau 10 tem-se: $QtdEntradas = 2^{10}-1 = 1024 - 1 = 1023$ entradas em cada nó da topologia. Não se considera, nesta avaliação, a possibilidade de existirem Máquinas Virtuais (VMs) ligadas aos nós do hipercubo, o que acarretaria mais entradas nas tabelas de encaminhamento dos nós sem compactação. Em contraste, a quantidade de entradas nas tabelas de encaminhamento permanece inalterada mesmo com a adição de VMs aos nós do hipercubo devido ao uso dos *bitmasks* do protocolo OpenFlow.

Se houver a utilização de VMs nos nós do hipercubo, no cenário sem compactação o número médio de entradas na tabela de encaminhamento em um nó será dado por $QtdEntradas = (2^{10}-1) \cdot m$, onde m corresponde ao número médio de VMs existentes em cada nó.



(a) (b)

Figura 4. Análise em função da quantidade de nós: (a) Número de entradas na tabela de encaminhamento de cada nó e (b) Tempo computacional para compactação.

Ainda na Figura 4(a) observa-se que com a compactação da tabela o incremento na quantidade de entradas é linear, onde tem-se: $QtdEntradas = n$. Em comparação ao exemplo anterior, para o hipercubo de grau 10 haverão apenas 10 entradas na tabela de encaminhamento, o que representa uma redução superior a 99% no número de entradas. A Figura 4(b) traz uma relação inicial do tempo necessário para compactação de todos os nós do hipercubo usando o REUNI. Observa-se que a complexidade da compactação é exponencial, mas é importante ressaltar que a ação de compactação pode ser feita *offline*, não acarretando prejuízo de desempenho para o *Data Center*.

4.3. Distribuição dos Fluxos entre os Enlaces

Conforme apresentado na Seção 3, a compactação da tabela de encaminhamento de um nó prioriza a agregação das entradas utilizando-se do conjunto completo dos k menores caminhos para cada destino. Contudo, dada a alta taxa de compactação do algoritmo proposto, especulou-se que tal resultado poderia impactar de forma negativa na distribuição dos fluxos nos enlaces do hipercubo. Por consequência, decidiu-se avaliar o comportamento da distribuição dos fluxos nos enlaces dos hipercubos de grau $\{3, 4, 5, 6, 7$ e $8\}$ durante a comunicação entre todos os pares origem e destino do hipercubo. Esta avaliação e a próxima (Seção 4.4) não foram realizadas para os graus $\{9$ e $10\}$ devido a limitações no hardware utilizado, o que estava ocasionando um alto tempo computacional para conclusão dos testes.

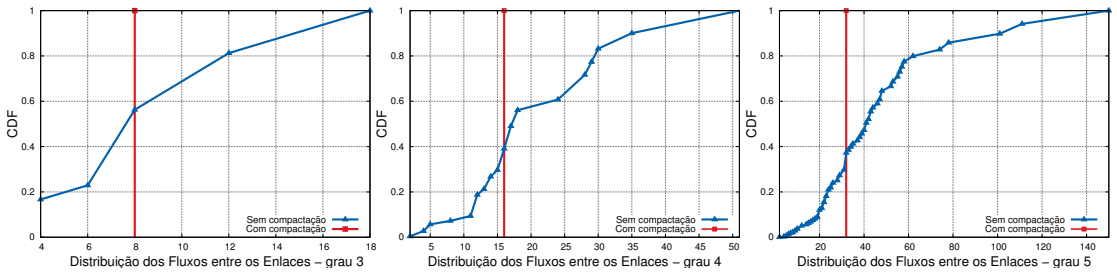
Os resultados são apresentados da seguinte maneira: é sabido que na topologia hipercubo todos os nós têm a mesma quantidade de enlaces incidentes, que é o grau do hipercubo (n). Dessa forma, a equação $QtdEnlaces = 2^{n-1} \cdot n$ define a quantidade total de enlaces em um hipercubo de grau n considerando-se todos os pares origem e destino. Além disso, para realizar a comunicação entre todos os pares origem e destino de um hipercubo de grau n produz-se um total de fluxos pode ser calculado pela Equação 1

$$QtdFluxos = \sum_{d=1}^{n} \frac{n!}{(n-d)d!} \cdot d \tag{1}$$

Conhecendo-se o total de fluxos, a distribuição ideal da quantidade de fluxos por enlace seria igual à média, portanto igual a $\frac{QtdFluxos}{QtdEnlaces} = 2^n$, que corresponde ao número de nós na rede. Desta forma, a distribuição ideal de fluxos para um hipercubo de grau 3 é de 8 fluxos por enlace, para um hipercubo de grau 4 é de 16 fluxos por enlace, e assim por diante.

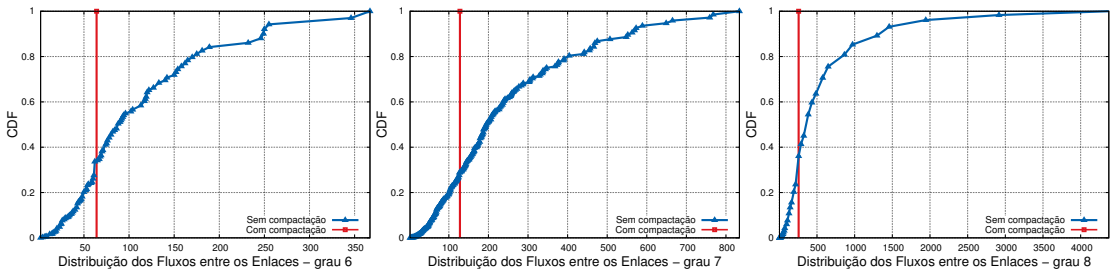
As Figuras 5 e 6 mostram a distribuição acumulada (CDF, ou *Cumulative Distribution Function*) da quantidade de fluxos por enlace sem compactação (usando encaminhamento XOR) e com compactação (segundo o algoritmo proposto), para hipercubos de grau $\{3, 4 \text{ e } 5\}$ e $\{6, 7 \text{ e } 8\}$, respectivamente. Como exemplo, na Figura 5(c) é possível observar um desbalanceamento na distribuição dos fluxos no cenário sem compactação (usando XOR), pois em mais de 60% dos enlaces existem mais fluxos do que o ideal. Entretanto, quando a compactação das tabelas de encaminhamento é utilizada, tem-se uma distribuição uniforme dos fluxos, ou seja, em 100% dos enlaces passam 32 fluxos na comunicação de todos para todos os nós do hipercubo.

Esse desbalanceamento natural dos fluxos usando XOR se repete em todos os casos avaliados. Em outro exemplo, no hipercubo de grau 4 apresentado na Figura 5(b), cerca de 61% dos enlaces passam mais de 16 fluxos no cenário sem compactação, ou seja, mais da metade dos enlaces teriam mais fluxos que o ideal. Esse resultado significa que em um total de 12 enlaces existem entre 17 e 51 fluxos. No total, 312 fluxos da comunicação de todos para todos seriam prejudicados por essa distribuição irregular sem a agregação das entradas. Contudo, com a compactação, novamente a distribuição é uniforme com 100% dos enlaces.



(a) (b) (c)

Figura 5. Análise da distribuição dos fluxos entre os enlaces no hipercubo: (a) com grau 3, (b) com grau 4 e (c) com grau 5.



(a) (b) (c)

Figura 6. Análise da distribuição dos fluxos entre os enlaces no hipercubo: (a) com grau 6, (b) com grau 7 e (c) com grau 8.

Na Figura 6(a) (grau 6), em cerca de 67%, (66 enlaces) trafegam mais fluxos do que a média em uma comunicação de todos para todos. Isso representa um total de 8152 fluxos disputando os 66 enlaces, com distribuição variando entre 65 e 367 fluxos por enlace. Contudo, após a compactação, além de reduzir a quantidade de entradas nas tabelas, observa-se uma distribuição equilibrada dos fluxos entre os enlaces do hipercubo, com exatamente 64 fluxos para cada um dos 192 enlaces. Esse comportamento repete-se nas Figuras 6(b) e 6(c).

Portanto, após avaliar o comportamento da distribuição dos fluxos, observou-se um comportamento completamente contrário ao esperado, ou seja, o algoritmo de compactação além de reduzir em escala logarítmica a quantidade de entradas nas tabelas de encaminhamento dos nós do hipercubo, também distribui os menores caminhos de maneira uniforme entre todos os enlaces da topologia. Esse resultado é devido ao fato de que o algoritmo considera todo o conjunto de menores caminhos entre todas as origens e destinos no hipercubo durante a compactação. Esse resultado leva a um balanceamento de carga natural entre os enlaces do hipercubo, facilitando dessa forma ações futuras de engenharia de tráfego em função das aplicações nos nós.

4.4. Comparativo da Vazão nos Enlaces com e sem Compactação das Tabelas

Neste experimento, o objetivo é avaliar o impacto da distribuição uniforme dos fluxos nos enlaces, ou seja, comprovar se essa uniformidade na distribuição dos fluxos beneficia a vazão em uma comunicação entre todos os pares origem e destino.

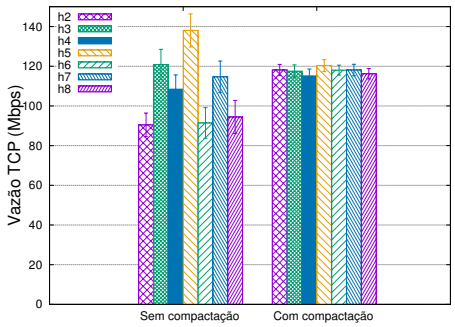


Figura 7. Comparativo da vazão (Mbps) com e sem compactação.

A Figura 7 apresenta a vazão de um *host* no nó $h1$ enviando simultaneamente tráfego para os outros 7 *hosts* em um hipercubo com a topologia da Figura 1(a) e enlaces de 1Gbps. Observa-se que após compactação das tabelas de encaminhamento de todos os

nós, a vazão também tende a ser uniforme. A pequena variabilidade apresentada é justificada devido ao mecanismo de controle de congestionamento do protocolo TCP. Neste experimento, todos os nós foram definidos como receptores de tráfego TCP gerado no Iperf e as medidas foram coletadas no cliente. Coletaram-se 30 amostras com duração de 10s cada amostra. A variabilidade foi obtida respeitando-se um intervalo de confiança de 95%. Esse comportamento se mantém para outros tamanhos de hipercubo, mas não foram apresentados aqui por limitações de espaço.

De acordo com os resultados obtidos nos experimentos, a distribuição uniforme dos fluxos aumenta a vazão média do hipercubo em cerca de 8%, além de apresentar uma menor variabilidade e, consequentemente, prover uma comunicação mais estável e com menor *jitter*.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou o REUNI, uma proposta de algoritmo que reduz em mais de 99% a quantidade de entradas nas tabelas de encaminhamento e uniformiza a distribuição dos fluxos, em uma comunicação de todos para todos, em 100% dos enlaces, ambos no plano de dados. Uma descrição detalhada do algoritmo foi apresentada e a proposta fora avaliada com hipercubos de grau $\{3, 4, 5, 6, 7, 8\}$, contribuindo para facilitar a adoção dessa topologia em *Data Centers* centrados nos servidores, dado que o crescimento exponencial das tabelas de encaminhamento era um problema importante nessa topologia. Foi implementado um protótipo usando o Mininet, controlador Ryu e OpenFlow 1.3 como prova de conceito.

O REUNI alcança uma taxa de compactação logarítmica das tabelas de encaminhamento dos nós, que passam a ter $log_2(2^n) = n$ entradas, independentemente da existência de virtualização de servidores no *Data Center*. Em todos os casos avaliados observou-se que a distribuição dos fluxos utilizando operações XOR apresenta um desbalanceamento na distribuição da carga de tráfego entre os enlaces. No pior caso, para o hipercubo de grau 7, cerca de 71% dos enlaces ficam sobrecarregados com mais de 128 fluxos, totalizando 40853 fluxos distribuídos em 159 enlaces. Os resultados de desempenho apresentados neste artigo, efetuados em um ambiente emulado, indicam que a distribuição uniforme dos fluxos entre os enlaces realizada pelo REUNI é capaz de melhorar a vazão média do hipercubo em cerca de 8%.

Os próximos passos no desenvolvimento deste trabalho serão: i) otimizar o processo de agregação das regras de entrada da tabela. Para isso, estamos trabalhando para reduzir e/ou concatenar etapas do algoritmo proposto, o que impactará na redução do tempo computacional; ii) integrar o REUNI com o FlexForward [Vencioneck et al. 2014], implementado no Open vSwitch, o que permitirá aumentar a vazão em cada nó do hipercubo e a mudança automática da forma de encaminhamento dos pacotes em caso de falhas; iii) avaliar a implementação no *Data Center* do NERDS (Núcleo de Estudos em Redes Definidas por Software, UFES), em ambiente real.

Durante os experimentos foi descoberto que o uso dos *bitmasks* no Open vSwitch força o uso do espaço do usuário (*user space*) no encaminhamento dos pacotes, o que limita a vazão de encaminhamento. Superar essa limitação do Open vSwitch (e não do protótipo) também é um possível trabalho futuro.

Referências

Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74.

Braun, W. and Menth, M. (2014). Wildcard compression of inter-domain routing tables for openflow-based software-defined networking. In *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, pages 25–30.

Dally, W. (1990). Performance analysis of k-ary n-cube interconnection networks. *Computers, IEEE Transactions on*, 39(6):775–785.

de Lima, D. S. A., Guimarães, R. S., Vassoler, G. L., Liberato, A. B., Martinello, M., and Villaca, R. S. (2015). Avaliação do uso do openflow na recuperação de falhas em data centers centrados nos servidores. In *Anais do XIV Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, WPerformance '15, Recife/PE, Brasil. SBC.

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. *SIGCOMM Comput. Commun. Rev.*, 39(4):63–74.

Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., and Lu, S. (2008). Dcell: A scalable and fault-tolerant network structure for data centers. *SIGCOMM Comput. Commun. Rev.*, 38(4):75–86.

Luiz Fernando T. de Farias, Morganna C. Diniz, S. C. d. L. (2014). Uma abordagem para redução da tabela de encaminhamento sob a ótica da interface de saída dos pacotes. In *XIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2014)*, Porto Alegre/RS. SBC.

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.*, 38(2):69–74.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., and Casado, M. (2015). The design and implementation of open vswitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA. USENIX Association.

Saad, Y. and Schultz, M. H. (1989). Data communication in hypercubes. *Journal of Parallel and Distributed Computing*, 6(1):115 – 135.

Vassoler, G. L. (2015). *TRIIIAD: Uma Arquitetura para Orquestração Automônica de Redes de Data Center Centrado em Servidor.* PhD thesis, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo (PPGEE/UFES), Vitória/ES, Brasil.

Vencioneck, R., Vassoler, G., Martinello, M., Ribeiro, M., and Marcondes, C. (2014). Flexforward: Enabling an sdn manageable forwarding engine in open vswitch. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 296–299.