Avaliação do Uso do OpenFlow na Recuperação de Falhas em *Data Centers* Centrados nos Servidores

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

[1] Núcleo de Estudos em Redes Definidas por Software (NERDS)
Universidade Federal do Espírito Santo (UFES) – Vitória, ES

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

Abstract. *In a server-centric Data Center, servers not only participate in the data processing task but also participate in the traffic forwarding in the network. In general these servers are organized in a topology known as hypercube, in which traffic forwarding is based in a simple exclusive-or operation. On the one hand, the simplicity of this operation contributes to improve throughput and reduce latency in the Data Center, but on the other hand, the traffic forwarding only works in cases where the hypercube is complete, ie, in the absence of nodes or links failure. This paper presents an OpenFlow based alternative to treat failures in hypercubes. In the proposed solution, when a failure occurs an SDN controller is notified and it modifies the forwarding method of the nodes affected by this failure, calculates alternative paths in the hypercube and installs them in the nodes, ensuring delivery of all messages.*

Resumo. *Em Data Centers centrados em servidores estes não somente participam no processamento dos dados, mas também no encaminhamento do tráfego de rede. Em geral os servidores são organizados em uma topologia conhecida como hipercubo, onde o encaminhamento é feito através de simples operações de ou-exclusivo. Se por um lado essa simplicidade contribui para o aumento da vazão e diminuição da latência, por outro o encaminhamento só funciona caso o hipercubo esteja completo, ou seja, na inexistência de falhas de nó ou enlace. Este artigo apresenta uma alternativa, baseada na tecnologia OpenFlow, para o tratamento de falhas em hipercubos. Na solução proposta, em caso de ocorrência de uma falha um controlador SDN é notificado e modifica a forma de encaminhamento dos nós afetados por essa falha, calcula rotas alternativas e instala novas regras de encaminhamento, garantindo a entrega das mensagens.*

1. Introdução

Data Centers são bastante utilizados no processamento e armazenamento de grandes volumes de dados e, em função disso, existem diversas arquiteturas para a montagem da sua infraestrutura, são exemplos: Dcell [Guo et al. 2008], BCube [Guo et al. 2009], Al-Fares et al. [Al-Fares et al. 2008]. Dentre essas arquiteturas destacam-se os *Data Centers* centrados nos servidores (*server-centric*). A característica mais importante dos *Data Centers*

*Agradecimentos à CAPES, CNPq e FAPES pelo financiamento parcial deste trabalho.

centrados nos servidores 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*. A Seção 2 discute alguns trabalhos relacionados.

Uma topologia de interligação dos servidores bastante comum nas arquiteturas de *Data Center* centrados nos servidores é o hipercubo [Saad and Schultz 1989]. 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, vide Seção 3. 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 [Vencioneck et al. 2014, Dally 1990], 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. Devido a isso, torna-se necessário a proposição de soluções capazes de lidar com esse problema, e este artigo apresenta 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. Maiores detalhes sobre o encaminhamento de tráfego baseado em operações de XOR em hipercubos virtuais e sobre o desenvolvimento da solução proposta serão apresentados na Seção 4.

Em [Vencioneck et al. 2014] foi apresentada uma solução, denominada FlexForward, capaz de alterar dinamicamente a forma de encaminhamento dos nós de um *Data Center* em função de mensagens de controle do protocolo OpenFlow. Baseado na ideia do FlexForward, este artigo propõe que o controlador SDN ao detectar que o hipercubo tornou-se incompleto, envie mensagens OpenFlow para alternar do modo de encaminhamento XOR para um modelo de encaminhamento baseado em tabelas de fluxo. Mais detalhadamente, na ocorrência de uma falha de enlace (ou na indisponibilidade de um nó) o controlador reconstrói o grafo de conectividade, calcula as rotas entre todas as possíveis origens e destinos para os nós afetados por essa falha, envia mensagens de modificação das tabelas de encaminhamento (Flow-Mod) do protocolo OpenFlow e alterna a forma de encaminhamento dos nós, garantindo-se assim a conectividade mesmo durante o período de restauração da falha.

Além da proposição de uma solução simples, baseada no protocolo OpenFlow,

para o problema da conectividade em hipercubos incompletos, outra contribuição deste artigo é a avaliação de desempenho da solução proposta, que pretende mostrar que, embora não seja tão eficiente quanto o encaminhamento baseado em XOR, mostra-se uma solução viável para garantir a conectividade em hipercubos incompletos. Na Seção 5 serão apresentados os resultados de algumas avaliações feitas com um protótipo desenvolvido no Mininet[1], com suporte ao OpenFlow 1.3, controlador Ryu[2] e um hipercubo com 8 nós. Foram avaliados o tempo médio de restauração das falhas, a taxa média de perda de pacotes em função da vazão e uma análise qualitativa que mostra a manutenção da conectividade mesmo em cenários de falha de enlace.

2. Trabalhos Relacionados

BCube [Guo et al. 2009] é uma proposta de *Data Center* modular baseada em uma arquitetura de redes centrada em servidores cuja topologia é bastante similar ao hipercubo, porém com um menor número de enlaces entre os servidores. Uma vez que a topologia não é um hipercubo original, a ocorrência de falhas não ocasiona interrupção de conexão entre os servidores com a contrapartida de aumentar o tamanho médio dos caminhos mesmo quando não existem falhas. Dcell [Guo et al. 2008] e [Al-Fares et al. 2008] são outros exemplos de arquiteturas centradas nos servidores, com flexibilidade de roteamento, porém não se baseiam em hipercubos virtuais.

A abordagem tradicional para contornar o problema da interrupção de continuidade da conexão em casos de falha de nó ou enlace nos hipercubos tem sido flexibilizar a topologia hipercubo, deformando-a. Faz-se uso de hipercubos virtuais, *torus*, grafos gêmeos [Vassoler et al. 2014] ou meshes entre os servidores. Além do BCube, já citado, são outros exemplos: [Pasquini et al. 2011, Fujiwara et al. 2014, Vassoler et al. 2014]. Porém, o compromisso imposto por todas essas soluções, inclusive a nossa, é a necessidade de incorporação de uma tabela de encaminhamento em cada nó do *Data Center*.

O FlexForward [Vencioneck et al. 2014] apresenta uma solução para gerenciar dinamicamente os mecanismos de encaminhamento de mensagens, no plano de dados, em redes com suporte SDN baseadas no protocolo OpenFlow. A implementação é feita considerando-se *switches* virtuais baseados em software, especificamente foi utilizado o Open vSwitch, e através de mensagens de controle do protocolo OpenFlow torna-se possível alterar dinamicamente o modo de encaminhamento no plano de dados, entre os modos previamente implementados.

A proposta deste trabalho se diferencia do FlexForward nos seguintes pontos: (i) nossa proposta é capaz de lidar com encaminhamento de pacotes em hipercubo incompleto, enquanto que no FlexForward uma falha pode gerar inconsistência na entrega dos pacotes, pois eles podem ser encaminhados para nós ou enlaces inoperantes, conforme será detalhado na Seção 3. (ii) ao detectar uma falha, as rotas para encaminhamento baseado em tabelas de fluxo são pré-instaladas, de forma que os pacotes permanecem no plano de dados. No FlexForward, ao optar por encaminhando baseado em tabelas de fluxo, para cada novo fluxo, é pós-calculada a nova rota. Isso pode sobrecarregar o controlador SDN.

A principal vantagem da nossa solução, proposta neste artigo, é que o hipercubo mantém-se completo em situações normais, garantindo-se sempre a menor rota,

[1]http://mininet.org/

[2]http://osrg.github.io/ryu/

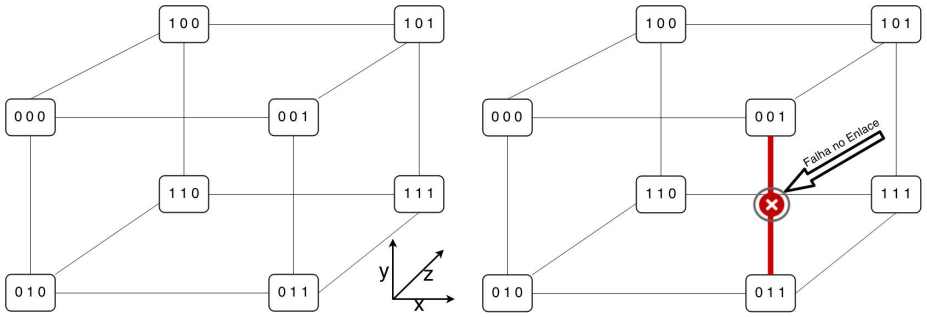
deformando-se somente em casos de falha, onde a garantia do menor caminho fica condicionada ao novo grafo reconstruido após a falha em detrimento da manutenção da completa conectividade entre os nós. Ou seja, nossa solução garante rotas mínimas, menor latência e maior vazão em situações normais por basear-se em hipercubos completos. Essas características sofrem algumas alterações em situações de falha nos nós, porém garantindo conectividade total entre os servidores do *Data Center*.

3. Roteamento XOR em Hipercubos

O roteamento baseado na operação XOR para redes em hipercubo, utiliza identificadores de comprimento de n bits, aonde n conecta até 2^n nós. Cada nó é identificado de forma única por uma representação de n bits. Esses identificadores podem ser utilizados no encaminhamento de pacotes, através da realização da operação lógica conhecida como ou-exclusivo (XOR) entre dois identificadores de nós, origem e destino.

Um nó se conecta a outros n nós, os quais são os seus vizinhos, se e somente se a distância de Hamming entre eles for igual a 1. Portanto, para que um nó possa alcançar outros nós com distância de Hamming maior que 1, ele necessita utilizar nós intermediários para encaminhar os pacotes.

De forma resumida, essa técnica de encaminhamento é baseado em um algoritmo de roteamento na origem, que tal como um *fabric*, não demanda nenhum sistema de controle ou estado (*stateless*). Para encaminhar os pacotes, um nó utiliza uma simples operação XOR sobre seus vizinhos para encontrar o vizinho mais próximo do destino.



(a) Topologia Hipercubo (b) Topologia com falha de 1 enlace

Figura 1. Hipercubo binário com três dimensões

Considere o cenário da Figura 1(a). Imagine que o nó 010 deseja enviar uma mensagem para o nó de destino 001. O nó de origem conhece o identificador do nó de destino, então efetua a operação XOR. A decisão de roteamento d entre os nós 010 e 001 é representada pela Equação 1, resultando em 011; aonde o primeiro bit 1 (menos significativo) representa o eixo *x*, o segundo bit 1 representa o eixo y e o terceiro bit 0 (mais significativo) representa o eixo *z*.

$$d = (no_{atual} \oplus no_{destino}) \tag{1}$$

Temos então dois caminhos de menor custo disponíveis, cujo o bit é igual a 1: um no eixo x e outro no eixo y. Isso quer dizer que encaminhar a mensagem nas direções x ou y resultará em caminhos de igual custo. Nosso mecanismo sorteia entre essas duas

possibilidades de forma igualitária, selecionando por exemplo o caminho do eixo *x*. Então a mensagem é transmitida para o nó 011. Por sua vez, o nó 011 recebe e efetua novamente a operação da Equação 1. No entanto o nó atual agora é 011, $d = (011 \oplus 001)$. Como resultado temos 010. Como o único bit igual a 1 é o do eixo *y* a mensagem é encaminhada por esse eixo chegando ao seu destino com o menor custo possível.

Agora vamos imaginar o cenário da Figura 1(b), aonde o enlace entre os nós 011 e 001 está inoperante. Se repetirmos o exemplo anterior e mantivermos a escolha aleatória do primeiro encaminhamento no eixo *x*, o roteamento XOR sempre irá tentar entregar a mensagem pelo enlace com falha mais adiante, resultando na perda da mensagem.

Conforme ilustrado na Figura 1(a), pode-se concluir que a eficiência do roteamento XOR está diretamente condicionada a estrutura topológica do hipercubo [Vencioneck et al. 2014, Dally 1990], ou seja, o endereço de cada nó têm relação direta com seus vizinhos, que deve ter a diferença exata de 1 (um) bit.

Em uma situação de falha (Figura 1(b)), seja em enlace ou nó, a condição de cubo completo deixa de ser atendida. Nossa proposta consiste em utilizar o protocolo Openflow[McKeown et al. 2008] para que os nodos afetados pelas falhas sejam reconfigurados para rotear com base em regras de fluxo OpenFlow, substituindo o roteamento XOR *stateless* por um esquema de roteamento *statefull* nesses nodos. Essas regras permitirão ao nó encaminhar os pacotes em situações de falhas, ou seja, em condição em que o cubo como infra estrutura física e lógica torna-se um cubo incompleto. A implementação da proposta será descrita a seguir.

4. Implementação

A implementação do protótipo descrito neste artigo foi dividida em duas partes principais: plano de controle e plano de dados. No plano de controle, descreveremos as características da aplicação do controlador, bem como questões relacionadas à versões dos software que serão necessários para o funcionamento da aplicação. Dentre as ações realizadas pela aplicação, destacam-se:

1. Detecção de falhas nos enlaces ou nós e sua posterior restauração;
2. Execução do cálculo das rotas utilizando o método de menor caminho;
3. Instalação das regras OpenFlow no plano de dados de cada nó.

No plano de dados iremos tratar de questões relacionadas com a instalação e atualização das regras de fluxo, suas características, assim como aspectos relacionados à versão do protocolo OpenFlow utilizado nas regras de fluxo instaladas em cada nó.

4.1. Plano de Controle

A aplicação proposta para o plano de controle será responsável pela detecção das falhas (e sua posterior restauração) dos enlaces ou nós em um hipercubo, pelo cálculo das rotas alternativas em uma topologia hipercubo com 1 (uma) ou mais falhas de nó ou enlace, bem como a instalação das regras de encaminhamento alternativas em cada nó. Para isso, foi desenvolvida uma aplicação utilizando a plataforma Ryu 3.18[3] como controlador SDN utilizando a linguagem Python. O Ryu foi escolhido por ser uma plataforma aberta que suporta as versões mais recentes do protocolo OpenFlow.

[3]http://osrg.github.io/ryu/

No desenvolvimento desta aplicação no plano de controle realizamos uma divisão das atividades em módulos, onde cada um é responsável por uma determinada etapa na construção das regras de fluxo, partindo desde a manipulação das mensagens de falha até a geração das regras de modificação que serão instaladas em cada um dos nós no hipercubo. Portanto, os módulos ficaram divididos da seguinte forma: 1) *Recepção das Mensagens*; 2) *Tratamento dos Eventos*; 3) *Representação Topológica e Cálculo das Rotas* e; 4) *Instalação das regras de encaminhamento*, identificado na Figura 2.

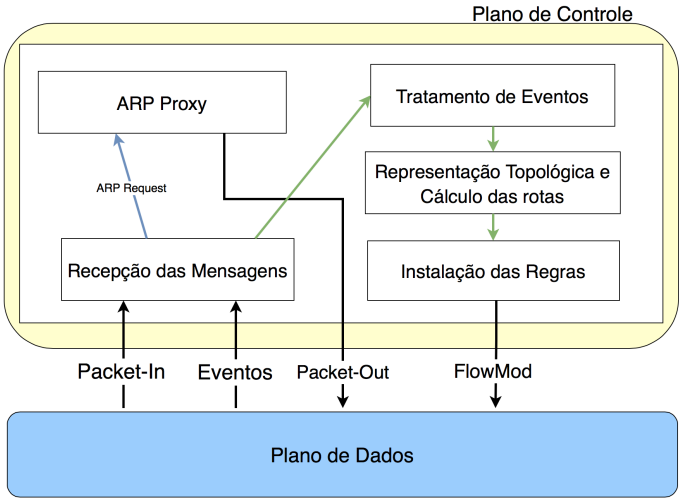


Figura 2. Estrutura da Aplicação.

Algoritmo 1: Recepção das mensagens

Dados: $Pacote$
Resultado: $Pacote$
início
  se $Pacote$ *é um ARP Request* **então**
    envia $Pacote$ para o módulo *ARP Proxy*;
  senão
    envia $Pacote$ para o módulo *Tratamento de Eventos*;
  fim se
fim

No módulo *Recepção das Mensagens* o controlador deverá, primeiramente, identificar o tipo de mensagem que foi encaminhado para o controlador, conforme mostrado no Algoritmo 1. Somente mensagens referentes às falhas e restauração dos enlaces ou nós do hipercubo serão tratadas por este módulo. Esta abordagem foi necessária para evitar problemas relacionados a inundações da rede causadas pelo protocolo ARP, que utiliza envio de mensagens em *broadcast* e necessitam ser tratadas separadamente. Para o caso de recebimento de mensagens ou eventos relacionados ao plano de dados (falha de enlace ou nó), a aplicação encaminhará esta mensagem para tratamento no módulo *Tratamento de Eventos*.

Assim que a aplicação interceptar os pacotes de *ARP* e encaminhar para a aplicação de *ARP Proxy*, o módulo em questão interpretará as mensagens de *ARP Request* para em seguida enviar uma mensagem de *ARP Reply* utilizando o mecanismo de *Packet-Out* presente no Openflow[ONF 2012]. Este mecanismo permite o envio de pacotes utilizando uma porta específica do nó controlado, pois as mensagens de *Packet-Out* contém o pacote completo e a suas respectivas ações, que pode ser o encaminhamento por uma determinada porta. Na montagem da mensagem de *ARP Reply*, a aplicação realiza um mapeamento prévio do endereço de rede (IP) para o endereço de enlace (MAC) definido através de um dicionário de dados na aplicação.

O módulo de *Tratamento de Eventos* basicamente irá identificar o(s) nó(s) ou enlace(s) afetados a partir das mensagens de falha, através de um evento Openflow informando falha em algum determinado enlace ou nó, recebido no plano de controle. Uma vez identificados o(s) nó(s) ou enlace(s) defeituoso(s), essa identificação será encaminhada para o módulo *Representação Topológica e Cálculo de Rotas*, que possui 2 (duas) funções específicas agrupadas em um mesmo módulo, utilizando a lógica descrita no Algoritmo 2.

Algoritmo 2: Tratamento de Eventos

Dados: $Mensagem$
Resultado: $Mensagem$
início
  se $Mensagem$ *é um Evento* **então**
    envia $Mensagem$ para a função de *Representação Topológica()*;
  senão
    envia $Mensagem$ para a função de *Cálculo das Rotas()*;
  fim se
fim

Através da função de *Representação Topológica* tem-se uma visão do grafo associado ao hipercubo. Ao ser identificado o(s) nó(s) ou enlace(s) defeituoso(s) a função de representação topológica modificará a representação do grafo do hipercubo, removendo ou adicionando nós ou arestas, conforme representado no Algoritmo 3. Para a representação e manipulação do grafo utilizamos a biblioteca *NetworkX*[4], pois, através dela, é possível criar diversos tipos de estruturas de redes, dinâmicas e complexas de forma simples e eficiente.

Depois da modificação do grafo causada por algum evento de falha em algum enlace ou nó, a função de *Cálculo das Rotas* determinará as novas rotas para os nós afetados por essa falha. No cálculo das rotas, utilizamos o algoritmo (*Shortest Path*) de *Dijkstra* responsável pelo cálculo das rotas alternativas de menor caminho no cubo incompleto. Essas novas rotas serão encaminhadas para o módulo *Instalação das Regras* que será responsável pela geração das regras de *Flow-Mod*[ONF 2012] a serem enviadas para o plano de dados. As regras de *Flow-Mod* são responsáveis por modificação das

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

tabelas de encaminhamento dos nós compatíveis com o protocolo OpenFlow 1.3, utilizado neste protótipo.

Algoritmo 3: Representação Topológica e Cálculo das Rotas

Dados: $Mensagem$, No, $Grafo$
Resultado: $Rotas$, $Dpid$
início
  se $Mensagem$ *é Falha no Enlace ou Falha de algum Nó* **então**
    remove enlace ou nó do $Grafo$;
    enquanto *Não chegar no fim da lista de Nós* **faça**
      calculodasrotas($Mensagem$, $Grafo$);
      enviando rotas para o módulo *Instalação das Regras*;
    fim enqto
  $Mensagem$ é Retorno do Enlace ou de algum Nó adiciona enlace ou nó ao $Grafo$;
  enquanto *Não chegar no fim da lista de Nós* **faça**
    calculodasrotas($Mensagem$, $Grafo$);
    enviando rotas para o módulo *Instalação das Regras*;
  fim enqto
fim

Em resumo, no caso de falha em algum nó ou enlace, assim que ocorrer a detecção do evento correspondente a aplicação deverá identificar e atualizar a mudança topológica ocorrida no grafo. Em seguida, as rotas deverão ser recalculadas, baseando-se no algoritmo de *Dijkstra*. Através do módulo de *Instalação das Regras* são geradas mensagens de *Flow-Mod* para todos os nós afetados pela falha.

4.2. Plano de Dados

No cenário das redes de *Data Center* centradas em servidores a topologia hipercubo enquadra-se como sendo uma das mais comuns. A Figura 3 exemplifica uma topologia mínina, de grau 3 (três), a qual interliga oito nós e cada um deles está interligado a um nó controlador no Plano de Controle. Essa figura foi utilizada como base para a avaliação do protótipo apresentado nesse artigo e será melhor explicada a seguir.

Conforme já visto na Seção 3, nesse tipo de topologia quando o hipercubo está completo o encaminhamento dos pacotes pode ocorrer de forma rápida e eficiente através de uma simples operação de ou exclusivo. Como pode ser observado na figura, cada nó apresenta um endereço binário que representa seu identificador no hipercubo.

Na identificação de cada nó, modificamos o seu endereço de enlace (*MAC Address*) e definimos uma parte do endereço para sua localização e outra para a sua identificação no hipercubo, conforme visualizado na Figura 4. Em uma rede com Openflow, utilizamos uma identificação exclusiva para cada plano de dados (ou nó), denominada de *DPID (DataPath IDentification)* [ONF 2012], que inicia sua identificação a partir do valor 1 (um). No caso da topologia hipercubo com grau três, a identificação de cada nó no controlador estará no intervalo de 1 a 8. Como a identificação em uma rede com

encaminhamento XOR é iniciada pelo valor 0 (zero) e, no caso de uma rede hipercubo com grau 3, o valor final será 7, a aplicação do controlador deverá mapear estes endereços do hipercubo utilizados anteriormente pelo encaminhamento XOR com os endereços correspondentes ao DPID de cada nó.

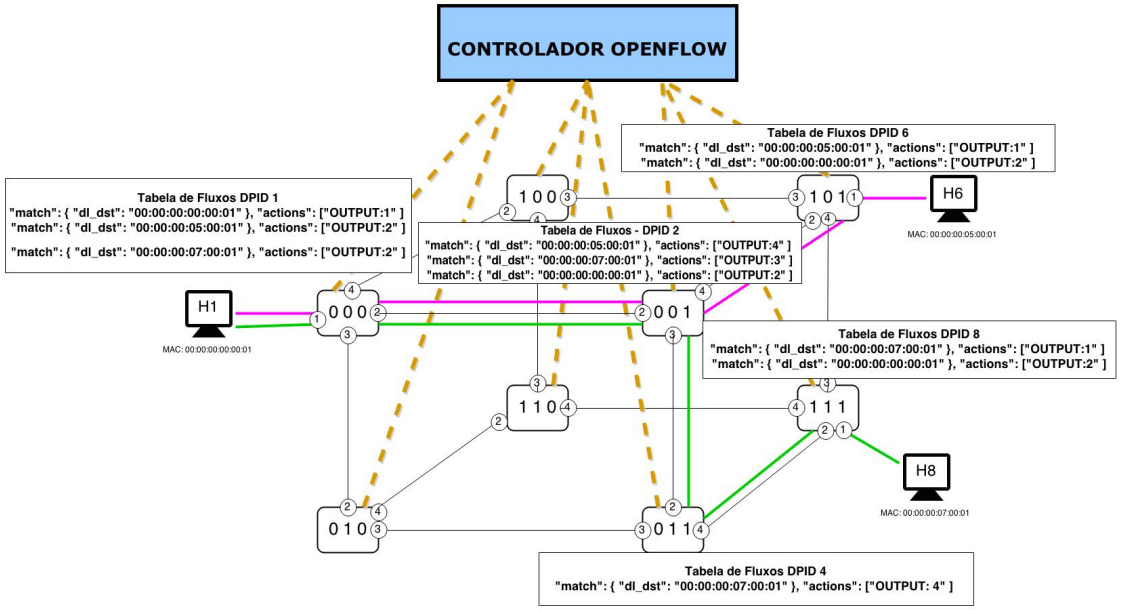


Figura 3. Topologia hipercubo com as rotas de encaminhamento

Conforme podemos observar na Figura 3, para realizar a comunicação do nó $H1$ com o nó $H8$, o controlador deverá instalar regras de fluxo baseando-se nos *endereços de destino* do hipercubo e, como ação, o fluxo deverá ser encaminhado para uma determinada porta de saída (*OF_OUTPUT*). Vejamos, como exemplo, a regra instalada no nó 0 (DPID 1): *match: dl_dst: 00:00:00:07:00:01, actions: OUTPUT 2*.

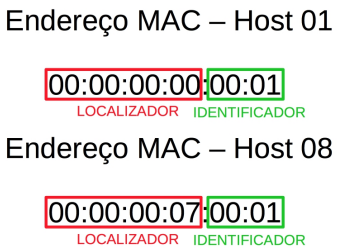


Figura 4. Localizador e Identificador de cada nó do hipercubo

Neste exemplo o nó 0 irá encaminhar os pacotes para a interface 2 quando o *MAC Address* de destino for 00:00:00:07:00:01, que está atrelado ao nó 7 (DPID 8), conforme ilustrado na Figura 4. Todas estas regras foram instaladas baseando-se no cálculo do menor caminho utilizado pela aplicação descrita no plano de controle. O número 1 presente no final do *MAC Address* deste exemplo será usado em trabalhos futuros, onde pretendemos alocar mais de um servidor virtual a um mesmo endereço de nó do hipercubo, neste caso ele seria um identificador de servidor. Por enquanto leia-se "servidor 1" na "localização 7" do hipercubo.

5. Resultados e Discussões

Esta seção apresenta o ambiente de experimentação usado na avaliação do protótipo e os resultados obtidos nos experimentos. Para melhor organização, optamos por dividi-la em 3 (três) subseções. A primeira (Seção 5.1) descreve o ambiente de experimentação. Em seguida, na Seção 5.2 avaliamos o tempo de instalação das novas rotas, medido desde o instante em que uma falha de enlace é notificada até o instante em que a comunicação é restaurada. Ainda nesta seção avaliamos a quantidade de pacotes pacotes perdidos durante esse intervalo em função da vazão em que o enlace está submetido. Por último, na Seção 5.3 apresentamos um resultado qualitativo que mostra a manutenção da conectividade mesmo durante a ocorrência de uma falha.

5.1. Ambiente de Experimentação

Como ambiente de experimentação utilizamos um hipercubo de 8 nós com um servidor (*host*) atrelado a cada nó e seus endereços MAC devidamente associados à sua posição no hipercubo, conforme mostra a Figura 3. Optamos em utilizar o OpenFlow na versão 1.3 para instalação das regras de fluxo, por ter suporte a diversas *features* que serão exploradas em trabalhos futuros, como por exemplo o suporte a múltiplos controladores. Como máquina de encaminhamento (*software switch*) utilizamos o Open vSwitch [5] na versão 2.3.

Utilizamos como ambiente de emulação o Mininet 2.2.0[6] por ter suporte ao OpenFlow 1.3. Em conjunto, fizemos o uso também da ferramenta Wireshark[7] para a aferição das medições de tempo. Para geração e análise do tráfego durante os experimentos foi utilizada a ferramenta Iperf3[8] na versão 3.0.4 com o protocolo UDP.

Para garantir a confiabilidade da coleta dos dados e minimizar o efeito de interferências externas ao experimento, a cada mudança nos parâmetros o ambiente de testes foi reinicializado.

5.2. Tempo para a Restauração de Conectividade e Avaliação da Perda de Pacotes

O objetivo deste experimento é avaliarmos o tempo de restauração da conectividade no hipercubo quando ocorre uma ou várias falhas simultâneas. A medição deste tempo é feita usando a ferramenta Wireshark instalada nos nós afetados pelas falhas. Mede-se o tempo decorrido desde a sinalização da falha ao controlador (envio de mensagens de *PORT DOWN* e *LINK DOWN* do protocolo OpenFlow 1.3) e a chegada da última mensagem de *Flow-Mod*, que corresponde ao fim da reconfiguração das tabelas de fluxo.

Da mesma forma, ainda nesta subseção avaliamos a variação de perdas de pacotes durante o tempo de restauração da conectividade em diferentes taxas de tráfego. Essa avaliação foi feita utilizando-se a ferramenta Iperf3 instalada entre origem e destino de um tráfego afetado pelas falhas e mediu a variação entre a quantidade de dados enviados e recebidos. A diferença representa a perda de conexão e, consequentemente, a perda de pacotes. Todos os dados coletados foram armazenados em arquivos texto no formato .csv para posterior análise.

[5]http://openvswitch.org/
[6]http://mininet.org/
[7]https://www.wireshark.org/
[8]https://iperf.fr/

Os resultados da Figura 5 mostram o tempo de restauração da conectividade no hipercubo, representado pelo tempo de instalação da nova tabela de fluxos do protocolo OpenFlow (eixo y), em função da quantidade de falhas simultâneas (eixo x). Nota-se que em nossos testes este tempo varia entre aproximadamente 0,1s à 0.4s. Para chegarmos a esses valores, com o hipercubo completo é provocada uma falha e contabiliza-se o tempo decorrido entre a detecção desta falha pela aplicação no plano de controle e a reconfiguração das tabelas de fluxo e restauração da conectividade. Nos cenários com mais de uma falha, como por exemplo, queda de dois enlaces, essas falhas ocorrem simultaneamente.

Ainda na Figura 5, utilizamos um nível de confiabilidade de 95% e o Intervalo de Confiança (IC) está representado pelas barras verticais no topo de cada resultado. O tempo de recuperação da falha de um único enlace apresenta uma média 0,090 segundos e 0,001 como IC, que não torna-se visível no gráfico. Os resultados nos mostram que o tempo não aumenta de forma diretamente proporcional em relação a quantidade de falhas, como mostra o gráfico referente a Figura 5.

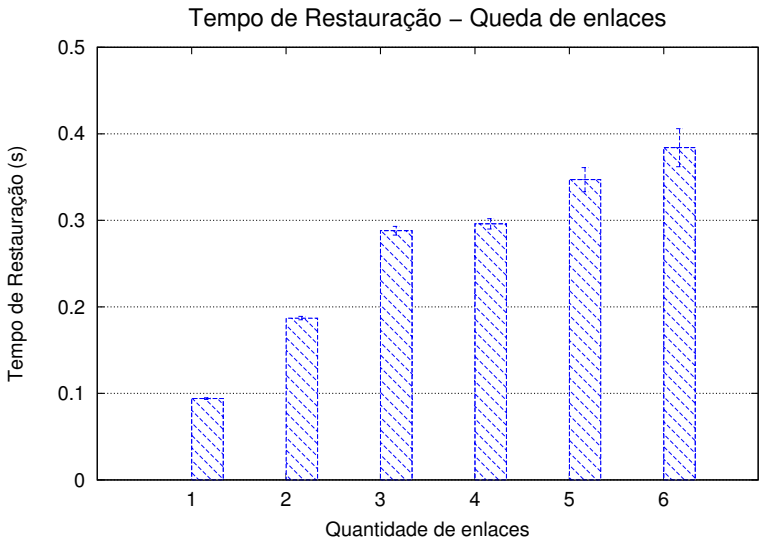


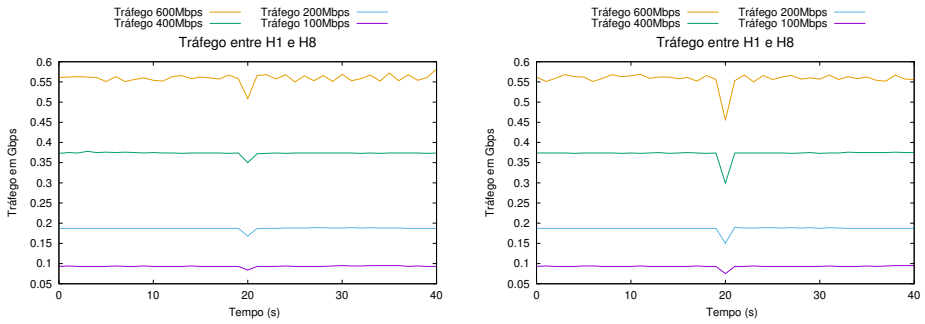
Figura 5. Tempo de restauração da conectividade na queda dos enlaces.

O aumento no tempo de restauração de falhas na queda de vários enlaces se dá em decorrência das ações que o controlador implementa ao detectar uma falha, mesmo que elas ocorram simultaneamente, pois para cada detecção de falha um novo grafo topológico precisa ser gerado, novas rotas são calculadas e em seguida são enviadas as mensagens *Flow-Mod* para as tabelas de encaminhamento de cada nó afetado pela falha. Todas essas ações ocorrem para cada falha detectada. Vale destacar aqui que a aplicação do plano de controle não foi implementada usando-se conceitos de multitarefa e concorrência, o que implica que as falhas simultâneas serão tratadas individualmente pelo controlador e essa característica tem um impacto direto nos resultados obtidos e serão alvo de melhoramentos em trabalhos futuros.

No próximo experimento apresentaremos um comparativo da perda de pacotes em função de diferentes taxas de tráfego em um determinado enlace no momento da ocorrência da falha. Como configuração inicial todos os nós estavam interligados por

seus respectivos enlaces, configurando-se assim um hipercubo completo. Com o uso do Iperf3 foi gerado um tráfego (por exemplo) entre os *hosts* $H1$ e $H8$ (Figura 1(b)) com um intervalo de tempo de 40s.

Durante esse tempo no instante de aproximadamente t=20s são provocadas falhas no caminho entre $H1$ e $H8$. As Figuras 6(a) e 6(b) mostram situações onde ocorrem a queda de 1 (um) e 2 (dois) enlaces no hipercubo apresentado na Figura 1(b). Os resultados foram obtidos comparando as diferentes vazões, com o objetivo de verificar o comportamento da perda de pacotes em função da variação da vazão, uma vez que o tempo de restauração da tabela de fluxos não varia com o fluxo no plano de dados.



(a) Queda de um enlace (b) Queda de dois enlaces

Figura 6. Perda de pacotes em função da vazão e da quantidade de falhas.

Na Figura 6(a) mostramos os resultados para a ocorrência da falha de um enlace, enquanto que na Figura 6(b) apresentamos os resultados para a ocorrência de falhas em dois enlaces. As falhas foram provocadas próximas ao instante t=20s. Podemos perceber que as perdas variam em função da taxa de tráfego entre a origem e o destino, quanto maior a vazão no momento da falha, maiores serão as perdas. Na Figura 6(b), onde ocorre a queda de dois enlaces, o cenário de resultados é semelhante a 6(a), portanto percebemos que há uma incidência um pouco maior de perdas. Isso se dá em decorrência da quantidade de enlaces que foram afetados por falhas desde a origem até o destino. Nesse experimento, as falhas ocorreram de forma simultânea em dois enlaces distintos em nós distintos. A quantidade de falhas impacta no tempo de restauração do hipercubo, evidenciado na Figura 5.

5.3. Manutenção da Conectividade no Hipercubo

O objetivo desse experimento é demonstrar que a solução proposta faz o chaveamento entre as interfaces de encaminhamento, garantindo a manutenção da comunicação entre nós no hipercubo por diferentes rotas.

Na Figura 7 mostramos uma visão qualitativa diretamente associada a um nó envolvido na comunicação entre 2 (dois) *hosts*. Para ilustrarmos esse cenário geramos um fluxo entre os *hosts* $H1$ e $H8$, sendo que, o *host* $H1$ está conectado à localização 0 do hipercubo (*DPID 1*) e o *host* $H8$ está conectado à localização 7 (*DPID 8*), conforme ilustrado na Figura 3. Como o fluxo é destinado ao host $H8$, por padrão, antes da falha, o fluxo é encaminhado pela interface de saída número 2, escolha tomada com base nas opções de menor caminho para o destino desejado. Em seguida provocaremos a falha neste enlace, entre os nós (localizações) 000 e 001 do hipercubo.

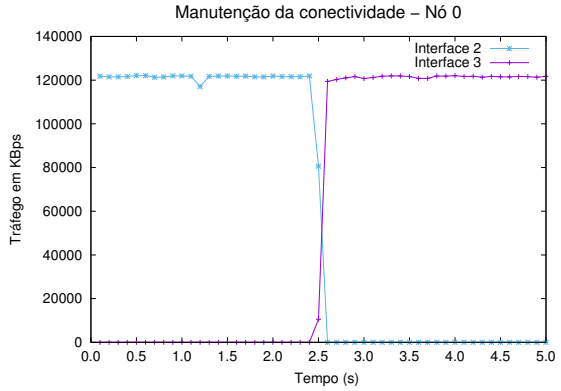


Figura 7. Manutenção da conectividade entre os *hosts* $H1$ e $H8$.

No instante em que essa falha ocorre a aplicação no plano de controle recebe o evento relacionado e procede ao cálculo das novas rotas entre os nós afetados. Desta maneira, a aplicação modificará o fluxo para o host $H8$ a partir do nó 0 que passará a encaminhar pela interface 3, rota alternativa calculada pela aplicação do plano de controle. Portanto, a Figura 7 mostra exatamente o período de transição entre a descoberta da falha até a reação da aplicação que instalará as novas regras de fluxo no nó em questão garantindo a conectividade mesmo enquanto a falha não é restaurada. Embora não tenhamos apresentado neste artigo por limitações de espaço, o tempo de retorno à rota original após a restauração da falha não influencia na perda de pacotes, ou seja, não haverá nenhuma perda neste caso.

6. Conclusões e Trabalhos Futuros

O artigo apresentou os resultados de uma avaliação de desempenho de uma solução baseada no protocolo OpenFlow para garantir conectividade em hipercubos incompletos, uma topologia tradicional em *Data Centers* centrados nos servidores. Para validação da proposta foi implementado um protótipo experimental com uso do Mininet, controlador Ryu e OpenFlow 1.3. Os resultados de desempenho apresentados neste artigo não têm a intenção de serem conclusivos e definitivos, mas mostram que, mesmo em ambientes emulados, há alguma perda de conectividade até que se detecte a falha e seja feita a instalação de rotas alternativas nos nós do hipercubo. Após isso, a conectividade é restaurada e pode-se afirmar que o tempo de instalação das rotas alternativas é pequeno, em geral menor do que o tempo de restauração da falha de um nó ou enlace, que pode ser da ordem de minutos ou horas.

Os próximos passos no desenvolvimento deste trabalho serão: (i) aperfeiçoar a aplicação residente no controlador SDN para minimizar o número de vezes que as rotas precisam ser recalculadas. Para isso, estamos trabalhando em um modelo de agregação de endereços de forma otimizar a quantidade necessária de informações para manter o hipercubo funcionando em casos de falha e recuperação dos controladores. (ii) a integração da solução proposta com o FlexForward, fornecendo informações para o método de encaminhanto em XOR, implementado no *datapath* do Open vSwitch. Dessa forma, mesmo em caso de falhas, o encaminhamento seria realizado sem o uso de tabelas de fluxo. Após isso, iremos refazer a avaliação de desempenho em hipercubos maiores, possivelmente em ambientes reais, não emulados.

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.

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

Fujiwara, I., Koibuchi, M., Matsutani, H., and Casanova, H. (2014). Skywalk: A topology for hpc networks with low-delay switches. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IPDPS '14, pages 263–272, Washington, DC, USA. IEEE Computer Society.

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.

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.

ONF, O. N. F. (2012). Openflow switch specification version 1.3.0 (wire protocol 0x04). Website. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf`.

Pasquini, R., Verdi, F. L., and Magalhães, M. (2011). Integrating servers and networking using an xor-based flat routing mechanism in 3-cube server-centric data centers. In *Anais do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2011)*, Campo Grande, MS. Sociedade Brasileira de Computação (SBC).

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

Vassoler, G., Paiva, M., Ribeiro, M., and Segatto, M. (2014). Twin datacenter interconnection topology. *Micro, IEEE*, 34(5):8–17.

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.