Equilibrando Energia, Redundância e Desempenho em Redes de Centros de Dados

Antônio Cleber de S. Araújo[1], **Leobino N. Sampaio**[1], **Artur Ziviani**[2]

[1]Mestrado Multinstucional em Ciência da Computação (MMCC)
Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Salvador, BA - Brasil

[2]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis, RJ - Brasil

{antonio.cleber,leobino}@ufba.br, ziviani@lncc.br

Abstract. *The current large-scale datacenters typically adopt redundancy of servers and communication devices for increased reliability and availability. Highly redundant infrastructure, however, is one of the challenges of the area due to high energy consumption. This paper presents the Beep, an energy-efficient strategy for datacenter networks based on the Fat Tree topology. Our strategy uses multipaths and the global overview offered by software-defined networks to balance the energy efficiency, the equipment redundancy level, and the performance gain when dealing with the traffic demands. The BEEP was implemented in an OpenFlow network and the multipaths using MPTCP (Multipath TCP). The experimental results in variants of the Fat Tree topology show gains in the energy efficiency with the strategy in the order of 30% to 45%, additionally to a better utilization of the available bandwidth, as more alternative paths are available.*

Resumo. *Os grandes centros de dados atuais tipicamente adotam redundância de servidores e equipamentos de comunicação para aumento de sua confiabilidade e disponibilidade. Infraestrutura altamente redundante, contudo, consiste num dos desafios da área devido ao alto consumo de energia. Este artigo apresenta a BEEP, uma estratégia energeticamente eficiente para redes de centro de dados baseadas na topologia Fat Tree. Nossa estratégia faz uso de múltiplos caminhos e da visão global oferecida por redes definidas por software para equilibrar eficiência energética, nível de redundância dos equipamentos e ganho de desempenho no atendimento às demandas de tráfego. A BEEP foi implementada através de uma rede OpenFlow e os múltiplos caminhos por MPTCP (Multipath TCP). Resultados experimentais em variantes da topologia Fat Tree demonstraram ganhos de eficiência energética com a estratégia na ordem de 30% a 45%, além de melhoria na utilização da largura de banda disponível conforme haja mais caminhos alternativos disponíveis.*

1. Introdução

Em anos recentes, os elevados requisitos de confiabilidade e disponibilidade dos serviços providos pelos atuais centros de dados de larga escala têm motivado a comunidade de pesquisa a propor novas soluções no intuito de melhorar seus desempenhos. Em geral, as

propostas envolvem a adoção de topologias de redes projetadas para garantir um mínimo de redundância na infraestrutura de servidores e equipamentos [Greenberg et al., 2009, Guo et al., 2009].

A redundância dos equipamentos pode, contudo, acarretar aumentos significativos no consumo de energia. Este aumento é ratificado, principalmente, pelo aumento da utilização de *switches* com velocidade de 10 Gbps em centros de dados [Bilal et al., 2014]. Estudos estimam que a infraestrutura de rede dos EUA usa 24 TWh por ano [Nordman, 2007], custando US$ 24 bilhões. Isso inclui *switches*, roteadores, pontos de acesso e placas de rede em *hosts* e servidores. Por tais motivos, estão em curso esforços para reduzir o consumo de energia de todos os componentes da Internet como os padrões *EnergyStar* e uma força-tarefa do IEEE para prover eficiência energética em redes *Ethernet* [IEEE, 2014].

As soluções de economia de energia passam muitas vezes pela adoção de estratégias de gerenciamento autonômico que buscam desligar os equipamentos com base na demanda de tráfego [Hammadi e Mhamdi, 2014]. O problema, entretanto, é que, muitas vezes, tais estratégias comprometem a redundância mínima necessária, assim como, o desempenho das suas redes em prol de ganhos no consumo energético.

Motivado por estas questões, este trabalho apresenta uma estratégia de gerenciamento de energia para redes de centro de dados baseadas em topologias do tipo *Fat Tree* - BEEP (do inglês, *Balanced Energy, rEdundancy, and Performance*). A estratégia busca equilibrar o consumo de energia, a redundância dos equipamentos e o desempenho da rede através de dois algoritmos que fazem uso da visão global da rede e exploram os múltiplos caminhos de comunicação entre os *hosts*. Nossa solução foi implementada através de uma rede *OpenFlow* [McKeown et al., 2008] e os múltiplos caminhos por MPTCP (*Multipath* TCP) [Barré et al., 2011]. Resultados experimentais em variantes da topologia *Fat Tree* demonstraram ganhos de eficiência energética com a estratégia BEEP na ordem de 30% a 45%, além de melhoria na utilização da largura de banda disponível conforme haja mais caminhos alternativos disponíveis.

O presente artigo está organizado da seguinte forma. Na Seção 2 são discutidos trabalhos relacionados. A Seção 3 apresenta a solução proposta. Na Seção 4 são descritos os experimentos realizados com a plataforma *OpenFlow* e MPTCP, assim como, são analisados os resultados experimentais obtidos. Por fim, a Seção 5 conclui o artigo, também discutindo trabalhos futuros.

2. Eficiência energética em centro de dados

Alguns trabalhos têm sido propostos na literatura com o objetivo de promover a redução do consumo de energia em redes de centros de dados. Entre os mais relevantes, podemos citar o *ElasticTree*, o GreenTE, o PCube e o HERO, os quais discutimos a seguir.

O *ElasticTree* [Heller et al., 2006] consiste numa proposta que busca a eficiência energética em redes de centro de dados baseadas na topologia *Fat Tree*. A solução faz uso de três módulos responsáveis por monitorar continuamente as condições de tráfego do centro de dados, selecionar o conjunto de elementos de rede que precisam estar ativos a fim de atender aos requisitos de desempenho e tolerância a falhas, e desligar os *links* e *switches* não envolvidos no roteamento. O GreenTE [Zhang et al., 2010] consiste na criação de um controlador lógico centralizado capaz de manipular caminhos através do OSPF

e criar um túnel MPLS de modo a deixar o menor número possível de roteadores ativos para satisfazer as restrições de desempenho, tais como demandas de tráfego e atrasos de pacotes. A eficiência energética é alcançada quando ele desativa os roteadores e *links* ociosos. O PCube [Huang et al., 2011] propõe melhorar a eficiência energética através do ajuste dinâmico da estrutura de redes baseadas na topologia BCube a partir das demandas de tráfego. A solução faz uso de um gerenciador que fica responsável por analisar e otimizar o tráfego da rede; escalonar os pacotes de dados; e realizar alterações na topologia de rede de forma que fiquem ativos apenas os *links* e equipamentos que estiverem em uso. O HERO (do inglês, *Hierarchical EneRgy Optimization*) visa alcançar eficiência energética em redes hierárquicas [Zhang e Ansari, 2012]. A solução desativa os *links* e equipamentos ociosos através de uma heurística que implementa um algoritmo "guloso".

Embora existam vários trabalhos relacionados ao provimento de eficiência energética em redes de centro de dados, geralmente as propostas não conseguem aliar eficiência energética com redundância e desempenho, possuindo como características: (i) os *switches* da camada de acesso não sofrem quaisquer tipo de alterações específicas; (ii) má utilização da largura de banda e do controle de congestionamento; (iii) alguns cenários contam com um alto valor de perda e atrasos de pacotes causado pelo *overhead* nos roteadores; (iv) como algumas propostas utilizam algoritmos "gulosos", as decisões tomadas são melhores somente em alguns momentos específicos da comunicação da rede.

3. Solução proposta

Este trabalho apresenta a estratégia BEEP (do inglês, *Balanced Energy, rEdundancy, and Performance*), voltada para o gerenciamento de energia para redes de centro de dados baseadas na topologia do tipo *Fat Tree*. Visando alcançar o equilíbrio entre consumo de energia, redundância mínima necessária de equipamentos e o desempenho da rede, a BEEP procura fazer com que o maior número possível de *switches* permaneçam em estado ocioso e que sejam explorados a maior quantidade possível de caminhos distintos entre os nós comunicantes. Para isso, dois algoritmos são utilizados com as seguintes funções: (i) monitorar a utilização dos *switches* da topologia a fim de deixá-los na maior parte do tempo possível em modo de espera ou com suas interfaces desligadas (com base em sua ociosidade); (ii) adequar o MPTCP à quantidade de fluxos simultâneos nas interfaces dos *hosts* comunicantes ao valor mais apropriado para a comunicação; e (iii) colocar as interfaces dos *switches* ociosos em modo de espera ou desligá-las, reativando-as conforme a necessidade.

Os algoritmos propostos são descritos nas próximas subseções. Eles foram concebidos dentro da perspectiva da *Fat Tree* a ser implementada seguindo o modelo de controle e gerenciamento centralizado, obtido por meio das Redes Definidas por Software – SDN (do inglês, *Software-Defined Network*).

3.1. Algoritmo Desativação de *Switches* (DSW)

O algoritmo de Desativação de *Switches* (DSW) — Algoritmo 1 — é executado no controlador da rede. O mesmo tem como função realizar uma monitoração constante dos *switches* da topologia, a fim de detectar a possível ociosidade dos mesmos. Uma vez que cada regra será inserida na *Flow-Table* com o `idle_timeout`[1] definido em 10 segun-

[1]O parâmetro `idle_timeout` define o tempo máximo de ociosidade que uma regra pode ter para continuar na *Flow-Table* do *Switch*.

dos, o DSW considera ocioso aquele *switch* que após este período tenha sua tabela de fluxos vazia. Esta premissa foi adotada por conta da grande e diversificada demanda de tráfego existente em centro de dados e pela baixa probabilidade de existirem tabelas de fluxos sem alterações em *switches* ativos. Logo, os equipamentos identificados pelo DSW no estado ocioso têm suas interfaces desligadas ou colocadas em modo de espera. Este procedimento é realizado a partir do conjunto de *switches* de acesso da topologia (entrada SW_{acesso}) e o conjunto de *switches* de agregação e *core* da topologia (entrada $SW_{FatTree}$).

A partir dos conjuntos de entrada, o DSW cria como saída dois conjuntos de *switches*. No primeiro, são reunidos os *switches* que estão com as interfaces em modo de espera (SW_{espera}), e no segundo, os *switches* que estão com as interfaces desligadas ($SW_{desligados}$). O primeiro passo do DSW é obter a quantidade dos *switches* da topologia (linha 1). Em seguida o DSW passa todos os *switches* para o modo de eficiência energética, a partir da configuração das interfaces dos *switches* de acesso para o modo de espera e desligamento daquelas pertencentes aos *switches* de agregação e *core* (linhas 2 e 3). O trecho entre as linhas de 4 a 16 consiste em um *loop* infinito, no qual a cada tempo δ (10 segundos) são verificadas as tabelas de fluxo de todos os *switches* da topologia. Contudo, como apenas os *switches* tem conexão direta com o controlador, se essas interfaces fossem completamente desligadas, não haveria uma forma de os *switches* da camada de acesso "perceberem" que determinados *hosts* desejariam trafegar dados. Por tais motivos, as interfaces dos *switches* de acesso devem permanecer em modo espera.

Algoritmo 1: Desativação de *Switches* (DSW)

Entrada: $SW_{acesso}, SW_{FatTree}$
Saída: SW_{espera}, $SW_{desligados}$
1 $qtde_{switches}$ = n(SW_{acesso}) + n($SW_{FatTree}$);
2 colocar todos os *switches* de SW_{acesso} em modo de espera e incluí-los em SW_{espera};
3 colocar todas os *switches* de $SW_{FatTree}$ em modo desligado e incluí-los em $SW_{desligados}$;
4 **para** *(cada δ)* **faça**
5 **para** *(i de 1 até $qtde_{switches}$)* **faça**
6 $qtde_{fluxos}$ = quantidade de entradas na *Flow Table* do *switch*[i];
7 **se** *($qtde_{fluxos}$ = 0)* **então**
8 **se** *($switch[i] \in$ ao conjunto SW_{acesso})* **então**
9 Colocar interfaces de SW_{acesso} em modo de espera e incluí-lo em SW_{espera};
10 **senão**
11 Colocar interfaces de $SW_{FatTree}$ em modo desligado e incluí-lo em $SW_{desligados}$;
12 **fim se**
13 **fim se**
14 **fim se**
15 **fim para**
16 **fim para**

3.2. Algoritmo Ativação de *Switches* (ASW)

O algoritmo de Ativação de *Switches* (ASW) — Algoritmo 2 — aproveita a visão global da rede para explorar os múltiplos caminhos existentes entre dois *hosts*. Assim, é possível alocar fluxos de dados em mais de um caminho distinto de forma a utilizar, da melhor maneira possível, toda a largura de banda que a topologia tem a oferecer. A partir da visão global da rede, o ASW não somente maximiza o uso de recursos da infraestrutura como também melhora o desempenho ao agregar seus fluxos em canais com menor tempo de resposta.

Quando há uma solicitação de tráfego, o ASW recebe como entrada os valores referentes aos *hosts* de origem e destino da comunicação, o conjunto de *switches* que estão em modo espera, o conjunto de *switches* que estão em modo desligado (recebidos do Algoritmo DSW) e, por fim, os limiares mínimo e máximo para a quantidade de sub-fluxos por interfaces (variáveis h_{origem}, $h_{destino}$, SW_{espera}, $SW_{desligados}$, α e β, respectivamente). A partir destes parâmetros, o ASW computa o *switch* que conecta o *host* de origem (SW_{origem}), o *switch* que conecta o *host* de destino ($SW_{destino}$), o POD (*Point of Delivery*) do *host* de origem (pod_{origem}) e o POD do *host* de destino ($pod_{destino}$). Em seguida, quatro conjuntos são criados contendo: (i) os *switches* de acesso (SW_{acesso}); (ii) os *switches* de agregação ($SW_{agregacao}$); (iii) os *switches core* (SW_{core}); e (iv) os *switches* que devem ficar ativos durante a comunicação (SW_{ativos}). O ASW, então, adiciona os *switches* de origem e destino no conjunto dos *switches* que devem estar ativos durante a comunicação. Se o *switch* de origem for igual ao *switch* de destino, o ASW adiciona o identificador de apenas um deles no conjunto de *switches* ativos (cláusula *distinct*).

Conforme os *hosts* de origem e destino, o algoritmo ASW se comporta de modo diferente de acordo com as três formas possíveis de comunicação: (i) comunicação entre *hosts* vizinhos conectados ao mesmo *switch* (Comunicação *Intra-Switch*); (ii) comunicação entre *hosts* pertencentes ao mesmo POD (Comunicação *Intra-POD*); e (iii) comunicação entre *hosts* pertencentes a PODs diferentes (Comunicação *Inter-POD*). A Figura 1 ilustra esses três tipos de comunicação em uma rede que segue a topologia *Fat Tree* K4.

Na comunicação *Intra-Switch* existe apenas um único caminho entre dois *hosts* vizinhos (linha 7). Como a vantagem do MPTCP está na possibilidade de envio de dados por caminhos distintos, este é o único caso em que o mesmo não pode ser utilizado. O algoritmo então prossegue pela definição da quantidade de sub-fluxos por interface ($fl_{interface}$) com um valor limiar mínimo α, utilizado para configuração da quantidade de sub-fluxos por interface a partir da linha 20. A Figura 1(a) apresenta um exemplo desse tipo de comunicação em que os *hosts* `h1` e `h2` estão conectados no mesmo *switch* `s13`. Nesta situação, todos os demais *switches* são passíveis de serem desligados pelo DSW.

Na comunicação *Intra-POD* o tráfego ocorre entre *hosts* que, embora estejam interligados por *switches* de acesso diferentes, estão dentro do mesmo POD. Neste caso, como o ASW já incluiu os *switches* de origem e destino no conjunto de *switches* que devem estar ativos (linha 6), ele adiciona também neste conjunto os *switches* de agregação do POD em que está havendo a comunicação. Além disso, o ASW define o valor da variável $fl_{interface}$ com a quantidade destes *switches* de agregação (linha 12). O algoritmo, então, prossegue configurando a quantidade de sub-fluxos por interfaces dos *switches* que estão no conjunto de *switches* ativos e verificando se os mesmos estão em modo de espera ou desligados (linhas 20 a 36). A Figura 1(b) apresenta um exemplo de comunicação

Algoritmo 2: Ativação de *Switches* (ASW)

Entrada: $h_{origem}, h_{destino}, SW_{espera}, SW_{desligados}, \alpha, \beta$

1 $SW_{espera} \Leftarrow$ conjunto dos *switches* que estão em espera (recebido do Algoritmo DSW);
2 $SW_{desligados} \Leftarrow$ conjunto dos *switches* que estão desligados (recebido do Algoritmo DSW);
3 $SW_{ativos} \Leftarrow$ conjunto de *switches* ativos (inicialmente $\emptyset$);
4 sw_{origem} = *switch* de origem baseado em h_{origem}; $sw_{destino}$ = *switch* de destino baseado em $h_{destino}$;
5 pod_{origem} = POD de origem baseado em sw_{origem}; $pod_{destino}$ = POD de destino baseado em $sw_{destino}$;
6 Incluir no conjunto SW_{ativos} os *switches* de SW_{origem} e $SW_{destino}$ (*distinct*);
7 **se** *(*$sw_{origem} = sw_{destino}$*)* **então**
8 $fl_{interface} = \alpha$ /* Há apenas 1 *switch* no caminho de comunicação */;
9 **senão se** *(*$pod_{origem} = pod_{destino}$*)* **então**
10 /* A quantidade de caminhos é igual a quantidade de *switches* de agregação no POD */
11 Incluir no conjunto SW_{ativos} os *switches* de agregação do POD de origem;
12 $fl_{interfaces}$ = quantidade de switches de agregação do POD de origem;
13 **senão**
14 /* A quantidade de caminhos é igual a quantidade de *switches core* da topologia */
15 Incluir no conjunto SW_{ativos} os *switches* de agregação do POD de origem e POD de destino, e os switches core da topologia;
16 $fl_{interfaces}$ = quantidade dos *switches core* da topologia;
17 **fim se**
18 **fim se**
19 **fim se**
20 **se** *(*$fl_{interface} = \alpha$*)* **então**
21 Configura a quantidade de sub-fluxos por interface igual ao limiar mínimo (α);
22 **senão se** *(*$fl_{interface} \leq \beta$*)* **então**
23 Configura a quantidade de sub-fluxos por interface igual ao valor de $fl_{interface}$;
24 **senão**
25 Configura a quantidade de sub-fluxos por interface igual ao limiar máximo (β);
26 **fim se**
27 **fim se**
28 **fim se**
29 **para** *(i de 1 até n(*SW_{ativos}*))* **faça**
30 **se** *(*$SW_{ativos}[i] \in SW_{espera}$*)* **então**
31 Retirar interfaces de $SW_{ativos}[i]$ do modo espera;
32 **fim se**
33 **se** *(*$SW_{ativos}[i] \in SW_{desligados}$*)* **então**
34 Retirar interfaces de $SW_{ativos}[i]$ do modo desligado;
35 **fim se**
36 **fim para**

Intra-POD. Note que a comunicação entre os *hosts* `h1` e `h4` pode acontecer por dois caminhos distintos, dado que os referidos *hosts* são interligados por *switches* diferentes.

Nos casos do tráfego acontecer entre *hosts* que estejam em PODs diferentes, o ASW adiciona também, no conjunto de *switches* ativos, os *switches* de agregação dos PODs de origem e destino e os *switches core* da topologia. Além disso, define o valor dos sub-fluxos com a quantidade destes *switches* core (linhas 20 a 27). A Figura 1(c) apresenta um exemplo da comunicação *Inter-POD*. Neste exemplo, a comunicação entre os *hosts* `h1` e `h16` acontece por *switches* diferentes, localizados em PODs também diferentes. Assim, existem quatro caminhos distintos possíveis em uma eventual comunicação entre

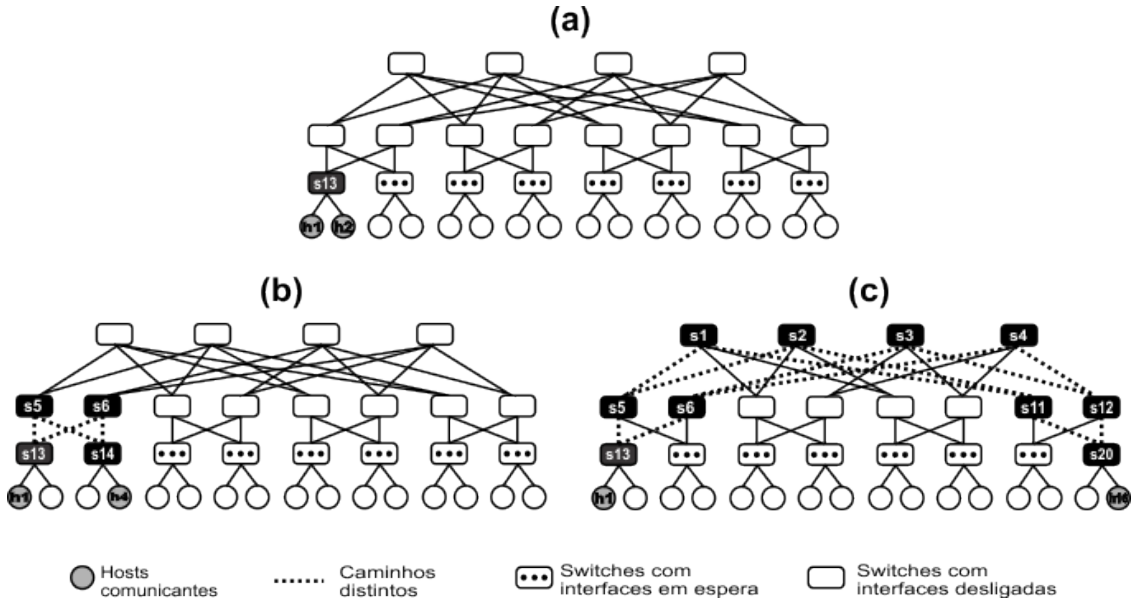

Figura 1. Formas de comunicação na topologia *FatTree*: (a) Intra-Switch; (b) Intra-POD; e (c) Inter-POD.

os *hosts*. Por fim, a partir da linha 23, o ASW configura a quantidade de sub-fluxos por interfaces e verifica se os *switches* que estão no conjunto de *switches* ativos estão em modo de espera ou desligados.

4. Experimentação: emulação e testes realizados

Os algoritmos descritos na Seção 3 foram avaliados através de redes de centro de dados definidas por *software* nas topologias *Fat Tree*, implementadas por meio da plataforma *OpenFlow* e o MPTCP. O *OpenFlow* oferece o controle centralizado da rede, permitindo maior flexibilidade na definição de regras das tabelas de fluxos no plano de encaminhamento formado pelos *switches*, assim como, na identificação dos possíveis caminhos entre *hosts* da *Fat Tree*. Já o MPTCP permite melhor utilização da largura banda disponível, uma vez que possibilita o envio do tráfego por caminhos distintos. Embora o ECMP (do inglês, *Equal-Cost Multi-Path routing*) [Hopps, 2000] consiga enviar o tráfego por mais de um caminho, o seu escalonamento não realiza uma divisão justa da largura de banda. Isso faz com que alguns *links* fiquem sub-utilizados e possuam maior taxa de perdas de pacotes e retransmissões durante a comunicação [Alizadeh et al., 2010].

Nos experimentos realizados, a solução descrita na Seção 3 foi analisada em comparação com uma rede utilizando o ECMP, a partir da média de 50 execuções, conforme [Paasch et al., 2013], com intervalos de confiança no nível de 95%, observando as seguintes métricas: (i) consumo de energia por *switch* (medida em *Watts* / Hora); (ii) *goodput*; (iii) latência; (iv) taxa de perdas; e (vi) quantidade de vezes em que as interfaces dos *switches* ficam em modo de espera ou desligadas.

A adoção do ECMP como referência de comparação deve-se à dificuldade de reproduzir a implementação de outros trabalhos da literatura. Ademais, a estratégia BEEP não pode ser comparada, integralmente, com outras soluções por ter o objetivo de buscar o equilíbrio entre Energia, Redundância e Desempenho.

4.1. Ambiente de testes

As tecnologias utilizadas no ambiente de emulação dos experimentos foram: (i) *PC Core* i5, 2.3*Mhz*, 8*GB* de *RAM*; (ii) Sistema Operacional *GNU/Linux*, *x*64, versão do *kernel* 3.17.2; (iii) *MultiPath* TCP em *Kernel* do *Linux*[2], versão estável 0.89.2; (iv) *Mininet*[3], versão 2.1.0p1; (v) *Switch Virtual OpenFlow (Open vSwitch)*[4], versão 2.0.2; (vi) Protocolo *OpenFlow*[5], versão 1.0; (vii) Controlador *OpenFlow* RYU[6], versão 3.15; e (viii) Gerador de tráfego D-ITG[7], versão 2.4.6. A escolha por estas ferramentas deveu-se à ampla documentação existente e à facilidade de implementação.

O desenvolvimento das funcionalidades do controlador RYU envolveu os seguintes módulos, que são executados simultaneamente: (i) **Switch MPTCP** — configurado para executar o MPTCP. Este módulo insere, mede e coleta indicadores da utilização dos *switches OpenFlow* da rede. Ao ser executado, esse módulo reduz a velocidade de todas as interfaces dos *switches* da topologia passando-as de 10Gbps para 1Gbps nos *switches* da camada *core* e de 1Gbps para 100Mbps os demais; (ii) **DSW** — desenvolvido para executar o algoritmo DSW; (iii) **ASW** — desenvolvido para executar o algoritmo ASW.

4.1.1. Configuração do MPTCP

O MPTCP foi configurado com o gerenciador de caminhos `ndiffports`. Este gerenciador foi projetado especificamente para utilização em redes de centros de dados com a função de explorar seus vários caminhos distintos [Paasch et al., 2013]. Essa configuração permite que as interfaces de rede possuam mais de um fluxo.

A possibilidade de criar mais de um fluxo por interface pode fazer o MPTCP perder o seu desempenho devido a problemas com a fragmentação do protocolo IP. A quantidade de fluxos muito elevada pode resultar em erros no cálculo do MTU. Por tais motivos, a fim de evitar tais perdas de desempenho, foram configurados um limiar mínimo e um máximo para a quantidade máxima de sub-fluxos por interfaces. Os limiares mínimo e máximo são passados para o controlador através das variáveis α (com valor 1) e β (com valor 5), definidas no algoritmo ASW. O limiar mínimo é utilizado quando há apenas um caminho para a conexão, caso contrário, de acordo com a origem e o destino da comunicação, o valor máximo pode ser até 5. Independente da topologia, se a quantidade de sub-fluxos por interface for maior que 5, o MPTCP apresenta perda de desempenho por conta do *overhead* gerado nas interfaces dos *hosts* comunicantes.

A partir da informação sobre o par de *hosts* envolvidos na determinada comunicação, os sub-fluxos por interfaces são configurados como:

- **Comunicação *Intra-Switch***: nesta comunicação só há um caminho possível. Por este motivo estas conexões utilizam o limiar mínimo para a quantidade de sub-fluxos por interface que tem o valor 1;

[2]http://multipath-tcp.org
[3]http://mininet.org/
[4]http://openvswitch.org/
[5]http://archive.openflow.org/
[6]http://osrg.github.io/ryu/
[7]http://traffic.comics.unina.it/software/ITG/

- **Comunicação *Intra-POD***: nesta comunicação há mais de um caminho distinto possível. Assim, o valor do limiar será configurado a partir da quantidade de *switches* de agregação presentes no POD em que há a comunicação. Se a quantidade de *switches* de agregação for menor ou igual ao limiar máximo (adotado no experimento como 5), a quantidade de sub-fluxos por interfaces será igual à quantidade de *switches* de agregação que o POD possui. Caso contrário, independente da quantidade de *switches* de agregação que o POD possua, o limiar é configurado com o valor máximo;
- **Comunicação *Inter-POD***: nesta comunicação também há mais de um caminho distinto. Assim, se a quantidade de *switches core* for menor ou igual ao limiar máximo, o valor da quantidade de sub-fluxos a ser configurado nas interfaces será igual a quantidade de *switches core* presentes na topologia. Caso contrário, independente da quantidade de *switches core* que a topologia possua, o valor será configurado com o valor máximo.

Outro aspecto relevante na configuração do MPTCP consiste na escolha do esquema de controle de congestionamento. Neste trabalho, foi implementado o esquema de controle de congestionamento *wVegas* [Cao et al., 2012]. Trata-se de um esquema baseado no tempo de atraso que possui um balanceador de carga melhorado. O *wVegas* consegue promover o "*princípio da igualdade no congestionamento*", em que o MPTCP escalona os pacotes nos caminhos com menor tempo de resposta. O *wVegas* também tem a vantagem de fazer com que até os *links* considerados mais lentos consigam obter um bom desempenho através de seu algoritmo iterativo.

4.1.2. Configuração dos *Switches*

O consumo de energia de um *switch* é dividido entre os seus quatro principais componentes. São eles: o *chassis*, o *switch-fabric*, as *linecards* e as portas. O *chassi* é responsável pelo sistema de refrigeração, dentre outros. O *switch-fabric* é responsável por aprender os MACs e/ou IPs das interfaces e manter as tabelas lógicas do equipamento. A *linecard* é responsável por encaminhar os pacotes do *switch-fabric* para as portas e as portas são responsáveis por realizar as comunicações na rede.

Como a *Fat Tree* opera em uma arquitetura em camada de 3 níveis, geralmente são utilizadas soluções de *switches* com diferentes velocidades. Frequentemente é adotada a velocidade de 10Gbps para *switches* da camada *core* e 1Gbps para os *switches* das camadas de agregação e acesso. Entretanto, como a solução proposta visa reduzir estas velocidades nas interfaces, o consumo dos *switches* de 100Mbps também foi considerado na avaliação. Os valores de consumo utilizados nos experimentos, descritos na Tabela 1, foram adotados de acordo com a linha de *switches OpenFlow* da HP [Packard, 2014].

O modo de espera realizado nas interfaces implementa o WoP (*Wake on Packet*) [Ananthanarayanan e Katz, 2008]. Neste modo, as interfaces ficam em estado de espera e só são reativadas quando há uma solicitação de tráfego. Esse pacote "reativador" da interface se assemelha ao pacote mágico utilizado pelo WoL (*Wake on Lan*), no qual equipamentos são retirados do modo de espera através do processamento de sua interface de rede. No WoP o *switch* deixa a interface consumindo apenas o mínimo possível para ser reativada quando houver necessidade.

Tabela 1. Parâmetros dos *Switches OpenFlow* utilizados nos experimentos.

Parâmetro	Descrição	Valores configurados nos *switches*		
		100Mbps	1Gbps	10Gbps
$Consumo_{maximo}$	Consumo com carga total	120W	130W	685W
$Consumo_{ocioso}$	Consumo no estado ocioso	98W	180W	320W
$Consumo_{porta-ativa}$	Consumo de cada porta ativa	5,4W	8W	14W
$Consumo_{porta-ociosa}$	Consumo de cada porta ociosa	3W	5W	9W
$Consumo_{porta-espera}$	Consumo de cada porta em espera	1,6W	3W	3W
$Consumo_{porta-shutdown}$	Consumo de cada porta desligada	0,1W	0,5W	0,5W
$Tempo_{shutdown-on}$	Tempo para colocar em *shutdown*	0,67ms	1ms	1ms
$Tempo_{shutdown-off}$	Tempo para retirar de *shutdown*	0,88ms	1,2ms	1,4ms
$Tempo_{espera-on}$	Tempo para colocar em modo espera	1ms	1ms	1ms
$Tempo_{espera-off}$	Tempo para retirar do modo espera	1ms	1,1ms	1,1ms
$Tempo_{mudanca-velocidade}$	Tempo para mudança de velocidade	1,2ms	1,3ms	1ms

4.1.3. Tráfego caracterizado

Conforme relatado em [Bilal et al., 2014, Benson et al., 2010], o tráfego TCP é o mais adequado para testes e avaliação de redes de centro de dados que tem por objetivo obter eficiência energética. Por tal motivo, neste trabalho foi utilizado tráfego TCP para os padrões de comunicação um-para-todos e todos-para-todos. Assim, foram geradas cargas de trabalho sintéticas, através do D-ITG (do inglês, *Distributed Internet Traffic Generator*) que se baseia na distribuição *Poisson*. No envio da carga de trabalho, o gerador de tráfego emite fluxos dos tipos "*elephant*", "*tortuoise*" e "*cheetars*".

Por fim, é preciso destacar que nos experimentos o tráfego ficou mais concentrado na camada *core*, com perdas mais freqüentes nas bordas e com 80% dentro do *rack*. Tais parametrizações tornam a geração de tráfego mais próxima de uma rede real e visam permitir que a solução proposta neste artigo seja comparável com qualquer cenários realistas de utilização [Benson et al., 2010, Chen et al., 2011].

4.2. Resultados obtidos

Nos experimentos foram utilizadas sete variantes da topologia *Fat Tree*, emuladas no *Mininet* de acordo com a descrição apresentada na Tabela 2. As velocidades dos *switches* foram configuradas com 10 Gbps para os *switches* da camada *core*, 1 Gbps para *switches* da camada de agregação e 1 Gbps para os *switches* da camada de acesso. A fim de avaliar a solução proposta em tais redes, foram realizados testes através da geração de tráfego com cargas de 1 GB, 5 GB, 10 GB e 20 GB, distribuídos entre os *hosts*.

Os resultados dos experimentos estão apresentados nas Tabelas 3, 4, 5 e 6. As colunas com a sigla ECMP se referem aos resultados obtidos sem qualquer solução para eficiência energética, ou seja, apenas com o ECMP. As colunas com BEEP se referem aos valores obtidos com a estratégia proposta neste trabalho. A coluna "Consumo dos *switches*" se refere a soma do consumo (em *Watts* / Hora), valendo salientar que: (i) quando os valores se referem ao ECMP, os *switches core* operam com velocidade de 10 Gbps, e os demais *switches* a 1 Gbps; (ii) quando os valores se referem a estratégia BEEP, os *switches core* operam com velocidade de 1 Gbps, e os demais *switches* a 100 Mbps.

Com base na Tabela 3, é possível notar que no quesito tempo de transmissão, a cada vez que a topologia é aumentada, o tempo de transmissão com a estratégia BEEP é diminuído. Esse fato se deve à melhor utilização da largura de banda e ao escalonamento

Tabela 2. Topologias *Fat Tree* emuladas nos experimentos.

Topologia	Qtde. de *Hosts*	Qtde. de *Switches*
K4	16	20
K5	20	31
K6	54	45
K7	63	61
K8	128	80
K9	144	101
K10	250	125

entre os vários caminhos distintos realizado pelo MPTCP. Dessa forma, quanto mais *switches* a topologia possuir, maior será o número de caminhos distintos que ela pode oferecer ao MPTCP, o que torna a estratégia, a partir da topologia K6, mais veloz que a solução convencional, mesmo operando com velocidade inferior. Outra questão a destacar é que à medida que a topologia cresce, diminui o percentual de economia de energia obtido com a BEEP. Os resultados também comprovam outra vantagem da estratégia com relação ao consumo médio por *switch*, *Goodput* e as perdas de pacotes. Nos experimentos, em nenhuma situação ele obteve valor pior que a infraestrutura operando apenas com o ECMP. Este ganho se deve ao melhor balanceamento de carga e gerenciamento de congestionamento realizado pelo MPTCP. Por fim, ressaltamos que por conta do maior tempo gasto para o tráfego da carga de trabalho de 1GB na topologia ter sido 6,1 segundos, o algoritmo DSW (Algoritmo 1) não colocou em modo de espera ou desligou nenhuma das interfaces dos *switches*.

Tabela 3. Experimento com carga de 1GB.

1GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por *Switch* (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	6,1	8,3	2,2	4.129,0	2.283,0	44,7	206,45	114,15	89	93	11	9	0	0
K5	5,6	5,7	0,1	6.421,0	3.884,0	39,5	321,05	194,20	89	94	10	8	0	0
K6	4,3	4,0	-0,3	8.427,0	5.128,0	39,1	187,27	113,96	88	91	12	7	0	0
K7	3,2	2,7	-0,5	12.111,0	7.665,0	36,7	198,54	125,66	88	93	13	8	0	0
K8	2,8	2,5	-0,3	15.489,0	10.123,0	34,6	193,61	126,54	88	93	11	7	0	0
K9	2,2	1,8	-0,4	21.321,0	14.889,0	30,2	211,10	147,42	89	94	10	8	0	0
K10	1,5	0,9	-0,6	27.931,0	19.376,0	30,6	223,45	155,01	89	92	11	8	0	0

Com base na Tabela 4, é possível observar que análogo aos resultados dos experimentos com a carga de trabalho de 1GB, a estratégia BEEP mostrou resultados melhores nos quesitos tempo de transmissão, consumo médio por *switch*, taxa de *goodput* e perdas. Como o maior tempo gasto para realizar o tráfego da carga de 5GB foi de 16,2 segundos, apenas 1 *switch* foi desligado e 1 *switch* entrou em modo de espera durante os experimentos nas topologias K4 e K5.

Na Tabela 5, é possível notar que assim como nos experimentos com as cargas anteriores, a solução com a estratégia BEEP continua melhor em relação a infraestrutura com o ECMP. É interessante destacar que o percentual de economia na topologia K5, que contém 31 *switches*, foi maior do que na topologia K4, que contém 20 *switches* (fato justificado pelo número maior de interfaces desligadas pelo Algoritmo DSW na topologia K5).

Com base na Tabela 6, nota-se que o ganho com a estratégia BEEP tem com-

Tabela 4. Experimento com carga de 5GB.

5GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por *Switch* (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	16,2	19,9	3,7	5.384,0	3.476,0	35,4	269,20	173,80	88	91	12	91	1	1
K5	13,8	14,4	0,6	8.003,0	5.008,0	37,4	400,15	250,40	88	92	13	92	1	1
K6	12,9	12,7	-0,2	10.120,0	6.139,0	39,3	224,89	136,42	89	92	15	92	0	0
K7	10,7	9,7	-1,0	13.743,0	9.039,0	34,2	225,30	148,18	89	93	15	93	0	0
K8	9,4	9,0	-0,4	17.347,0	11.075,0	36,2	216,84	138,44	86	94	16	94	0	0
K9	8,3	7,5	-0,8	23.876,0	15.627,0	34,5	236,40	154,72	85	91	15	91	0	0
K10	7,0	5,9	-1,1	29.654,0	19.221,0	35,2	237,23	153,77	85	91	15	91	0	0

Tabela 5. Experimento com carga de 10GB.

10GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por Switch (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	33,0	41,0	8,0	6.876,0	4.228,0	38,5	343,80	211,40	88	91	10	7	4	3
K5	31,8	38,0	6,2	9.327,0	5.550,0	40,5	466,35	277,50	87	90	9	8	6	1
K6	29,5	29,0	-0,5	11.298,0	7.332,0	35,1	251,07	162,93	89	92	12	9	4	1
K7	27,6	24,3	-3,3	14.498,0	9.418,0	35,0	237,67	154,39	87	94	14	9	3	0
K8	26,8	21,8	-5,0	19.287,0	12.511,0	35,1	241,09	156,39	88	92	14	8	1	0
K9	25,2	19,5	-5,7	25.554,0	17.038,0	33,3	253,01	168,69	88	92	16	8	2	0
K10	20,1	14,8	-5,3	30.754,0	20.287,0	34,0	246,03	162,30	88	93	16	7	1	0

portamento similar com as demais cargas, exceto por um detalhe: nas cargas emuladas anteriormente, em que o MPTCP começa a ser mais veloz que a infraestrutura convencional da topologia a partir da topologia K6 em diante, quando a carga foi de 20GB. A topologia K6 com o MPTCP obteve latência um pouco maior que a infraestrutura com apenas o ECMP, o que leva a crer que quanto maior for a carga de trabalho, a latência pode se tornar maior com a estratégia, dada a diferença entre as velocidades das soluções.

Como neste experimento a carga de trabalho foi maior do que as demais e, consequentemente, o seu tempo gasto para transmissão foi maior, ficou mais evidente a importância do algoritmo DSW. O mesmo foi capaz de desligar ou colocar em espera os *switches* que ficaram ociosos durante o referido experimento.

Tabela 6. Experimento com carga de 20GB.

20GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por Switch (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	55,0	72,0	17,0	7.438,0	4.611,0	38,0	371,90	230,55	89	93	11	7	12	4
K5	54,3	66,0	11,7	10.180,0	6.343,0	37,7	509,00	317,15	89	92	13	8	8	2
K6	53,8	58,0	4,2	12.132,0	8.028,0	33,8	269,60	178,40	88	92	14	8	6	3
K7	50,0	46,3	-3,7	16.268,0	10.665,0	34,4	266,69	174,84	88	93	15	8	3	3
K8	48,8	45,1	-3,7	21.568,0	13.697,0	36,5	269,60	171,21	87	92	12	9	3	1
K9	46,9	42,1	-4,8	27.052,0	18.033,0	33,3	267,84	178,54	87	93	14	9	3	0
K10	44,7	39,5	-5,2	33.700,0	23.128,0	31,4	269,60	185,02	85	92	14	9	2	1

4.3. Avaliação geral dos resultados obtidos

Com base nos resultados obtidos, destacamos que: (i) devido à proposta utilizar o maior número possível de equipamentos (caminhos) durante a transmissão a redundância das conexões é mantida; (ii) como o MPTCP utiliza melhor a largura de banda (principalmente por distribuir o tráfego em caminhos menos congestionados) ele possui menor

número de perdas e retransmissões, o que torna as comunicações rápidas, seguras e com melhor consumo energético. Uma vez que os *switches* trafegam mais rápido, ficam mais tempo ociosos, o que os tornam passíveis de entrarem em modo de eficiência energética); e (iii) quanto maior for o número de caminhos distintos existente entre o par origem e destino, melhor é o desempenho do MPTCP.

Dado o exposto, ressaltamos que a utilização da estratégia BEEP traz benefícios quando utilizada em redes de centro de dados, sendo que sua utilização proporciona um equilíbrio entre o consumo de energia e a redundância da topologia, além de aumentar o desempenho das conexões.

5. Conclusão e trabalhos futuros

Com o aumento da escala de centro de dados, os sistemas de gestão de energia centralizados para redução do consumo de energia nos equipamentos destas redes possuem problemas de escalabilidade. Com base na topologia hierárquicas *Fat Tree* e padrões de tráfego de centro de dados, propusemos e avaliamos a solução para redução do consumo de energia através da utilização de SDN, *OpenFlow* e o *MultiPath* TCP sem violar as restrições de conectividade e de redundância.

Através dos resultados experimentais obtidos por emulação (Seção 4.2), foi verificado que a estratégia BEEP mostra-se competitiva diante das soluções atualmente existentes para infraestrutura de redes de centro de dados. Através da exploração do controle centralizado obtido pelo controlador *OpenFlow*, aliado ao desempenho oferecido pelo MPTCP e pelo seu modo de escalonar e controlar congestionamentos, a solução proposta consegue aliar eficiência energética com melhor distribuição de banda e recursos.

Para trabalhos futuros é necessário fazer um estudo com uma gama maior de cargas e de topologias para centro de dados, realizar testes de escabilidade, testar o melhor cenário para aplicação de regras pró-ativas e reativas, bem como emular o tráfego mais próximo possível de ambientes de produção. Uma outra forma de avaliar melhor a solução seria implementar um mecanismo capaz de não só reduzir as velocidades das interfaces, mas fazer estes ajustes de acordo com a caracterização do tráfego e volume de carga de trabalho nos *switches* em comunicações originadas dentro e fora do centro de dados.

Agradecimentos

Os autores agradecem o apoio da CAPES, CNPq e FAPERJ.

Referências

Alizadeh, M., Greenberg, A., Maltz, D. a., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., e Sridharan, M. (2010). Data center TCP (DCTCP). *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM '10*, page 63.

Ananthanarayanan, G. e Katz, R. (2008). Greening the Switch. *HotPower. Pages 1-5*.

Barré, S., Paasch, C., e Bonaventure, O. (2011). Multipath tcp: From theory to practice. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING'11, pages 444–457, Berlin, Heidelberg. Springer-Verlag.

Benson, T., Anand, A., Akella, A., e Zhang, M. (2010). Understanding data center traffic characteristics. *SIGCOMM Comput. Commun. Rev.*, 40(1):92–99.

Bilal, K., Malik, S. U. R., Khalid, O., Hameed, A., Alvarez, E., Wijaysekara, V., Irfan, R., Shrestha, S., Dwivedy, D., Ali, M., Shahid Khan, U., Abbas, A., Jalil, N., e Khan, S. U. (2014). A taxonomy and survey on Green Data Center Networks. *Future Generation Computer Systems*, 36:189–208.

Cao, Y., Xu, M., e Fu, X. (2012). Delay-based congestion control for Multipath TCP. *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–10.

Chen, K., Hu, C., Zhang, X., Zheng, K., Chen, Y., e Vasilakos, A. (2011). Survey on routing in data centers: insights and future directions. *Network, IEEE*, 25(4):6–10.

Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., e Sengupta, S. (2009). Vl2: A scalable and flexible data center network. *SIGCOMM Comput. Commun. Rev.*, 39(4).

Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., e Lu, S. (2009). Bcube: A high performance, server-centric network architecture for modular data centers. *SIGCOMM Comput. Commun. Rev.*

Hammadi, A. e Mhamdi, L. (2014). A survey on architectures and energy efficiency in Data Center Networks. *Computer Communications*, 40:1–21.

Heller, B., Seetharaman, S., e Mahadevan, P. (2006). ElasticTree : Saving Energy in Data Center Networks. *Energy*, page 17.

Hopps, C. (2000). Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992 (Informational).

Huang, L., Jia, Q., Wang, X., Yang, S., e Li, B. (2011). PCube: Improving power efficiency in data center networks. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 65–72.

IEEE (2014). IEEE P802.3az Energy Efficient Ethernet Task Force. Disponível em: www.ieee802.org/3/az/. Acessado em 06/11/2014.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., e Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Nordman, B. (2007). Energy Efficient Ethernet: Outstanding Questions. *IEEE 802 interim meeting*, 1(510):1–10.

Paasch, C., Khalili, R., e Bonaventure, O. (2013). On the benefits of applying experimental design to improve Multipath TCP. *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies - CoNEXT '13*, pages 393–398.

Packard, H. H. (2014). Software-defined networking. Disponível em: www8.hp.com/us/en/networking/solutions/technology/sdn/. Acessado em 23/10/14.

Zhang, M., Yi, C., Liu, B., e Zhang, B. (2010). Greente: Power-aware traffic engineering. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 21–30.

Zhang, Y. e Ansari, N. (2012). Hero: Hierarchical energy optimization for data center networks. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2924–2928.