PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESTRATÉGIAS PARA REDUÇÃO DO CONSUMO DE ENERGIA EM REDES DE DATA CENTER

MARCELO DA SILVA CONTERATO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Tiago Coelho Ferreto

Porto Alegre
2016

Dados Internacionais de Catalogação na Publicação (CIP)

C761e Conterato, Marcelo da Silva

Estratégias para redução do consumo de energia em redes de Data Center / Marcelo da Silva Conterato. – 2016.
79 f.

Diss. (Mestrado) – Faculdade de Informática, PUCRS.
Orientador: Prof. Dr. Tiago Coelho Ferreto.

1. Redes de Computadores. 2. Energia Elétrica – Consumo. 3. Informática. I. Ferreto, Tiago Coelho. II. Título.

CDD 23 ed. 004.65

Ramon Ely CRB 10/2165
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Estratégias para Redução do Consumo de Energia em Redes de Data Center" apresentada por Marcelo da Silva Conterato como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 15 de janeiro de 2016 pela Comissão Examinadora:

Prof. Dr. Tiago Coelho Ferreto– Orientador — PPGCC/PUCRS

Prof. Dr. Fernando Luís Dotti – — PPGCC/PUCRS

Prof. Dr. Marcelo Veiga Neves - — FACIN/PUCRS

Homologada em 02/06/2016, conforme Ata No. 011 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

Campus Central
Av. Ipiranga, 6681 – P. 32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320–3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

DEDICATÓRIA

Dedico este trabalho a minha família, em especial a minha esposa Christiane e ao meu pequeno Bernardo, que foram a inspiração para que este trabalho tivesse êxito.

"É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota. (Theodore Roosevelt)"

AGRADECIMENTOS

A Deus por me amparar nos momentos difíceis, me dar força interior para superar as dificuldades, mostrar os caminho nas horas incertas e me suprir em todas as minhas necessidades.

Ao meu orientador Prof. Dr. Tiago Coelho Ferreto, por ter me guiado no caminho do conhecimento e por ter acreditado em mim, abrindo as portas do mundo acadêmico.

Aos Profs. Dr. Fernando Luís Dotti e Dr. Marcelo Veiga Neves, pela presença na banca e contribuições à dissertação.

À minha família, a qual amo muito, pelo carinho, paciência e incentivo.

Aos amigos que fizeram parte desses momentos, sempre me ajudando e incentivando.

Aos colegas de trabalho na Casa Civil do Estado do Rio Grande do Su, pela motivação e pelo apoio nos momentos finais deste trabalho.

Aos colegas da Direção e Coordenação de Curso na Faculdade Senac Porto Alegre, pela motivação e apoio incondicional em todos os momentos deste trabalho.

ESTRATÉGIAS PARA REDUÇÃO DO CONSUMO DE ENERGIA EM REDES DE DATA CENTER

RESUMO

Atualmente, Data Centers têm seus recursos utilizados em cargas extremamente altas, o que leva à utilização de recursos de forma descontrolada. Isto mantém um elevado consumo de energia, até mesmo em momentos em que a demanda de tráfego é baixa. Até agora, os pesquisadores realizaram diversas pesquisas no campo da eficiência energética para Data Centers, no entanto, a maior parte dos resultados se concentra em dois componentes principais: servidores e sistemas de refrigeração. Neste trabalho, são propostas estratégias para configuração da rede de Data Center visando reduzir o consumo de energia, através do paradigma de SDN (*Software Defined Networking*). Tais estratégias foram combinadas com técnicas de redução do consumo de energia e avaliadas em comparação ao consumo de energia em ambientes sem preocupação com economia de energia. Além disso, o impacto de diferentes tamanhos de topologias fat-tree e as estratégias propostas foram simuladas e comparadas quanto a economia de energia apresentada. Ao aplicarmos os fatores de sobrecarga de 1:5 e 1:20, a taxa de economia de energia na rede chegou a 70,02% com uma topologia fat-tree de tamanho k = 12, e a 64,82% com uma topologia fat-tree de k = 8, em comparação com a rede tradicional.

Palavras-Chave: SDN; Redes de Data Center; Consumo de Energia em Redes; Fat-tree.

STRATEGIES FOR REDUCING ENERGY CONSUMPTION IN DATA CENTER NETWORKS

ABSTRACT

Currently, Data Centers have their resources used at extremely high loads, which leads to the uncontrolled use of resources. It maintains a high energy consumption, even at times when traffic demand is low. At this time, researchers are performing several researches in energy efficiency for data center, however, most of the results were focused on two major components: servers and cooling systems. In this study, we propose strategies for data center network configuration to reduce energy consumption through the Software Defined Networking paradigm. Such strategies were combined with techniques to reduce energy consumption and evaluated compared to the power-agnostic environments. Besides, the impact of different fat-tree topology sizes and proposed strategies were simulated and compared taking into account the presented energy savings. By applying the overload factors of 1:5 and 1:20, energy-saving rate on the network reached 70.02% for a fat-tree topology size of $k = 12$, and 64.82% for a fat-tree topology size of $k = 8$, when compared to a traditional network.

Keywords: SDN; Data Center Network; Energy Consumption in Networks; Fat-tree.

LISTA DE ALGORITMOS

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A Internet e a explosão de informações desde meados da década de 90 foram fatores que motivaram a rápida expansão dos Data Centers. Para lidar com o crescimento constante de novas aplicações e serviços de TI, cada vez mais Data Centers têm sido construídos, ampliados e utilizados ao longo da última década. Com este crescimento, o consumo de energia tornou-se significativo. Por exemplo em 2006 o custo do consumo de energia em infraestrutura de TI nos EUA foi estimada em 4,5 bilhões de dólares com estimativa de consumo de 61 milhões de MWh por ano EUA [18].

Os Data Centers têm recebido atenção significativa como infraestrutura econômica para o armazenamento de grandes volumes de dados e hospedagem de aplicações em larga escala. Hoje, grandes empresas como Google, Facebook, Yahoo! e Amazon usam rotineiramente os Data Centers para armazenamento, busca e processamento de dados em grande escala. Com a ascensão da computação em nuvem, o serviço de hospedagem nos Data Centers se tornou um negócio muito lucrativo, e que desempenha um papel crucial no futuro da tecnologia da informação. Porém, as redes de Data Centers ainda são largamente dependentes da pilha de protocolos TCP/IP, resultando em inúmeras limitações tais como: falta de isolamento de desempenho, aumento dos riscos de segurança, pouca flexibilidade de gerenciamento, suporte limitado para novas inovações na rede e uso cada vez maior de energia [15].

A computação verde [38] exige a combinação de responsabilidade social, a utilização de recursos de forma inteligente e inovação tecnológica, visando o consumo mais eficiente e com foco ambiental. Os governos têm incentivado as empresas a reavaliar suas políticas de uso dos recursos de TI. Encontrar novas formas de melhorar a eficiência energética e diminuir os custos com refrigeração não é mais uma discussão apenas nas salas de reuniões, mas uma realidade para muitas empresas e instituições. Estudos indicam que o consumo total de energia de todos os Data Centers no mundo aumentou em 56% entre 2005 e 2014, com o custo total de energia atingindo bilhões de dólares [36]. O consumo de energia deverá aumentar no futuro, resultando em um aumento no custo total de operação (TCO) [46]. A necessidade dos Data Centers por alto desempenho requer o uso consciente de energia, sendo necessário para isso uma nova arquitetura de rede mais reconfigurável, para que novas funcionalidades possam ser adicionadas de forma flexível dentro da rede, tais como o tráfego e monitoramento de energia, ou ainda, novos esquemas de encaminhamento. Uma oportunidade de implementar essas novas funcionalidades pode ser através da programação dos dispositivos de rede.

O paradigma das redes definidas por *software* (SDN – *Software Defined Networks*) [39] atende a essas demandas, pois separa fisicamente o plano de dados do plano de controle. Um controlador logicamente centralizado (plano de controle) de forma independente,

controla cada fluxo único na rede através da instalação de regras de fluxo personalizados nos *switches* que são responsáveis pelo encaminhamento (plano de dados). As regras são compostas por campos que fazem a combinação com as informações do fluxo de dados, um campo de instruções que detalha as ações a serem tomadas sobre o fluxo e os contadores que mantêm essas estatísticas do fluxo. Uma das implementações do paradigma SDN, o protocolo *OpenFlow* [47], permite através dos contadores, o monitoramento otimizado de cada fluxo do tráfego de dados na rede. Esta arquitetura tem a vantagem de proporcionar visibilidade central na rede, permitindo que vários esquemas de engenharia de tráfego possam ser implementados, bem como o controle centralizado, oferecendo uma opção de desenho atraente para as redes de Data Centers.

Quando se trata de soluções para economizar energia em redes, diferentes estratégias são possíveis, e dentre elas, é interessante investigar se otimizações possíveis na topologia da rede podem realmente endereçar essa questão. Isso se torna importante, na medida em que atualmente, as redes são projetadas para lidar com cargas de pico, isto significa que, quando as cargas são mais baixas, uma sobre-capacidade está presente na rede.

Neste trabalho, examinamos o problema da eficiência energética nas redes de Data Center e desenvolvemos um algoritmo de eficiência energética, visando reduzir o consumo de energia utilizando o paradigma de SDN. Utilizando a consolidação de fluxos através dos algoritmos propostos, dinamicamente desligamos os elementos de rede (*switches* e links) que não são utilizados, realizando também a adaptação da velocidade do link para economizar energia. Analisamos ainda o impacto da utilização de diferentes fatores de sobrecarga para aumentar a economia de energia.

Foram propostas três estratégias que realizam o mapeamento dos fluxos na rede do Data Center. A heurística utilizada foi baseada na combinação do algoritmo de Dijkstra (*Shortest Paths*) [25] com o problema *Bin-packing* [22], utilizando os algoritmos First-Fit, Best-Fit e Worst-Fit. Para cada uma das estratégias propostas, foram avaliadas diferentes configurações de rede, incluindo três diferentes tamanhos de pod's, duas configurações de rede, três cargas de redes, uma média de fluxos por host e três fatores de sobrecarga 1:1, 1:5 e 1:20 .

1.1 Organização do texto

O restante deste trabalho está estruturado da seguinte forma: o Capítulo 2 apresenta e define os conceitos e tecnologias pertinentes a este trabalho; os trabalhos relacionados são apresentados no Capítulo 3; a proposta dessa pesquisa, o modelo de consumo de energia e as estratégias de mapeamento dos fluxos na rede do Data Center são explicadas no Capítulo 4; o Capítulo 5 descreve o simulador que foi desenvolvido para a simulação

dos fluxos, a metodologia de avaliação e a análise dos resultados obtidos; os trabalhos futuros e a conclusão são expostos no Capítulo 6.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais sobre redes de Data Center, *oversubscription* em redes de Data Center, programabilidade de rede, consumo de energia em redes de Data Center e redes definidas por *software*.

2.1 Redes de Data Center

Segundo Budhiraja et al. [19], um Data Center é um repositório centralizado, seja físico ou virtual, para o armazenamento, gestão e disseminação de dados e informações organizadas em torno de uma base específica de conhecimento, ou que pertençam a uma empresa particular.

Os Data Ceters podem ser públicos ou privados. O Centro Nacional de Dados Climáticos (NCDC)[1], por exemplo, é um Data Center público que mantém o maior arquivo mundial de informações meteorológicas. Um Data Center privado pode existir dentro das instalações de uma organização ou pode ser mantido por uma empresa especializada, como por exemplo a Amazon [2]. Na verdade, cada organização tem um Data Center mesmo que em pequena escala, podendo ser referenciado como uma sala de servidores ou mesmo um *rack* de servidores.

A rede de Data Center [15] é a infraestrutura de comunicação que pode ser descrita pela topologia de rede, equipamentos de roteamento/comutação e protocolos utilizados (como por exemplo, *Ethernet* e *Internet Protocol* (IP)).

Em uma topologia convencional, o *Switch Top-of-Rack* (ToR) na camada de acesso fornece conectividade para os servidores alocados em cada *rack*. Cada *switch* de agregação (SA) na camada de agregação (por vezes referida como camada de distribuição) encaminha o tráfego da camada de acesso múltiplo (ToR) para a camada de núcleo. Cada *switch* ToR é conectado a múltiplos *switches* de agregação para redundância. A camada de agregação fornece a conectividade segura entre *switches* de agregação e roteadores de núcleo (RN) conectados à Internet. A Figura 2.1 mostra uma topologia de rede de um Data Center convencional [3].

De acordo com Hammadi et al. [31], os inúmeros problemas nos Data Centers convencionais impulsionaram os pesquisadores a propor e criar várias arquiteturas de Data Center para resolver estas questões. Os Data Centers podem ser classificados principalmente em duas classes, *switch-centric* e *server-centric*. Em *switch-centric*, os *switches* são componentes dominantes para a interconexão e o encaminhamento, enquanto que em

[1] http://www.ncdc.noaa.gov/
[2] http://www.amazon.com/

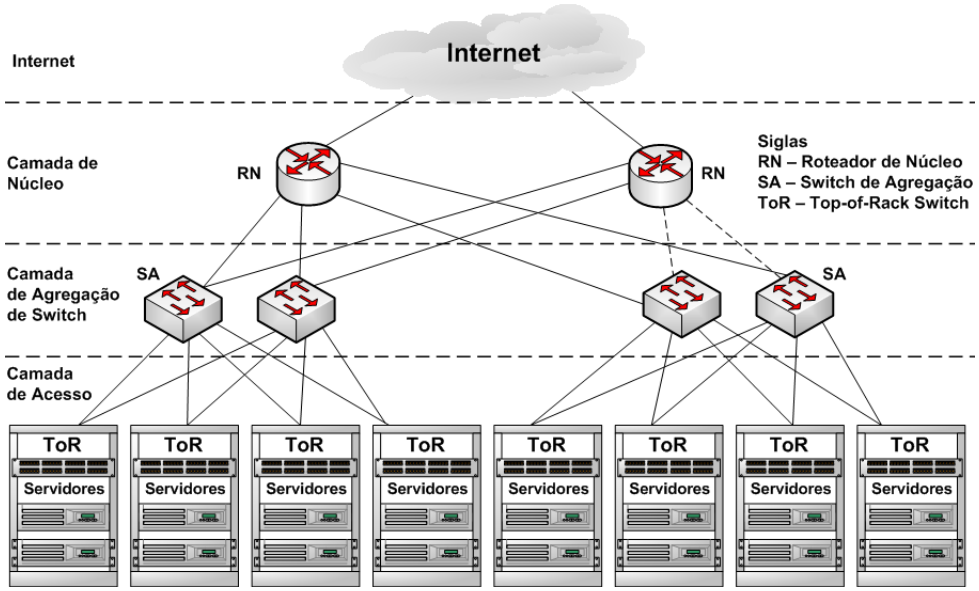


Figura 2.1 – Topologia convencional de rede de Data Center.

server-centric os servidores com múltiplas NIC *(Network Interface Cards)* participam do roteamento e das decisões de encaminhamento dos pacotes. As principais evoluções nas topologias de rede de Data Center são descritas a seguir.

A topologia Clos é uma topologia construída a partir de várias camadas de *switches* [24]. Ela é caracterizada por 3 estágios, (m, n, r) em que m é o número de *switches* de fase do meio, n é o número de entrada (saída) de portas em cada entrada (saída) *switches*, e r é o número de *switches* de entrada e de saída. Em uma rede Clos, cada *switch* intermediário tem uma ligação de entrada de cada *link* de entrada e um *link* de saída para cada opção de saída. A Figura 2.2 exemplifica a topologia de rede Clos.

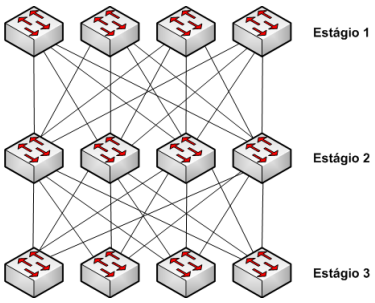


Figura 2.2 – Topologia Clos.

A topologia de rede Fat-tree [42] é um tipo especial de topologia Clos que é organizado em uma estrutura de árvore k-nária. Conforme demonstrado na Figura 2.3, a árvore possui dois tipos de conjuntos: o núcleo e os POD's (*Performance Optimized Data Center*). O núcleo é formado por *switches* que possuem cada uma de suas portas conectadas a um POD diferente. O POD é formado por *switches* de agregação e de borda, e também pelos servidores do Data Center. Os *switches* de agregação realizam a conexão entre o pod e o núcleo, e possuem conexão com os *switches* de borda e os de núcleo. Os *switches* de

borda possuem ligações com um conjunto diferente de servidores. Todos os *switches* da rede são idênticos e possuem k portas. Assim, a rede possui k pod's, sendo que cada POD possui $k/2$ *switches* de agregação e outros $k/2$ de borda [23].

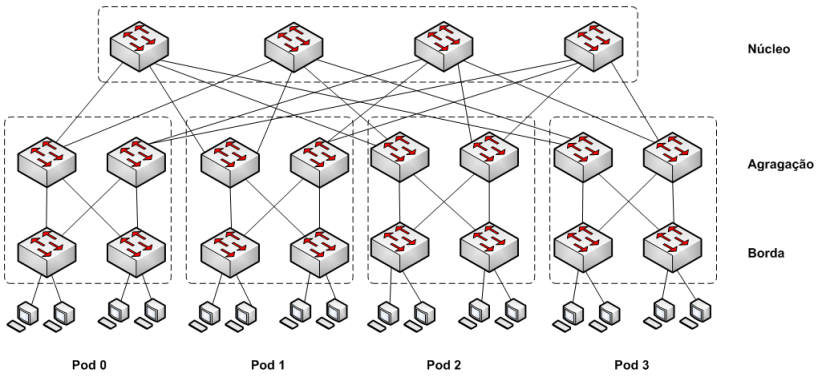


Figura 2.3 – Topologia Fat-tree (k = 4).

BCube é uma arquitetura de rede de Data Center, onde existem dois tipos de dispositivos: servidores com múltiplas interfaces de rede e *switches* que se conectam a um número (pequeno) constante de servidores. A estrutura do BCube é definida através de recursividade, em que *BCube*0 nada mais é que um conjunto de n servidores conectados a um *switch* de n portas. *BCube*1 é constituído por n *BCube*0 e *n switches* de *n* portas. De forma genérica, um $BCube_k$ ($k \geq 1$) é constituído de $nBCubek-1$ e n_k *switches* de *n* portas. Cada servidor em um *BCubek* possui $k+1$ portas, as quais são numeradas a partir do nível 0 até o nível k. Com base nestas definições, podemos perceber que um *BCubek* possui $N = nk+1$ servidores e $k+1$ níveis de *switch*, onde cada nível possui *nk switches* de *n* portas [56]. Na Figura 2.4 pode ser vista a topologia da arquitetura BCube.

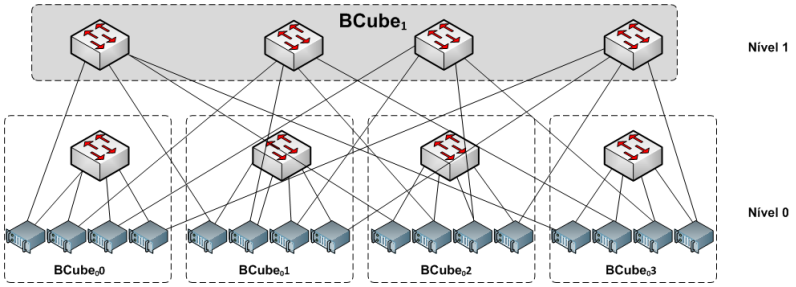


Figura 2.4 – Topologia BCube.

2.2 Sobrecarga em redes de Data Center

Atualmente existe uma demanda muito alta nas redes de Data Center, onde podemos citar o tráfego de aplicações como MapReduce e migração de servidores que colaboram para esse tráfego. Em [14], Al-Fares et al introduz a sobrecarga em redes de Data

Center como um meio para reduzir o custo total do projeto. Define-se o termo sobrecarga como sendo a relação da largura de banda agregada entre os hosts finais para a largura de banda total de uma topologia particular de comunicação. Uma sobrecarga de 1:1 por exemplo, indica que todos os hosts podem se comunicar com potencialmente outros hosts com sua largura de banda total.

2.3 Gerenciamento de energia em redes de Data Center

O interesse em soluções que conservam energia em sistemas de computação vem crescendo. Em ambientes corporativos e grandes Data Centers, onde os recursos de rede assumem dimensões significativas, reduzir o consumo de energia dissipada pelo meio de comunicação se torna uma opção interessante. Ações como a redução da taxa de transmissão de enlaces sub-utilizados ou seu desligamento completo (e de dispositivos de rede, da mesma forma) se tornam viáveis a partir do momento em que uma rede definida por software oferece uma visão global do estado da rede, que simplifica a identificação desses elementos ociosos e mesmo a redefinição de rotas a fim de desviar tráfego de elementos passíveis de desligamento.

Segundo Mahadevan et al. [44], algumas estratégias podem ser utilizadas para economizar energia em redes de Data Center, onde podemos destacar as seguintes:

- Power On/Off de *switches* e links: Tem como finalidade, ligar e desligar *links* de dados e equipamentos ociosos, reduzindo assim, o consumo de energia na rede do Data Center.

- Mapeamento do tráfego: Esta estratégia consiste em mapear os fluxos de tráfego em uma rede de Data Center para um conjunto de links e *switches*, de tal forma que os equipamentos de rede não utilizados sejam desligados para economizar energia.

- Adaptação da velocidade do link: Pode ser utilizada para reduzir a velocidade de uma porta com base na sua utilização de acordo com a carga de trabalho.

2.4 Redes definidas por *software*

As redes definidas por *software* (SDN – *Software Defined Networks*) [39], facilitam a criação de políticas de encaminhamento de pacotes através do uso de fluxos, permitindo o uso de caminhos com características específicas, como por exemplo: maior largura de banda, menor latência ou menor número de saltos. São caracterizadas pelo uso de um

controlador para programar o funcionamento da rede separando o plano de dados do plano de controle, tendo por objetivo facilitar inovações estruturais na rede.

Para a implementação das redes SDN, dois requisitos devem ser plenamente atendidos: (i) deve existir uma arquitetura comum em todos os equipamentos de rede envolvidos na comunicação (e.g., *switches* e roteadores) ou qualquer outro dispositivo gerenciado por um controlador SDN e, (ii) deve existir um protocolo padronizado seguro para a comunicação entre o controlador SDN e os dispositivos de rede.

Essas necessidades podem ser satisfeitas através da utilização da programação das redes que é uma característica da redes SDN. Uma rede programável é aquela em que o comportamento dos dispositivos de rede e de controle de fluxo são gerenciados por *software*, funcionando independentemente do *hardware* de rede. Essa programabilidade permite que um engenheiro de rede possa reprogramar uma infraestrutura de rede, ao invés de ter que reconstruí-la de forma manual.

A programabilidade de rede tem algumas vantagens sobre a rede tradicional [11]:

- Redução dos custos a longo prazo;
- Capacidade para aplicações manterem informações sobre as capacidades dos dispositivos;
- Capacidade para a rede responder as exigências de *status* dos aplicativos e recursos;
- Melhor alocação de largura de banda e recursos;
- Maior flexibilidade operacional e uma maior transparência;
- Suporte para as tecnologias de segurança e privacidade emergentes.

2.5 Protocolo OpenFlow

O protocolo *OpenFlow* [47] foi proposto em 2008 e atualmente está na versão 1.5.1 [9]. A arquitetura de uma rede que utiliza o protocolo *OpenFlow* é composta por *switches* e controladores. O objetivo do controlador é, através de um canal seguro, modificar a tabela de fluxos de um *switch OpenFlow*. Quando o controlador recebe uma mensagem de um *switch*, ele examina o cabeçalho desta mensagem e verifica se um novo fluxo precisa ser criado ou qual ação precisa ser realizada. Se uma nova entrada precisa ser criada, o controlador envia uma mensagem ao *switch* para instalar um novo fluxo. O controlador tem a capacidade de adicionar, atualizar e remover fluxos da tabela dos *switches OpenFlow*.

Em equipamentos de rede como roteadores e *switches* clássicos, o rápido encaminhamento de pacotes (plano de dados) e as decisões de encaminhamento (plano de controle) ocorrem em um mesmo dispositivo. Um *switch OpenFlow* tem a finalidade de separar

estas duas funções. A função de encaminhamento dos dados permanece sendo função do *switch*, enquanto que, as decisões de encaminhamento de alto nível são executadas agora por um controlador remoto, normalmente localizado em uma máquina servidora.

O controlador e o *switch OpenFlow* realizam a comunicação através do protocolo *OpenFlow*, que define as mensagens, tais como: pacotes recebidos, encaminhamento dos pacotes de saída, modificação da tabela de encaminhamento, etc.

A Figura 2.5 apresenta a arquitetura de rede usando o protocolo *OpenFlow*, e também, a separação do plano de dados e do plano de controle.

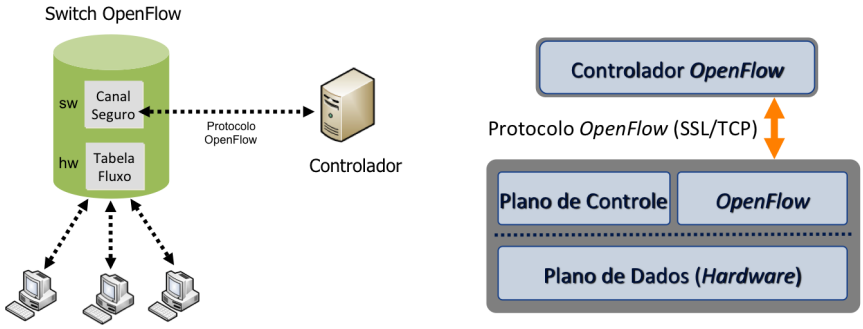


Figura 2.5 – Arquitetura de rede utilizando o protocolo OpenFlow. (Figura adaptada [47])

O *OpenFlow* é composto por três partes principais [47]:

- Tabela de Fluxos: A tabela de fluxos contém um conjunto de entradas comuns à ação associada. Cada entrada na tabela consiste em campos do cabeçalho (utilizados para definir um fluxo), ações (definem como os pacotes devem ser processados e para onde devem ser encaminhados) e contadores (utilizados para estatísticas ou remoção de fluxos inativos).

- Canal Seguro: Para que a rede não sofra ataques de elementos mal intencionados, o Canal Seguro garante confiabilidade na troca de informações entre o *switch* e o controlador. A interface de acesso ao tráfego utilizada é o protocolo *Secure Socket Layer* (SSL).

- Protocolo *OpenFlow*: O *switch OpenFlow* disponibiliza um protocolo aberto para estabelecer comunicação externa com o controlador. O controlador atua sobre as tabelas de fluxos dos switches através do canal seguro e, o protocolo *OpenFlow* é utilizado na comunicação entre o controlador e o *switch* tomando as decisões de acordo com as regras definidas para cada entrada na tabela de fluxos (ações).

O *OpenFlow* permite o controle dos fluxos de dados, escolhendo o caminho que cada pacote segue e o processamento que irá receber. Desta forma, o *OpenFlow* permite

experimentar novos protocolos de roteamento, mecanismos de segurança ou modelos de endereçamento na rede.

A Figura 2.6 exemplifica este controle de fluxo realizado através das regras que definem o fluxo, e que consiste em fazer o *match* dos campos do cabeçalho do pacote. As ações definem de que maneira os pacotes serão processados. Ainda, são geradas estatísticas responsáveis por registrar o número de pacotes e os bytes para cada fluxo existente, registrando o tempo que foi decorrido desde que o último pacote realizou *match*.

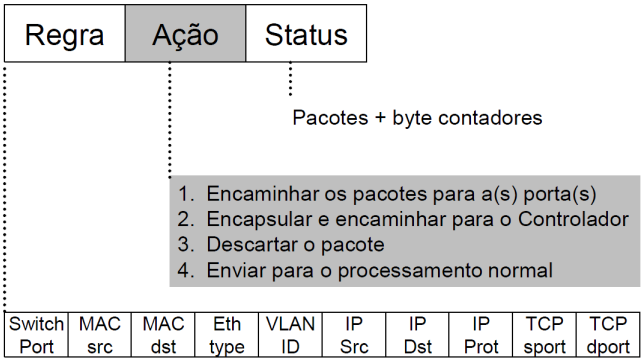


Figura 2.6 – Arquitetura de fluxos do protocolo OpenFlow. (Figura adaptada [47])

2.5.1 Controlador *OpenFlow*

O controlador é responsável por manter todas as regras de rede e distribuir as instruções adequadas para os dispositivos de rede. Ele determina como os fluxos sem correspondência válida serão tratados, gerenciando a tabela de fluxo principal, e adicionando e removendo entradas de fluxo através do canal seguro usando o protocolo *OpenFlow*. O controlador centraliza essencialmente a inteligência da rede, enquanto a rede mantém um plano de encaminhamento distribuído através dos *switches* e roteadores. Por esta razão, o controlador fornece uma interface para gerenciar, controlar e administrar as tabelas de fluxos dos *switches*. Normalmente, o controlador é executado em um servidor conectado à rede, existindo diferentes configurações possíveis de ser executadas [49]. A Tabela 2.1 apresenta os controladores mais utilizados atualmente.

Tabela 2.1 – Lista dos controladores *OpenFlow*

Controlador	Linguagem	Sistema Operacional	Característica
NOX[29]	C++, Python	Linux	Controlador de referência OpenFlow
POX[10]	C++, Python	Windows, Mac, Linux	Evolução do NOX
Maestro[5]	Java	Windows, Mac, Linux	Explora o paralelismo
Beacon[2]	Java	Windows, Mac, Linux	Multiplataforma, Multithreaded
Floodlight[4]	Java	Windows, Mac, Linux	Integração com redes não OpenFlow
Frenetic[27]	Funcional	Linux	Programa a rede como um todo
Onix[37]	C++, Python, Java	Windows, Mac, Linux	Gerência redes de grande porte
Trema[53]	C, Ruby	Linux	Script; Emulador
OpenDayLight[8]	Java	Linux	Controlador de referência da indústria

3. TRABALHOS RELACIONADOS

Este capítulo apresenta uma comparação entre os trabalhos relacionados na área de consumo de energia em redes de Data Centers.

O consumo de energia nos Data Centers se tornou um componente chave para o crescimento sustentável de alguns paradigmas, como por exemplo computação em nuvem. Como os servidores estão se tornando cada vez mais eficientes no quesito energia, o foco das preocupações gira em torno do aumento do consumo de energia da rede. Vários trabalhos tem abordado técnicas que visam através da programabilidade da rede, utilizar de forma eficiente a energia nas redes de Data Center.

Estudos recentes concentram-se na redução do consumo de energia em redes de Data Center, através do conceito de energia proporcional. Os principais trabalhos partem do pressuposto que o consumo de energia de uma rede de Data Center depende principalmente da capacidade do *switch* e da capacidade dos *links* que interligam estes equipamentos, onde é utilizado o modo de suspensão para indicar que um determinado componente da rede pode ser desligado quando não há tráfego através dele.

No trabalho ElasticTree [32] é proposto um otimizador de energia para a rede, com a finalidade de monitorar continuamente o tráfego de dados do Data Center, onde um conjunto de elementos de rede são selecionados, devendo permanecer ativos para garantir tolerância de desempenho e falha, onde são desativados os *links* desnecessários nos ativos de rede. Os autores demonstraram que os fluxos de tráfego em uma rede de Data Center pode ser consolidado em um pequeno conjunto de *links* e switches, que são suficientes para suprir a demanda de largura de banda. Os métodos analisados para decidir quais subconjuntos de *links* e *switches* podem ser desligados são: um modelo formal, heurística e métodos de previsão. Os autores estimam uma economia de energia em torno de 50%.

No trabalho *Energy Proportional Data Center Networks* [12] são propostas várias maneiras de projetar uma rede de Data Center de alto desempenho cujo consumo de energia é mais proporcional à quantidade de tráfego. O objetivo principal é fornecer uma rede que suporta a comunicação proporcional ao uso de energia. Ou seja, a quantidade de energia consumida é proporcional à intensidade de tráfego na rede. O trabalho demonstra que uma rede de Data Center com base na topologia *Flattened Butterfly* resulta em uma rede mais eficiente em termos de energia e, portanto, de baixa despesa operacional. São estimados periodicamente as necessidades de largura de banda futuras de cada link e reconfigurada de forma dinâmica sua taxa de dados para atender a essas exigências. Os autores apresentam resultados de redução do consumo global de energia de até 60%.

No trabalho ECODANE [34] é proposta a redução do consumo de energia na rede do Data Center com base na engenharia de tráfego, concentrando-se em otimizar o consumo de energia dos componentes de rede através da concepção de um sistema de con-

trole de rede inteligente. Este sistema se adapta dinamicamente ao conjunto de componentes de rede (ativos), onde são verificados padrões do tráfego total que passam pelo Data Center e com bases nesses padrões, são definidas ações a serem executadas por esses equipamentos. O algoritmo de otimização proposto pelos autores economizou nos testes realizados até 35% energia em uma rede com k=4.

Em CARPO [59] é proposto a construção de um sistema baseado na topologia ElasticTree [32] que possa ser flexível e adaptar a topologia da rede do Data Center ao consumo de energia que satisfaça as necessidades do tráfego. O objetivo é consolidar os fluxos de tráfego com base na análise da correlação entre os fluxos em uma rede de Data Center. Outra característica importante do trabalho é a correlação da consolidação do tráfego com a adaptação da taxa do link para o aumento da economia de energia. O trabalho faz uma relação dos resultados com o uso de programação linear para determinar a taxa de consolidação e de cada link de dados na rede do Data Center, propondo uma heurística para encontrar soluções de configuração e consolidação. Neste trabalho, os autores relataram resultados de economia de energia na rede do data center de até 46%.

No trabalho *PowerAware* [58] é proposta uma extensão para os *switches* que suportam o protocolo *OpenFlow* para suportar diferentes modos de economia de energia. A extensão inclui novas mensagens na pilha do protocolo *OpenFlow* e novas rotinas integradas no controlador que adicionam a função de gerenciar os *switches OpenFlow* através da capacidade de ligar / desligar e ativar / desativar as portas dos equipamentos. O trabalho implementa as funções do controlador integrado com um *switch OpenFlow* baseado na tecnologia NetFPGA [1], utilizando o *framework* citado anteriormente no trabalho ECODANE[34].

Em *HERO: Hierarchical Energy Optimization for Data Center Networks* [60] os autores propõem um modelo hierárquico de otimização de energia, que tem como finalidade a redução dos da energia consumida na rede do Data Center. A partir de uma topologia Fattree, os autores avaliam a possibilidade de desligar e ligar alguns *switches* e links de forma hierárquica, sem violar as restrições de conectividade e de QoS. Eles projetam algumas heurísticas baseadas em diferentes critérios de eliminação de *switches* para resolver o problema da otimização hierárquica.

No trabalho *Path consolidation for dynamic right-sizing*[13], os autores propõem um algoritmo *on-line* para dinamicamente fazer o dimensionamento correto da rede do Data Center para ter proporcionalidade de energia. O algoritmo proposto controla o nível de redundância visando garantir robustez, bem como a eficiência energética. O papel do otimizador em cada nível é encontrar um subconjunto mínimo de rede para atender desempenho e tolerância a falhas com objetivo de desligar os *switches* e links desnecessários. Neste trabalho, os autores relatam a redução de 80% no consumo de energia da rede.

No trabalho *Legacy Network Infrastructure Management Model for Green Cloud Validated Through Simulations* [57], os autores propõem reduzir o consumo de energia dos

[1] http://netfpga.org

Data Centers mediante a aplicação de um modelo de gerenciamento dos equipamentos de rede baseado em regras que tornam o consumo proporcional ao tráfego. Eles realizaram a extensão do simulador CloudSim [20] para que fosse possível a simulação do consumo de energia na rede. A partir de uma topologia tradicional de rede de Data Center, com base no modelo de gerenciamento proposto, eles avaliaram a aplicação das regras de desligar e ligar alguns *switches* e links. O modelo leva em consideração ainda a alocação de máquinas virtuais e algumas estratégias de migração para consolidação dos servidores. Nos experimentos demonstrados pelos autores, a economia de energia depende da configuração do data centers, do tipo dos equipamentos de rede e das cargas de trabalho, sendo relatada um economia de cerca de 50% no consumo de rede e 5% no consumo total do data center.

No trabalho *Energy-aware routing based on power profile of devices in data center networks using SDN* [48], os autores propõem um novo esquema de economia de consumo que pode controlar de forma flexível e rotear o tráfego baseando-se em perfis de energia nos dispositivos de rede. Eles utilizaram o protocolo *OpenFlow* em conjunto com placas NetFPGA baseadas em SDN *Software Defined Networks* para implementar a rede do Data Center. Os perfis de consumo de energia da rede foram definidos com base nas informações de cada dispositivo. Segundos os autores, a proposta do uso de perfis de energia na rede para os processos de otimização das topologia e para o roteamento dos fluxos, torna flexível trabalhar com dispositivos heterogêneos de diferentes fornecedores.

A Tabela 3.1 apresenta as características mais relevantes das soluções da literatura cujo foco se encontra na redução do consumo de energia em redes de Data Center. Da análise dessas soluções, identificamos a ausência de uma solução que utilizasse a combinação das estratégias de redução do consumo de energia em conjunto com diferentes fatores de *oversubscription*.

A coluna ”Estratégia de Redução do Consumo de Energia” indica quais estratégias a solução proposta utilizou para implementar a eficiência energética. A coluna ”Modelo” indica quais modelos são usados pelo trabalho para analisar os comportamentos da rede (matemático, heurística, etc.). A coluna ”Testbed” indica qual plataforma para a realização dos testes o trabalho em questão utilizou. Finalmente, a coluna ”Topologia” indica qual a topologia de rede de Data Center a solução proposta utilizou no seu desenvolvimento.

Tabela 3.1 – Trabalhos Relacionados

Trabalho	Estratégia de Redução do Consumo de Energia					Modelo	Testbed	Topologia
	Consolidação	On/Off Portas	On/Off Switches	Adaptação Link	Sobrecarga			
Elastic Tree[32]	Sim	Sim	Sim	Não	Não	Matemático Heurística Bin-Packing	Híbrido (Físico / virtual)	Fattree Modificada
Energy Proportional Network[12]	Não	Sim	Sim	Não	Não	Heurística Encaminhamento Adaptativo	N/A	Flattened Butterfly
ECODANE[34]	Não	Sim	Sim	Não	Não	Heurística Bin-Packing	Híbrido (Mininet / NetFPGA)	Elastictree
CARPO[59]	Sim	Sim	Sim	Sim	Não	Heurística Bin-Packing	Híbrido (Físico Virtual)	Fattree
PowerAware[58]	Não	Sim	Sim	Não	Não	Heurística	NetFPGA	N/A
HERO[60]	Não	Sim	Sim	Não	Não	Heurística	NetFPGA	Elastictree
Path Consolidation[13]	Sim	Sim	Sim	Não	Não	Heurística	Virtual	Fattree
Legacy Network Infrastructure Management[57]	Não	Sim	Sim	Sim	Não	Matemático Heurística	Simulador CloudSim)	Convencional
Energy-aware routing[48]	Sim	Sim	Sim	Sim	Não	Heurística	NetFPGA	Fattree

4. ESTRATÉGIAS DE MAPEAMENTO DE FLUXOS

Este capítulo apresenta a motivação, descrição do problema, objetivos do trabalho e as estratégias utilizadas no mapeamento dos fluxos gerados no processo de comunicação na rede do Data Center, com a finalidade de prover um caminho eficiente para economizar energia. A heurística utilizada foi baseada na combinação do algoritmo de Dijkstra (*Shortest Paths*) [25] com o problema *Bin-packing* [22], utilizando os algoritmos First-Fit, Best-Fit e Worst-Fit.

4.1 Motivação

De acordo com Koomey [36], de 2005 a 2010, a eletricidade consumida por Data Centers no mundo aumentou cerca 56% em vez de dobrar como ocorreu entre 2000 e 2005, enquanto que nos Data Centers dos EUA esse aumento foi de 36%. O relatório também indica que em 2010 a eletricidade usada por Data Centers em todo o mundo foi de cerca de 1,3% do uso total de energia elétrica do mundo, e de cerca de 2% do uso total de eletricidade nos EUA. A energia consumida pelos Data Centers ainda continua a ser substancial e é importante para provedores de serviços de Data Center minimizar o consumo de energia elétrica visando a proteção do ambiente, bem como a redução de custos operacionais.

Servidores consomem energia não proporcional, isso significa que estes equipamentos não controlam seu consumo de energia de acordo com a carga de trabalho. Assim, a fim de melhorar a eficiência energética de um Data Center, os pesquisadores têm concentrado seus esforços na otimização de servidores e sistemas de resfriamento, que correspondem a cerca de 70% do orçamento total de energia de um Data Center [39].

Nos últimos anos, houveram diversos esforços em métodos para melhorar a proporcionalidade de energia de servidores. Alguns artigos introduziram novas políticas para a distribuição de tarefas ou migração de máquinas virtuais, que podem minimizar o número de servidores ativos. Alguns exemplos de trabalhos incluem [28], [43] e [54]. Há também trabalhos focados em sistemas de refrigeração, tais como [40]. No entanto, existem poucas pesquisas focadas na economia de energia para a rede de Data Centers. Isso é em parte porque em comparação com os servidores, a rede consome menos energia, apenas entre 10% e 20% do total de energia do Data Center [39].

4.2 Descrição do problema

Apesar de uma rede de Data Center com uma topologia Fat-tree consumir apenas 12% da energia total quando um Data Center está em plena utilização, se os servidores são proporcionais em termos energéticos, o Data Center está com apenas 15% de utilização, a rede irá consumir quase 50% da potência geral [12], como pode ser visto na Figura 4.1.

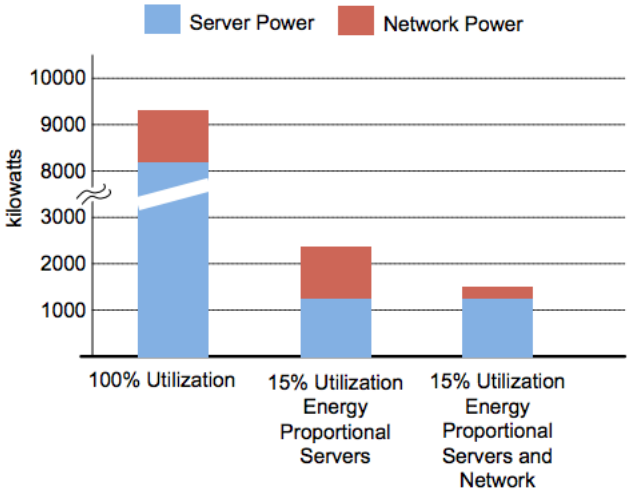


Figura 4.1 – Comparação da energia de servidores e rede [12].

Como os servidores são cada vez mais proporcionais em termos energéticos, a rede de Data Center vai consumir uma parcela maior da energia global. Portanto, a pesquisa por técnicas para economia de energia utilizada por parte dos dispositivos de rede torna-se fundamental. Assim como é realizado com os servidores minimizando o número de servidores ativos, também podemos encontrar formas de minimizar o número de dispositivos de rede ativos (*switches* e links).

4.3 Objetivos

Neste trabalho, são propostas estratégias para configuração da rede de Data Center visando reduzir o consumo de energia, utilizando o paradigma de SDN. Essas estratégias de mapeamento tem como finalidade: (i) consolidar os fluxos com base no modelo de consumo de energia proposto, (ii) realizar a adaptação da velocidade do link para economizar energia e (iii) dinamicamente desligar elementos de redes (*switches* e links) que não estão sendo utilizados.

Os objetivos específicos são os seguintes:

- Análise de diferentes formas de mapeamento dos fluxos.

- Análise da influência da utilização de diferentes fatores de sobrecarga no aumento da economia de energia na rede do Data Center

- Implementação e avaliação das estratégias propostas através do desenvolvimento de um simulador em Python [51] para automaticamente gerar os cenários de Data Center utilizados nos experimentos, implementando os algoritmos das estratégias de mapeamento de fluxos e a realização do cálculo do consumo de energia.

4.4 Descrição geral

A consolidação de tráfego nos fornece uma maneira eficiente para economizar energia na rede do Data Center. Os algoritmos de mapeamento propostos são baseados no *Bin-packing Problem* [22] em combinação com o algoritmo de Dijkstra (*Shortest Paths*). Foram utilizadas as heurísticas Firt-Fit, BestFit e WorstFit, sendo proposta a combinação das estratégias de redução do consumo de energia, desligando links e *switches* que não tem tiveram fluxos mapeados, a adaptação da velocidade dos links e ainda, o impacto do uso de diferentes fatores de sobrecarga, de modo que o consumo de energia na rede do Data Center possa ser reduzida.

4.5 Equação de consumo de energia

Um *switch* de rede é comumente composto de chassis, *line-cards* e portas. O chassi do *switch* inclui equipamentos de refrigeração, como ventiladores, que geralmente consomem uma quantidade fixa de energia. As *line-cards* fazem o *buffering* de todos os pacotes de entrada e de saída. As portas contêm o circuito de rede, que consome uma quantidade diferente de energia em velocidades diferentes [59].

De acordo com as medições feitas em [45] e [59], o consumo de energia ociosa de um *switch* de 48 portas varia de 70W a 150W. Cerca de 40 watts ou mais de consumo de energia pode ser adicionado, se o *switch* estiver trabalhando sob a sua capacidade máxima. Estas medições indicam que cada porta do *switch* só consome 1-2 Watts, onde dependendo da velocidade da porta, o consumo também pode variar, sendo 1W a 1 Gbps, 0,3W a 100 Mbps e 0,15 W a 10 Mbps, enquanto o *chassis*, a estrutura de comutação e as *line-cards* consomem a maior parte da energia. Portanto, em comparação com as estratégias que só redefinem a taxa de transmissão de cada porta, a economia de energia mais significativa pode ser alcançada desligando os *switches* desnecessários.

Por outro lado, Benson et al. estudaram as características de utilização dos links em um Data Center [17] e constataram que os Data Centers de produção reais tem uma

utilização média dos links na camada de agregação de cerca de 8% durante 95% do tempo, enquanto a utilização média dos links na camada de borda e os links na camada de núcleo são de aproximadamente 20% e 40%, respectivamente. O aproveitamento dos *links* com baixa taxa de utilização fornece um grande espaço para consolidar os fluxos de dados em *links* diferentes, de tal forma que, um número menor de links e *switches* são necessários para suportar o tráfego de dados existente.

A partir dos resultados apresentados em [45] e [59], foi construída a equação de consumo de energia de um *switch*:

$$\text{Consumo}_{switch} = \textit{Consumo}_{chassi} * \sum_{i=1}^{Num_{portas}} \text{Consumo}_{porta(i)} \quad (1)$$

Onde:

- $\textit{Consumo}_{switch}$: total de energia consumida por um *switch*;
- $\textit{Consumo}_{chassi}$: energia consumida pelo chassi do *switch*, com base no tipo do *chassi*, que pode ser Edge, Aggregation ou Core;
- $\textit{Num}_{portas}$: quantidade de portas ativas no *switch*;
- $\textit{Consumo}_{porta(i)}$: consumo que cada tipo de porta vai ter com base na sua velocidade, por exemplo 10/100/1000, ou se estiver desligado.

A equação consiste em calcular o consumo total de energia de cada *switch* dentro da topologia de rede do Data Center através da verificação dos principais parâmetros que influenciam no consumo. São levados em consideração na equação o consumo de energia do chassi do *switch*, o número de portas em uso em cada *line-card* e a velocidade de cada porta que pode ser de 10 Mbps, 100 Mbps ou 1 Gbps.

Exemplificando a equação de consumo de energia definido, considerando-se uma topologia linear, com 10 *switches*, onde cada *switch* tem 1 servidor conectado a 100 Mbps e entre os *switches*, existe um *link* de 1 Gbps. Com base nos parâmetros definidos, sabemos que o *chassi* de cada *switch* consome em média 60 W, cada porta de 1 Gbps tem um consumo de 1W e cada porta de 100 Mbps tem um consumo médio de 0,3 W. Assim, podemos verificar que o consumo da topologia exibida na Figura 4.2, baseada na equação é de 621 Watts, pois existem 18 portas com velocidade de 1 Gbps interconectando os *switches* e 10 portas com velocidade de 100 Mbps conectando os servidores aos 10 *switches*.

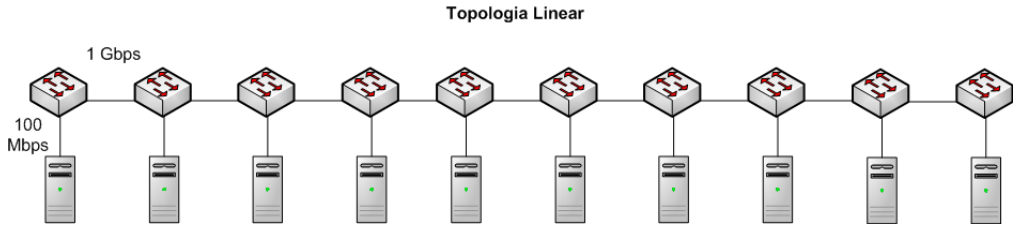


Figura 4.2 – Topologia com exemplo da equação proposta de consumo de energia

4.6 Estratégias de mapeamento de fluxo

O uso das heurísticas *Bin-packing* tem a finalidade de melhorar a escalabilidade da consolidação de fluxos na rede do Data Center. Soluções ótimas não são garantidas, mas na prática podem resultar em bons resultados. Para cada fluxo, os algoritmos avaliam os caminhos possíveis e escolhem o que tiver capacidade suficiente. Quando todos os fluxos forem atribuídos (o que é não garantido), os algoritmos retornam o conjuntos de ativos de rede (*switches* e links que os fluxos passaram) além de cada caminho utilizado.

Os caminhos em uma rede podem ser facilmente representados como um grafo (G), isto é, um conjunto de nodos (vértices) V e arestas E, onde cada aresta conecta dois nodos $G = (V, E)$. Para cada aresta é atribuído um peso (*weight*). Por exemplo, o comprimento da estrada ou uma estimativa do tempo necessário para viajar ao longo da estrada. Em teoria dos grafos, o cálculo dos caminhos mais curtos (*Shortest Paths*) entre dois nós é um problema clássico, dentre as variações deste problema, utilizamos neste trabalho a que calcula o comprimento do caminho mais curto a partir de um determinado nó de origem $s \in V$ para um determinado nó de destino $t \in V$ [21].

Dijkstra é um algoritmo iterativo que nos fornece o caminho mais curto (*Shortest Paths*) de uma determinada origem até um destino ou todos os destinos dentro do grafo. Com base neste mapeamento, foi modificado o parâmetro *weight* no algoritmo de Dijkstra, para atender as premissas das estratégias. As estratégias utilizadas foram chamadas de S1, S2 e S3.

As três estratégias realizam o mapeamento dos fluxos na rede do Data Center com base no peso definido para cada aresta do grafo. S1 vai encontrar o primeiro caminho (bin) que se encaixa com a demanda de fluxo, buscando assim, o caminho que passa por um número menor de *switches* para mapear o fluxo. S2 avalia todos os caminhos disponíveis e mapeia o fluxo com base no link que forneça a menor capacidade de largura de banda, porém essa capacidade tem de ser suficiente para atender a demanda do fluxo. S3 por sua vez é análoga à S2, avaliando todos os caminho disponíveis, porém mapeia o fluxo com base no caminho que sobrar mais capacidade.

As estratégias utilizam abordagens iniciais análogas no processo de mapeamento dos fluxos. Inicialmente, após a criação dos fluxos, todas as capacidades dos links são

acrescidas dos fatores de sobrecarga 1:1, 1:5 e 1:20. Após isso, para cada par (u, v) de origem e destino, foi feito o cálculo do conjunto de caminhos que suportam a demanda de fluxo total (largura de banda) entre origem e destino (u, v), se a capacidade do link for menor que a largura de banda, então os arestas são removidos da topologia. O processo de seleção de caminho continua até que os requisitos de fluxo para todas as origens e destinos sejam atendidos.

4.7 Estratégia S1

O modelo criado para a S1 tem a premissa de que todos os links possuem peso igual a 1, logo, o algoritmo de Dijkstra irá buscar o caminho que passa por um número menor de *switches*. S1 realiza o mapeamento dos fluxos utilizando *shortest paths* nas arestas com capacidade suficiente, definindo o parâmetro *weight* = 1, ou seja, pesos iguais para todos os links em todas as extremidades dos vértices do grafo. O Algoritmo 4.2 apresenta S1.

Nas linhas 2 e 3 do Algoritmo 4.1 é possível verificar a aplicação dos fatores de sobrecarga definidos, que são utilizados em S1, S2 e S3.

Algoritmo 4.1 Algoritmo de sobrecarga

```
1: procedure APPLYOVERLOAD(topology, overloadfactor)
2:     for all link ∈ topology.links do
3:         link.capacity ← link.capacity * overloadfactor
4:     end for
5: end procedure
```

Antes da realização do mapeamento dos fluxos, o Algoritmo 4.2 realiza o processo de retirada de todos os links que não tem capacidade para atender a demanda, como pode ser visualizado nas linhas 3, 4, 5 e 6 do Algoritmo 4.2, onde é feita a cópia da topologia inicial e os links tem sua capacidade de atender a demanda do fluxo verificada. Após o processo anterior de retirada dos links que não atendem a demanda do fluxo, o algoritmo de Dijkstra recebe a topologia e aplica o peso como sendo *weigth = 1*, conforme é demostrado na linha 10. Após a plicação do peso, é realizada a remoção da capacidade utilizada pelo fluxo dos links indicados no caminho encontrado, como pode ser visualizado nas linas 14, 15 e 16.

A Figura 4.3 apresenta um exemplo de funcionamento do algoritmo com o mapeamento dos fluxos definidos na Tabela 4.1 em uma topologia fat-tree com $k = 4$. Os links que fazem a ligação da camada de núcleo com a camada de agregação são de 1Gb. Os links que fazem a ligação da camada de agregação com a camada de borda são de 100Mb e os links que fazem a ligação da camada de borda com os hosts tem largura de banda de 10Mb.

Algoritmo 4.2 Algoritmo da estratégia S1

```
1: procedure ESTRATÉGIAS1(topology, flows)
2:     for all flow ∈ flows do

3:         prunedtopology ← copy(topology)
4:         for all link ∈ topology.links do
5:             if link.capacity < flow.bw then
6:                 prunedtopology.removelink(link)
7:             end if
8:         end for

9:         for all link ∈ prunedtopology.links do
10:            link.weight ← 1
11:        end for

12:        path ← DijkstraShortestPath(prunedtopology, flow)
13:        if path ≠ NULL then
14:            for all link ∈ path.links do
15:                tlink ← topology.getlink(link)
16:                tlink.capacity ← tlink.capacity - flow.bw
17:            end for
18:        end if
19:    end for
20: end procedure
```

Tabela 4.1 – Exemplo do mapeamento de fluxos utilizando S1

Fluxo	Origem / Destino	Demanda	Caminho
f1	H20 → *H*24	1Mb	H20 → *S*6 → *S*4 → *S*0 → *S*8 → *S*10 → *H*24
f2	H20 → *H*27	2Mb	H20 → *S*6 → *S*4 → *S*0 → *S*8 → *S*11 → *H*27
f3	H20 → *H*23	3Mb	H20 → *S*6 → *S*4 → *S*7 → *H*23

A estratégia S1 encontra o primeiro caminho que se encaixa com a demanda de fluxo. Na Tabela 4.2, podemos verificar que os fluxos f1, f2 e f3 foram mapeados no caminho mais curto entre a origem e o destino de cada fluxo, buscando assim, o caminho que passa pelo menor número de *switches*.

4.8 Estratégia S2

Na estratégia S2, o modelo tem a premissa de que os links possuem como peso a capacidade do link. Portanto, o algoritmo de Dijkstra irá buscar o caminho com menor capacidade utilizada. O mapeamento dos fluxos com S2 tenta encontrar um link que forneça a menor capacidade de largura de banda, porém essa capacidade tem de ser suficiente para atender a demanda. O Algoritmo 4.3 apresenta o funcionamento da estratégia S2.

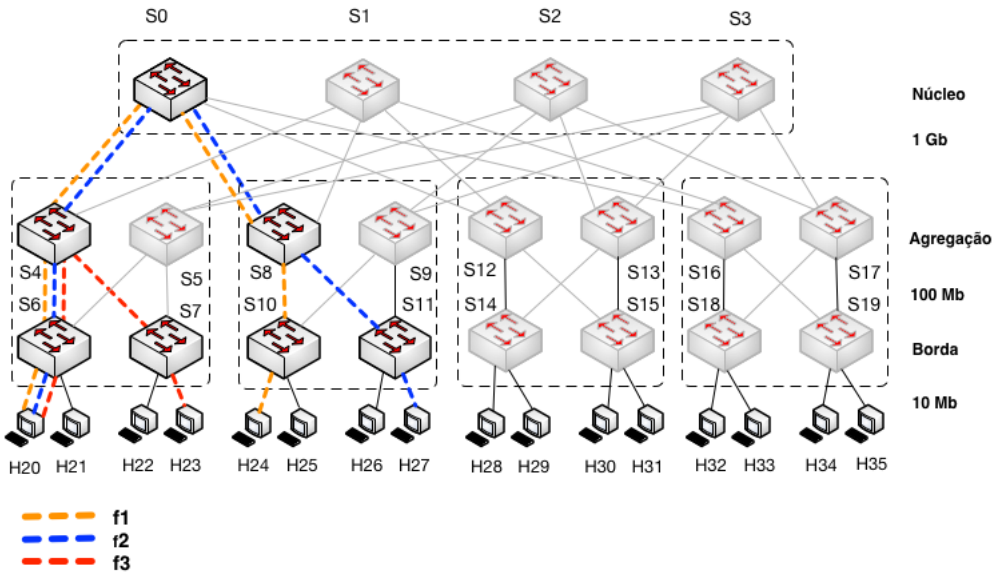


Figura 4.3 – Mapeamento de fluxos utilizando S1.

No Algoritmo 4.3 podemos visualizar o processo semelhante ao do Algoritmo 4.2, onde nas linhas 3, 4, 5 e 6 é feita a cópia da topologia inicial e os links tem sua capacidade de atender a demanda do fluxo verificada. Após o processo de retirada dos links que não atendem a demanda do fluxo, o algoritmo Dijkstra recebe a topologia com os links que não atendem a demanda retirados e aplica o peso como sendo *weigth = capacity*, conforme é demostrado na linha 10. Analogamente ao que ocorre em S1, após a plicação do peso é realizada a remoção da capacidade utilizada pelo fluxo dos links indicados no caminho encontrado, como pode ser visualizado nas linas 14, 15 e 16.

A Figura 4.4 exemplifica o funcionamento da estratégia S2 com o mapeamento dos fluxos da Tabela 4.2 em uma topologia fat-tree com $k = 4$. Os links que fazem a ligação da camada de núcleo com a camada de agregação são de 1Gb, já os links que fazem a ligação da camada de agregação com a camada de borda são de 100Mb e os links que fazem a ligação da camada de borda com os hosts tem largura de banda de 10Mb.

Tabela 4.2 – Exemplo do mapeamento de fluxos utilizando S2

Fluxo	Origem / Destino	Demanda	Caminho
f1	H20 → *H24*	1Mb	H20 → *S6* → *S4* → *S0* → *S8* → *S10* → *H24*
f2	H20 → *H25*	2Mb	H20 → *S6* → *S4* → *S0* → *S8* → *S10* → *H25*
f3	H20 → *H23*	3Mb	H20 → *S6* → *S4* → *S7* → *H23*

A estratégia S2 mapeia o fluxo no link que tenha a menor capacidade de largura de banda restante, desta forma, vai sempre preferir os links que já tenham fluxos mapeados pois a medida que os fluxo passam por um determinado link, a largura de banda deste link vai sendo decrementada. Na Tabela 4.2, podemos verificar que no f3 por exemplo, o caminho escolhido passou por S6 - S4 - S7, onde S4 já tinha fluxos mapeados anteriormente nos fluxos f1 e f2 e não por S5 que ainda não tinha fluxos mapeados.

Algoritmo 4.3 Algoritmo da estratégia S2

```
1:  procedure ESTRATÉGIAS2(topology, flows)
2:      for all flow ∈ flows do

3:          prunedtopology ← copy(topology)
4:          for all link ∈ topology.links do
5:              if link.capacity < flow.bw then
6:                  prunedtopology.removelink(link)
7:              end if
8:          end for

9:          for all link ∈ prunedtopology.links do
10:             link.weight ← link.capacity
11:         end for

12:         path ← DijkstraShortestPath(prunedtopology, flow)
13:         if path ≠ NULL then
14:             for all link ∈ path.links do
15:                 tlink ← topology.getlink(link)
16:                 tlink.capacity ← tlink.capacity - flow.bw
17:             end for
18:         end if
19:     end for
20: end procedure
```

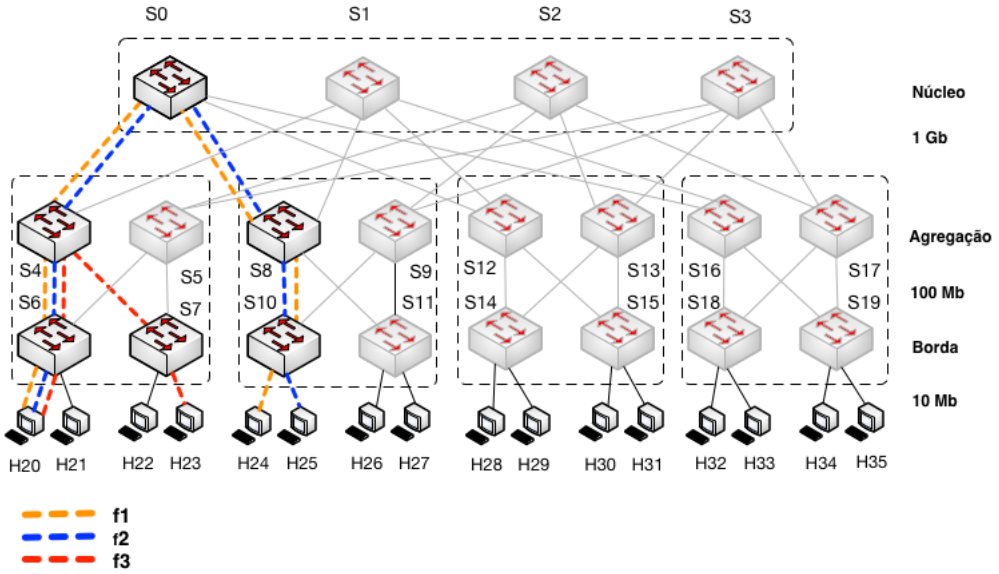


Figura 4.4 – Mapeamento de fluxos utilizando S2

4.9 Estratégia S3

Na estratégia S3, o modelo desenvolvido tem a premissa de que os links possuem como peso o inverso da capacidade dos links. Logo, o algoritmo de Dijkstra irá buscar

o caminho que sobra mais capacidade restante nos links. Nesta estratégia, o parâmetro *weigth* é utilizado como peso das arestas pelo algoritmo *Shortest path*, o peso atribuído a cada link é proporcional ao inverso da sua capacidade, $\frac{1}{1000}$, $\frac{1}{100}$ e $\frac{1}{10}$. O Algoritmo 4.4, apresenta a estratégia S3.

Algoritmo 4.4 Algoritmo da estratégia S3

```
 1: procedure ESTRATÉGIAS3(topology, flows)
 2:     for all flow ∈ flows do

 3:         prunedtopology ← copy(topology)
 4:         for all link ∈ topology.links do
 5:             if link.capacity < flow.bw then
 6:                 prunedtopology.removelink(link)
 7:             end if
 8:         end for

 9:         for all link ∈ prunedtopology.links do
10:             link.weight ← 1/link.capacity
11:         end for

12:         path ← DijkstraShortestPath(prunedtopology, flow)
13:         if path ≠ NULL then
14:             for all link ∈ path.links do
15:                 tlink ← topology.getlink(link)
16:                 tlink.capacity ← tlink.capacity - flow.bw
17:             end for
18:         end if
19:     end for
20: end procedure
```

No Algoritmo 4.4 o processo é semelhante ao já apresentado em S1 e S2, onde após a remoção dos links que não tem capacidade de atender a demanda do tráfego, o algoritmo de Dijkstra recebe a topologia e aplica o peso de como sendo *weigth = inverse_capacity*, onde o valor de *inverse_capacity* é igual ao inverso das capacidades atribuídas a cada link, $\frac{1}{1000}$, $\frac{1}{100}$ e $\frac{1}{100}$ como demonstrado na linha 10. Após a plicação do peso é realizada a remoção da capacidade utilizada pelo fluxo dos links indicados no caminho encontrado, como pode ser visualizado nas linas 14, 15 e 16.

A Figura 4.5 apresenta um exemplo análogo a S2. Os fluxos utilizados são os mesmos da Tabela 4.2, porém, desta vez os caminhos por onde os fluxos são mapeados são diferentes.

A estratégia S3 mapeia os fluxos nos links que tenham mais capacidade restante nos links. Por este motivo, vai sempre preferir os links que não tenham fluxos mapeados ou que tenham maior capacidade de banda restante. Na Tabela 4.3, podemos verificar que ao contrário do que ocorre na estratégia S2, no fluxo f3 o caminho escolhido passou por S6 -

Tabela 4.3 – Exemplo do mapeamento de fluxos utilizando S3

Fluxo	Origem / Destino	Demanda	Caminho
f1	H20 → *H24*	1Mb	H20 → *S6* → *S4* → *S0* → *S8* → *S10* → *H24*
f2	H20 → *H25*	2Mb	H20 → *S6* → *S5* → *S2* → *S9* → *S10* → *H25*
f3	H20 → *H23*	3Mb	H20 → *S6* → *S5* → *S7* → *H23*

S5 - S7, pois ao contrário de S4, S5 não tinha nenhum fluxo mapeado e estava com 100% da capacidade do link.

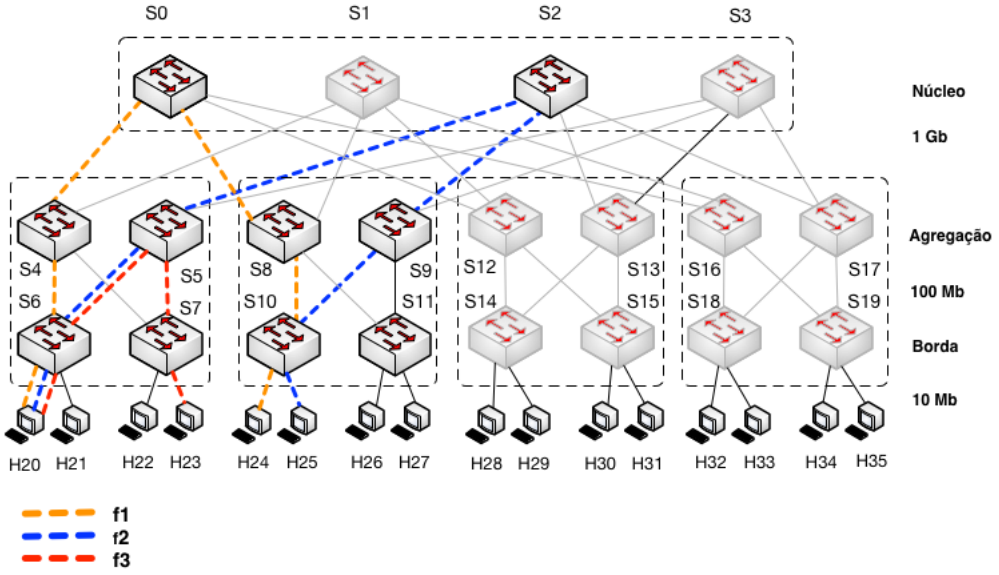


Figura 4.5 – Mapeamento de fluxos utilizando S3

4.10 Estratégias de redução do consumo de energia e redes de Data Center

Com base nas pesquisas relacionadas ao consumo de energia em redes de Data Center realizadas por Mahadevan et al. [44], foram definidos um conjunto de três estratégias principais (power on/off de *switches* e links, mapeamento do tráfego e adaptação da velocidade do link), que foram exploradas para o desenvolvimento do algoritmo responsável pela economia de energia.

No Algoritmo 4.5 é possível visualizar os passos realizados para a aplicação das estratégias de redução do consumo de energia. Nas linhas 2, 3 e 4 podemos visualizar a verificação dos links que não tiveram fluxos mapeados e como consequência, seu desligamento. Entre as linhas 7 e 15, são verificados se os switches tem links ativos, se não existir links em uso, o *switch* é desligado. Nas linhas 18, 19 e 20 é realizada a redução da velocidade das portas que estão com baixa utilização.

Algoritmo 4.5 Algoritmo de aplicação das estratégias de redução de energia

```
 1: procedure MECANISMOSREDUÇÃOENERGIA(topology)
 2:     for all link ∈ topology.links do
 3:         if link.maxcapacity = link.capacity then
 4:             link.status ← off
 5:         end if
 6:     end for

 7:     for all switch ∈ topology.switches do
 8:         usedlinks ← 0
 9:         for all link ∈ switch.links do
10:             if link.status = on then
11:                 usedlinks ← usedlinks +1
12:             end if
13:         end for
14:         if usedlinks = 0 then
15:             switch.status ← off
16:         end if
17:     end for

18:     linkcapacities ← [10, 100, 1000]
19:     for all link ∈ topology.links do
20:         usedcapacity ← link.maxcapacity - link.capacity
21:         for all lcap ∈ linkcapacities do
22:             if usedcapacity <= lcap then
23:                 link.maxcapacity ← lcap
24:                 BREAK
25:             end if
26:         end for
27:     end for
28: end procedure
```

No Algoritmo 4.6 podemos visualizar os passos realizados para o cálculo da energia consumida pela topologia. Nas linhas 2, 3, 4 e 5 é verificado se os *switches* estão ligados e qual a sua camada que pode ser de núcleo, de agregação ou de borda. Após, a verificação é atribuído o valor de consumo com base nos valores definidos anteriormente. O mesmo acontece com as portas dos *switches* nas linhas 6, 7 e 8, que após terem seu *status* verificado, tem os valores de consumo referente a cada tipo de porta atribuído. Por fim, é retornado o consumo total da topologia na linha 13.

Algoritmo 4.6 Algoritmo de cálculo do consumo de energia

```
 1: procedure CALCULOCONSUMOENERGIA(topology)
 2:     energycost ← 0
 3:     for all switch ∈ topology.switches do
 4:         if switch.status = on then
 5:             energycost ← energycost + switch.cost
 6:             for all link ∈ switch.links do
 7:                 if link.status = on then
 8:                     energycost ← energycost + link.cost
 9:                 end if
10:             end for
11:         end if
12:     end for
13:     return energycost
14: end procedure
```

Após o desenvolvimento da equação de consumo de energia e dos algoritmos de mapeamento dos fluxos, o próximo passo foi avaliá-los para verificar a eficácia da redução do consumo de energia. O Capítulo 5 apresenta os ambientes utilizados nos experimentos e os resultados a partir de simulações com diferentes configurações de rede.

5. AVALIAÇÃO

Neste capítulo são apresentados diversos experimentos visando avaliar as estratégias propostas com diferentes configurações de topologia e carga de trabalho. Os experimentos foram conduzidos através de um simulador desenvolvimento especificamente para esse fim.

5.1 Simulação

Atualmente a simulação tem um papel decisivo no projeto, análise e implementação de sistemas de comunicação, principalmente quando estes sistemas são caros e complexos. A simulação de um sistema real pode ser definida como o processo de avaliação numérica de um modelo de simulação, que deve representar o mais fielmente possível o sistema real a ser simulado. As informações resultantes deste processo são utilizadas para estimar variáveis de interesse deste sistema.

As organizações dependem cada vez mais de novas tecnologias de rede e aplicações para apoiar as suas necessidades críticas de negócios. A fim de avaliar as várias soluções alternativas para um determinado objetivo de design, projetistas de rede cada vez mais dependem de métodos que ajudem a avaliar várias propostas de projeto antes da decisão final e da construção dos sistemas físicos. Um método amplamente aceito é a utilização de ambientes de simulação. Um modelo de simulação pode ser usado por um projetista de rede para analisar alternativas de design e estudar o comportamento de um novo sistema ou as modificações de um sistema existente sem fisicamente construí-lo. Um modelo de simulação pode também representar as tecnologias da rede e as funções executadas em uma rede, a fim de obter resultados estatísticos sobre ela [30].

Segundo Hughes [6], alguns dos principais motivos pelos quais é recomendável que os ambientes de redes sejam previamente simulados antes da sua instalação são:

- A experimentação no mundo real pode causar alguma espécie de dano ao ambiente;
- A modelagem/análise requer altos níveis de abstração e possíveis falhas na especificação de detalhes;
- Alterações em redes existentes podem requerer uma carga de trabalho expressivo: Falhas na configuração de determinadas tecnologias podem gerar um custo elevado.

5.2 Ambiente e parâmetros de simulação

Devido a dificuldade para implementar novos mecanismos de eficiência energética e observar seus efeitos em um Data Center de forma controlada e de modo repetitivo, pesquisadores costumam utilizar simulação para modelar o mecanismo e avaliar os resultados. Através da simulação de um Data Center, se evita gastos financeiros e tempo desnecessários para configurar um ambiente de testes.

Nesta pesquisa, foi desenvolvido um simulador em Python [51] para automaticamente gerar os cenários de Data Center utilizados nos experimentos, implementando os algoritmos das estratégias de mapeamento de fluxos e realizar o cálculo do consumo de energia com base no modelo de consumo proposto. Foi escolhida a topologia fat-tree com diferentes tamanhos de POD's *(Performance Optimized Data Centers)* para simular a rede do Data Center, sendo a escolha justificada tendo em vista que a topologia fat-tree é tipicamente utilizada em redes de Data Center, e pode fornecer caminhos paralelos de tamanhos iguais entre quaisquer hosts de origem e destino, sendo assim, muito benéfica para o paradigma do processamento de caminhos independentemente da topologia.

Uma k fat-tree é a arquitetura de rede de Data Center que usa de forma similar k ports *switches*. Existem k pod's, cada pod contém $k/2$ *switches* de agregação e $k/2$ *switches* de borda. O número de *switches* de núcleo é $k^2/4$ e cada *switch* de núcleo tem um link para cada pod. Nesta topologia, cada *switch* de borda está ligado a $k/2$ hosts. Uma fat-tree com $k = 4$ pod's tem $k^3/4$ hosts.

Para a construção da topologia fat-tree, foi utilizado o *Fast Network Simulation Setup* (FNSS) [50] que é um conjunto de ferramentas de rede que permite que pesquisadores simplifiquem o processo de criação de cenários para experimentos de rede. Ele permite analisar uma topologia de um conjunto de dados ou um gerador de topologia ou gerá-lo de acordo com uma série de modelos sintéticos, configurar as características dos links como capacidade, *weights*, *delay* e tamanhos de *buffers*, gerar matrizes de tráfego, implantação e configuração de topologias de rede e ainda, configuração de *workloads* para uma série de simuladores e emuladores, como por exemplo: ns-2 [7], ns-3 [33], Mininet [39], Omnet++ [55] e Autonetkit [35]. O FNSS utiliza para a construção do grafo o NetworkX [52], que por sua vez realiza a criação, manipulação e estudo da estrutura dinâmica e função de redes complexas.

5.3 *Workload*

Atualmente um dos maiores desafios para a realização de simulações de ambiente de redes de Data Center é como gerar tráfego da rede em produção, pois a maioria

dos *traces* de Data Centers são confidenciais e não são disponibilizados de forma pública. Devido ao tamanho da carga de tráfego, grandes quantidades de fluxos são gerados em uma rede de Data Center de tamanho razoável, por isso, não é prático capturar todos os fluxos da rede. Para garantir que nossos resultados sejam aplicáveis, as estratégias propostas foram submetidas aos padrões de tráfego *Random*, *Stride*(1) e *Stride*(*i*) apresentados anteriormente por Al-Fares et al. [14] e descritos a seguir:

- *Random*: Um host envia para qualquer outro host na rede com probabilidade uniforme. A origem e o destino são selecionados aleatoriamente. Este padrão de tráfego reproduz um processo médio em que os aplicativos são colocados aleatoriamente na rede do Data Center.

- *Stride*(1): O destino de um fluxo a partir do host x é o host [(x + i) *mod* (*numhosts*)], onde os hosts são numerados da esquerda para a direita como 0, 1, ⋯ , num hosts-1.

- *Stride*(*i*): É caracterizado pela comunicação inter-pod, forçando assim o uso da camada de núcleo. Sendo $i = (k/2)^2$, onde k é a quantidade de pod's utilizada nos experimentos ($k = 4, 8, 12$).

Por exemplo, em uma topologia fat-tree de tamanho $k = 4$, *Stride*(1) tem quase metade do tráfego passando entre os hosts conectados ao mesmo *switch* de borda e a outra metade vai para o tráfego de agregação e para os *switches* núcleo. Por outro lado, *Stride*(4) envia todo o tráfego entre pod's, resultando em um maior número de opções de participação no tráfego encaminhado.

A carga de tráfego foi definida como a razão entre a largura de banda ocupada média em links dos hosts dividido pela capacidade do link. Testamos 3 grupos com cargas de 20%, 50% e 80% que foram chamadas de baixa, média e alta respectivamente.

5.4 Experimentos

Uma das tarefas mais pesadas necessárias para executar uma simulação de rede é a configuração de um cenário completo, sua aplicação, e tipo de ambiente a ser simulado. Este processo inclui a seleção de uma topologia, parâmetros necessários e por fim, configuração de uma matriz de tráfego e a carga de trabalho. Nosso simulador modela a topologia do Data Center como um grafo de rede com arestas dirigidas.

As três estratégias propostas realizam o mapeamento dos fluxos na rede do Data Center com base no peso definido para cada aresta do grafo. S1 utiliza *weight* = 1 e vai encontrar o primeiro caminho que se encaixa com a demanda de fluxo, buscando assim, o caminho que passa por um número menor de *switches* para mapear o fluxo. S2 utiliza

o *weight* = *capacity* e avalia todos os caminhos disponíveis, mapeando o fluxo com base no link que forneça a menor capacidade de largura de banda. S3 é análoga à estratégias S2 porém utiliza *weight* = *inverse_capacity*, avaliando todos os caminho disponíveis e mapeando o fluxo com base no link que sobrar mais capacidade.

A verificação da economia de energia depende dos padrões de tráfego, da carga e da configuração de rede e do tamanho da rede do Data Center. Para os experimentos realizados, variamos esses parâmetros com a finalidade de verificar os níveis de consumo de energia em diferentes cenários de redes de Data Center.

Para cada uma das três estratégias propostas, foram avaliados diferentes parâmetros de rede, incluindo três diferentes tamanhos de pod's (K=4, K=8 e K=12), uma configurações de rede, onde a velocidades de portas de comunicação foi de 1Gb, 100Mb, 100Mb representam respectivamente, os links que fazem a ligação da camada de núcleo com a camada de agregação, a ligação da camada de agregação com a camada de borda e a ligação da camada de borda com os hosts. Foram definidos ainda três cargas de redes 20%, 50% e 80% e uma média de 10 fluxos por host. Foram realizados uma série de experimentos que avaliaram as estratégias de mapeamento de fluxo com os fatores 1:1, 1:5 e 1:20 de sobrecarga.

De acordo com Al-Fares et al. [14], diversos projetos de Data Center tem introduzido o sobrecarga como um meio para reduzir o custo total do projeto. Por exemplo, um fator de sobrecarga 1:5 significa que apenas 20% da largura de banda disponível no hosts esteja disponível para alguns padrões de comunicação. Conforme citado anteriormente, alguns estudos atuais revelam que diversas redes de Data Center tem usado o conceito de sobrecarga [16] e [41], com diversos fatores 1:5, 1:20 e até 1:40. Este último e mais alto é utilizado nos Data Centers do Facebook [26].

Na Tabela 5.1, são apresentados os diferentes parâmetros utilizados nas configurações dos experimentos.

Tabela 5.1 – Configuração dos experimentos realizados

Experimentos			
POD	k=4	k=8	k=12
Configuração de Rede	1000,100,100		
Workloads	Random	*Stride*(1)	*Stride*((*k*/2)^2)
Fatores de Sobrecarga	1:1	1:5	1:20
Carga de Rede	20%	50%	80%
Estratégias	S1	S2	S3

5.5 Resultados

Nesta Seção são apresentados e discutidos os resultados obtidos nos experimentos realizados com o simulador de redes de Data Center.

5.5.1 POD=4 - Configuração de rede [1000, 100, 100]

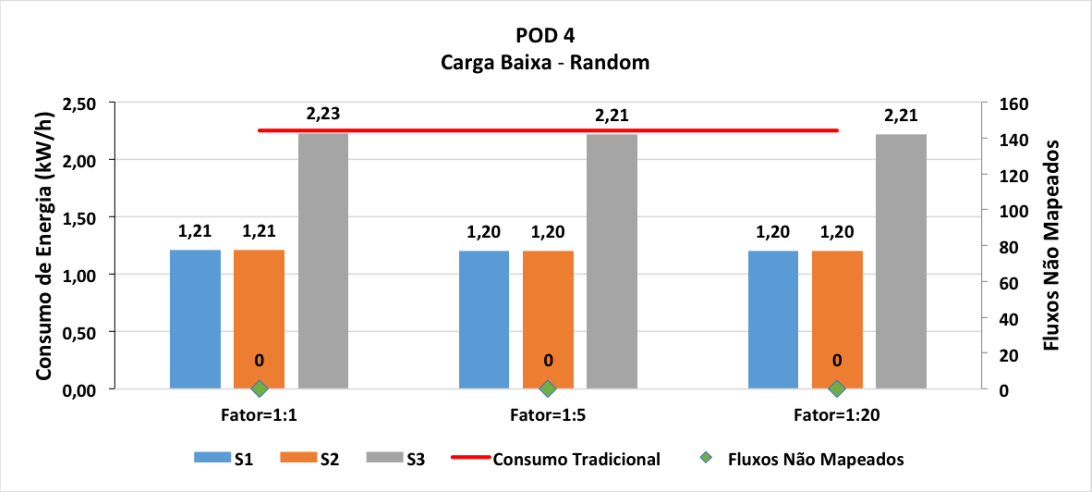
Os experimentos apresentados a seguir foram realizados com uma topologia fat-tree de tamanho $k = 4$, com a configuração de rede [1000, 100, 100]. Essa configuração representa respectivamente, os links que fazem a ligação da camada de núcleo com a camada de agregação, a ligação da camada de agregação com a camada de borda e a ligação da camada de borda com os hosts. A topologia fat-tree com $k = 4$ utilizada é composta por 4 *switches* de núcleo, 8 *switches* de agregação, 8 *switches* de borda e 16 hosts. A seguir, são descritos os experimentos e exibidas as considerações e os resultados organizados por tipo de *workload: Random, Stride*(1) *e Stride*(4):

5.5.1.1 Workload Random

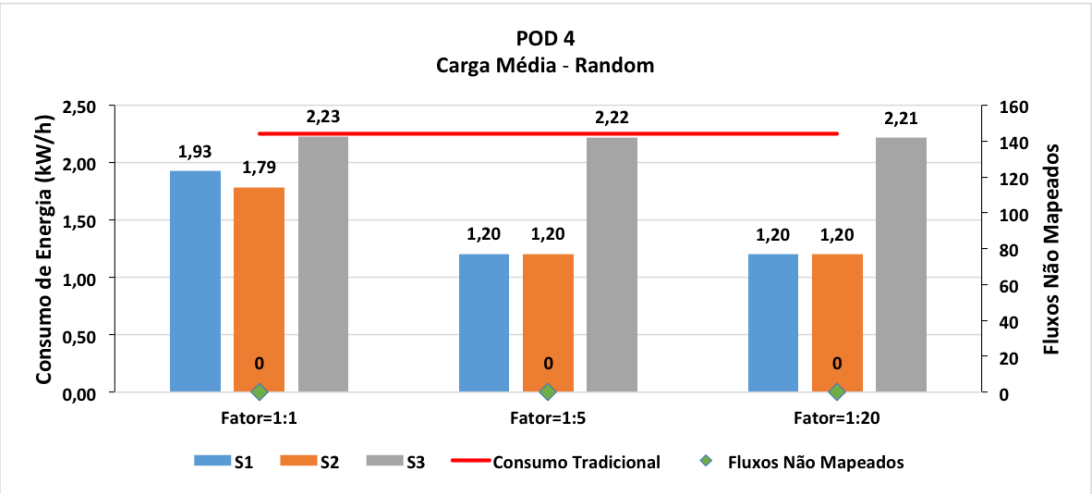
As Figuras 5.1a, 5.1b e 5.1c, apresentam os gráficos com o consumo de energia da rede para uma topologia de Data Center fat-tree com tamanho $k = 4$, com uma carga de rede baixa (20%), média (50%) e alta (80%) respectivamente, com o padrão de tráfego *Random* e uma média de 10 fluxos por host. O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

Com base na análise dos resultados apresentados nas Figuras 5.1a, 5.1b e 5.1c é possível constatar que a alteração da configuração da rede para [1000, 100, 100] teve influência no consumo de energia nas cargas de rede média e alta, mantendo o consumo inalterado na carga baixa como pode ser visto na Figura 5.1a. As Figuras 5.1b e 5.1c demonstram que o uso das estratégias S1 e S2 tiveram pouco impacto na redução do consumo de energia com o fator de sobrecarga 1:1, demonstrando que a medida que carga de rede foi aumentada, foi necessário manter mais equipamentos ligados para atender aos fluxos gerados.

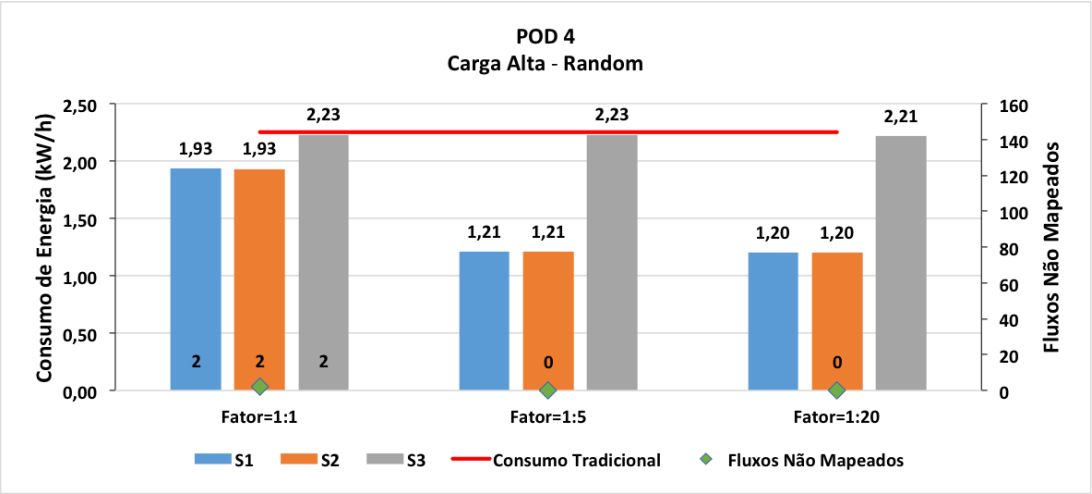
Foi verificado a ocorrência de 2 fluxos não mapeados, o que equivale a 1,25% do total em S1, S2 e S3 com o fator de sobrecarga 1:1 como pode ser observado na Figura 5.1c, diferentemente dos fatores 1:5 e 1:20 que tiveram todos os fluxos mapeados e geraram



(a) Carga baixa



(b) Carga média



(c) Carga alta

Figura 5.1 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Random* e k = 4.

uma economia de energia de 46,22%, enquanto que com o fator 1:1 a economia foi de 14,22%.

5.5.1.2 Workload Stride(1)

As Figuras 5.2a, 5.2b e 5.2c apresentam os gráficos com o consumo de energia da rede para a mesma topologia fat-tree de tamanho $k = 4$, com cargas baixa, média e alta, utilizando o padrão de tráfego *Stride*(1) com 10 fluxos por host. O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

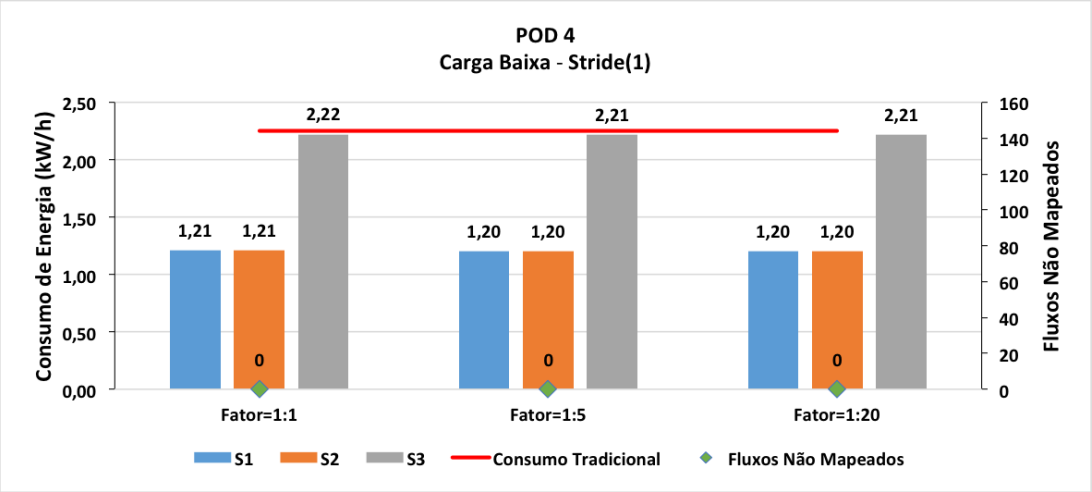
5.5.1.3 Workload Stride(4)

As Figuras 5.3a, 5.3b e 5.3c, apresentam os gráficos com o consumo de energia da rede para uma topologia fat-tree de tamanho $k = 4$, com uma carga de rede baixa (20%), média (50%) e alta (80%) respectivamente, utilizando o padrão de tráfego *Stride*(4) com 10 fluxos por host. Foram verificados nos experimentos ainda, o número de fluxo não mapeados por cada uma das estratégias de mapeamento

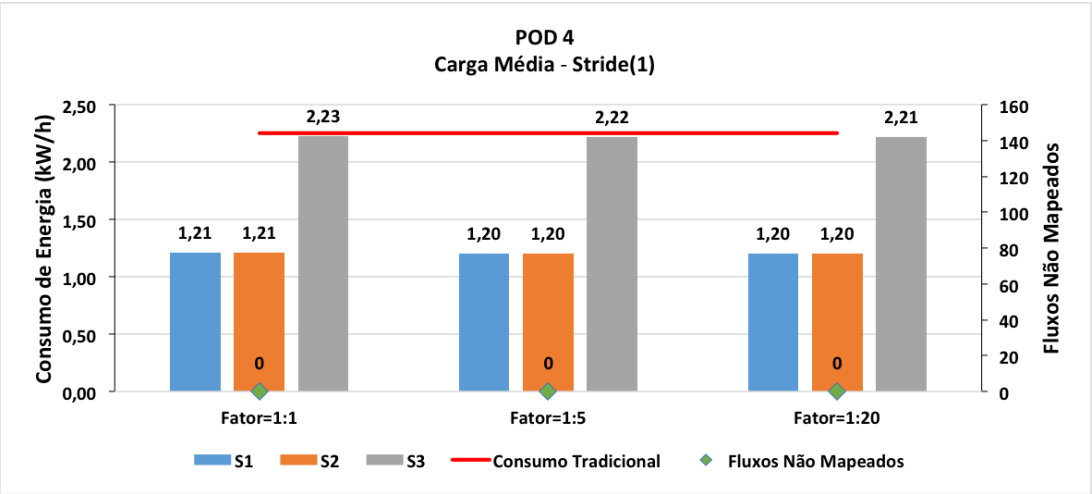
O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

Podemos verificar que o padrão de tráfego *Stride*(4) exibido nas Figuras 5.3a, 5.3b e 5.3c, teve variações no consumo de energia com diferentes cargas de rede. Com carga de rede média (50%) e uso de sobrecarga 1:1, as estratégias S1 e S2 tiveram uma variação entre 16% e 20% de economia de energia. Os demais fatores de sobrecarga permaneceram com a taxa de economia em torno de 46,67% conforme verificado nos resultados anteriores. S3 permaneceu com 1,78% de economia em relação ao consumos tradicional, não trazendo nenhuma modificação em comparação com os experimentos anteriores utilizando o pod de tamanho $k = 4$.

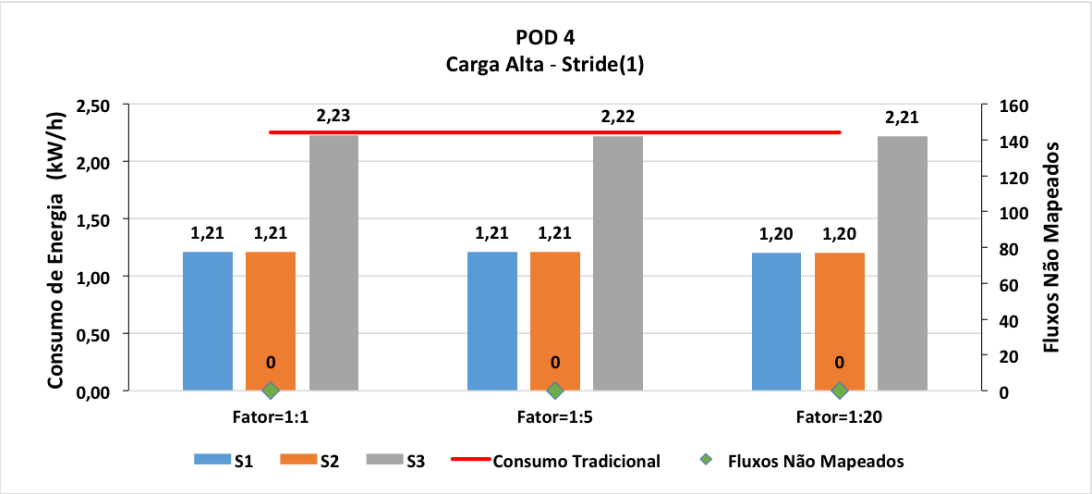
Na carga de rede alta (80%) foi constatado que o consumo de rede aumentou com sobrecarga 1:1, ficando próximo dos valores demonstrados com o padrão de tráfego *Random* na Figura 5.1c, porém sem apresentar perda de fluxos em nenhuma das três estratégias.



(a) Carga baixa

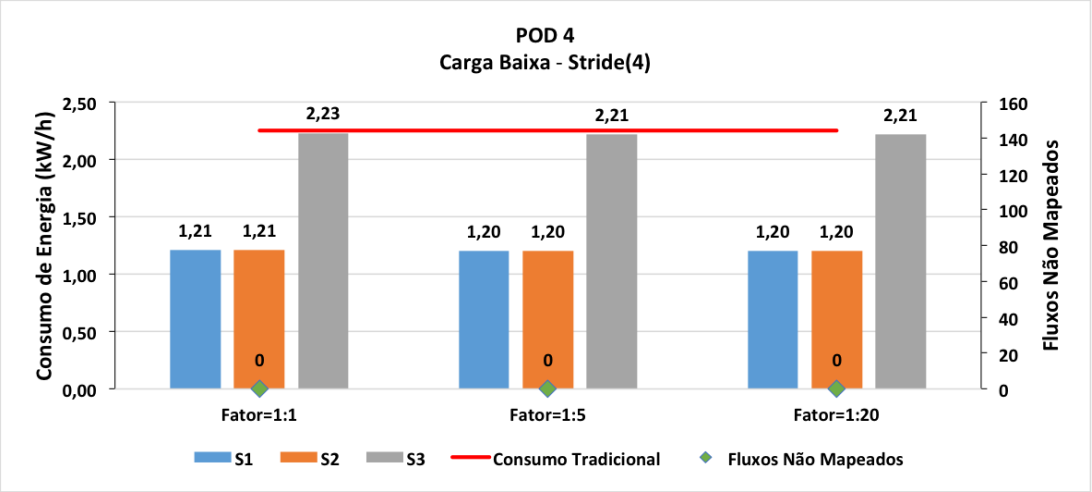


(b) Carga média

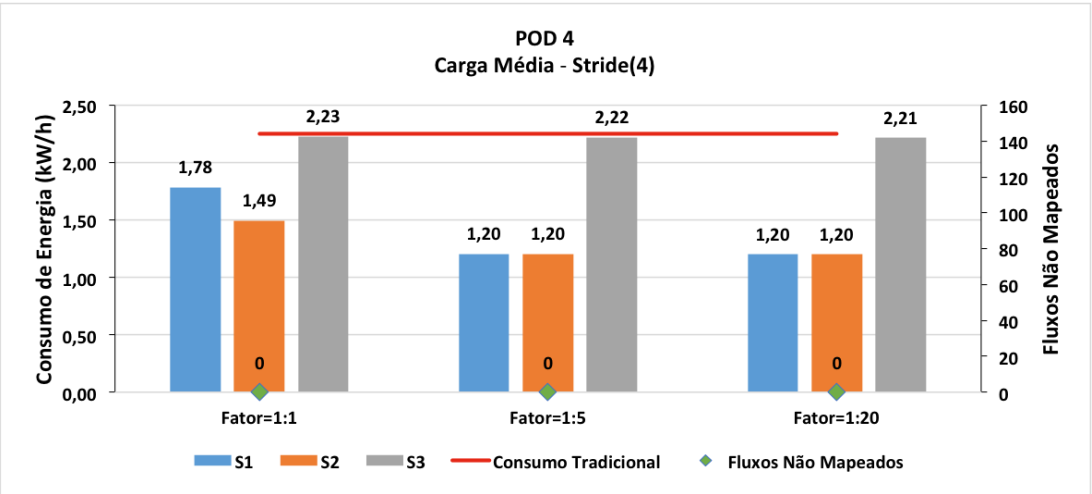


(c) Carga alta

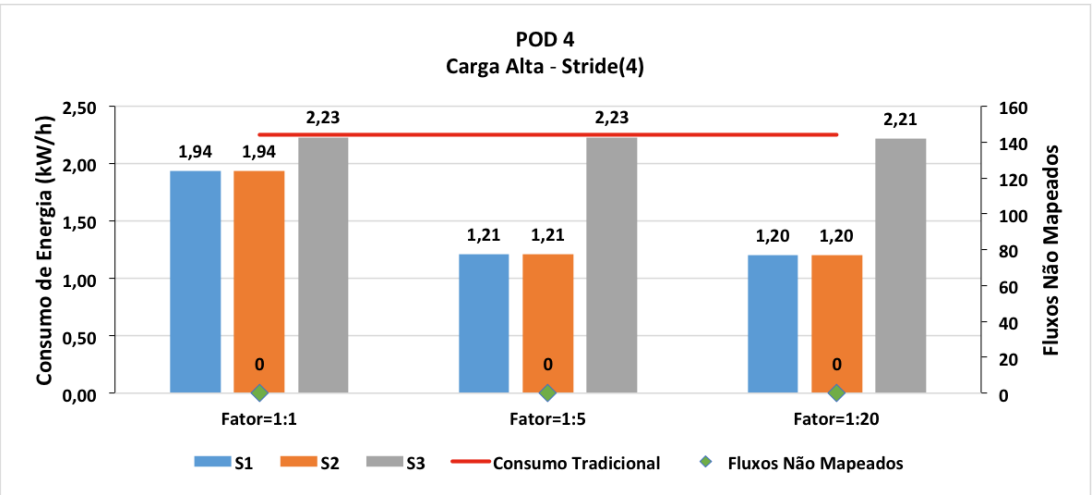
Figura 5.2 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Stride*(1) e k = 4.



(a) Carga baixa



(b) Carga média



(c) Carga alta

Figura 5.3 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Stride*(4) e k = 4.

5.5.2 POD=8 - Configuração de rede [1000, 100, 100]

Os experimentos apresentados a seguir foram realizados com uma topologia fat-tree de tamanho $k = 8$, com a configuração de rede [1000, 100, 100]. Essa configuração representa respectivamente, os links que fazem a ligação da camada de núcleo com a camada de agregação, a ligação da camada de agregação com a camada de borda e a ligação da camada de borda com os hosts. A topologia fat-tree com $k = 8$ utilizada é composta por 16 *switches* de núcleo, 32 *switches* de agregação, 32 *switches* de borda e 128 hosts. A seguir são descritos os experimentos e exibidas as considerações e os resultados organizados por tipo de *workload: Random, Stride*(1) *e Stride*(16):

5.5.2.1 Workload Random

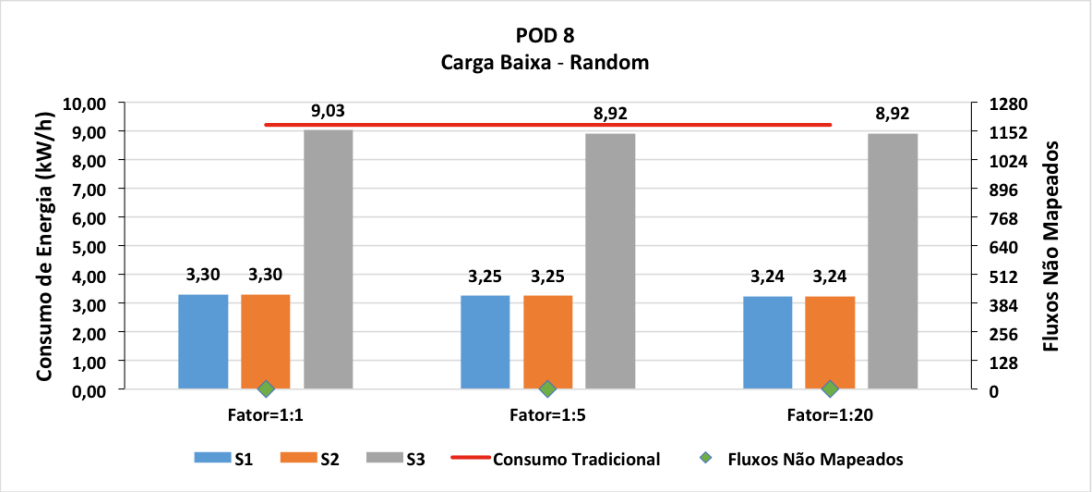
As Figuras 5.4a, 5.4b e 5.4c, apresentam os gráficos com o consumo de energia da rede para uma topologia de Data Center fat-tree com tamanho $k = 4$, com uma carga de rede baixa (20%), média (50%) e alta (80%) respectivamente, com o padrão de tráfego *Random* e uma média de 10 fluxos por host. O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

Com a análise dos resultados obtidos com configuração de rede [1000,100,100] do padrão de tráfego *Random*, percebeu-se uma variação na taxa de economia de energia de 21% $\approx$ 35% para as cargas de rede média e alta com o uso das estratégias S1 e S2. Os resultados obtidos utilizado a estratégia S3 permaneceram baixos, com uma taxa 3,15% de redução no consumo em relação à topologia tradicional. Percebeu-se a ocorrência de 32 fluxos não mapeados, o que equivale a 2,5% do total com o uso do fator de sobrecarga 1:1, permanecendo os fatores 1:5 e 1:20 sem a incidência de fluxos não mapeados.

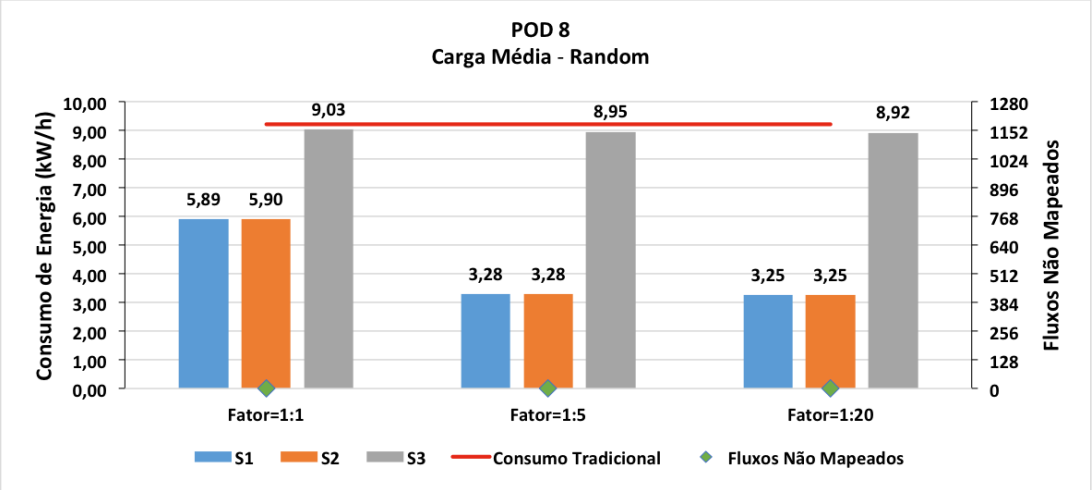
5.5.2.2 Workload Stride(1)

As Figuras 5.5a, 5.5b e 5.5c, apresentam os gráficos com o consumo de energia da rede para a mesma topologia fat-tree de tamanho $k = 4$, com cargas baixa, média e alta, utilizando o padrão de tráfego *Stride*(1) com 10 fluxos por host. O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

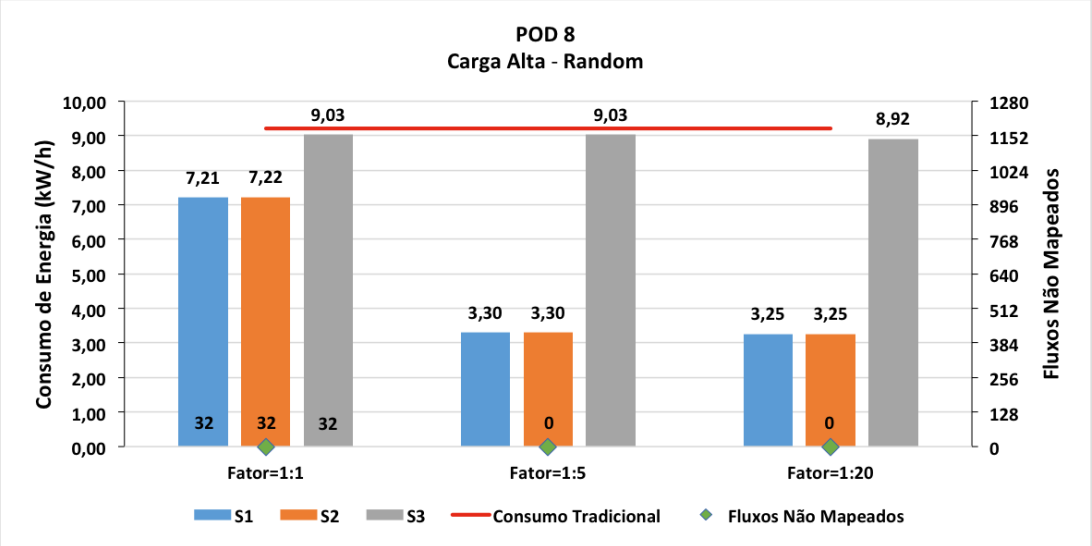
Podemos observar nos resultados dos experimentos com o padrão de tráfego *Stride*(1), que S1 e S2 tiveram os mesmos valores de economia de energia independente



(a) Carga baixa

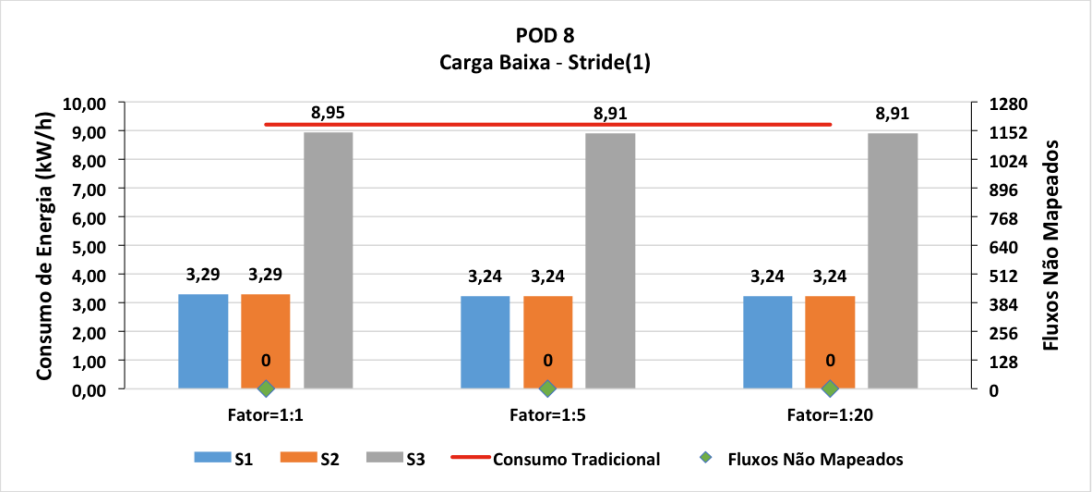


(b) Carga média

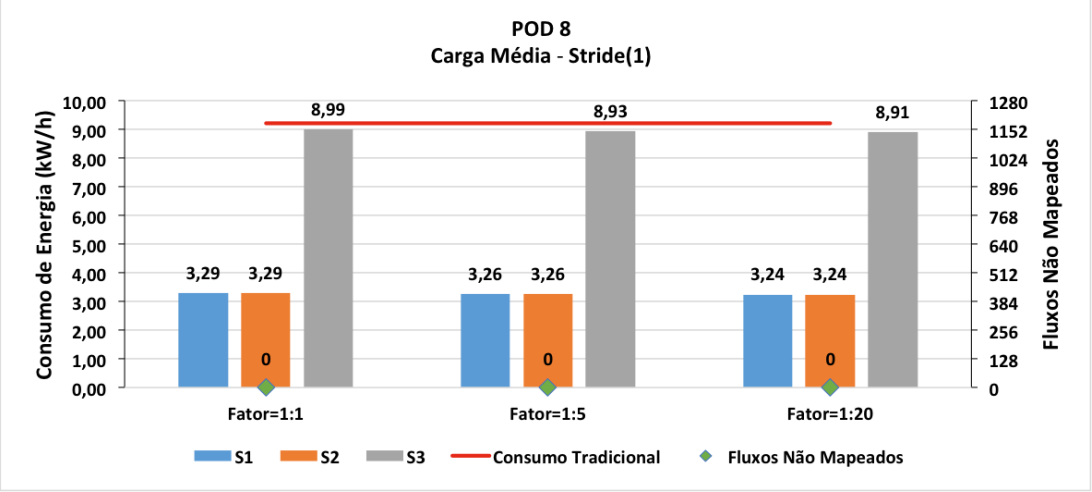


(c) Carga alta

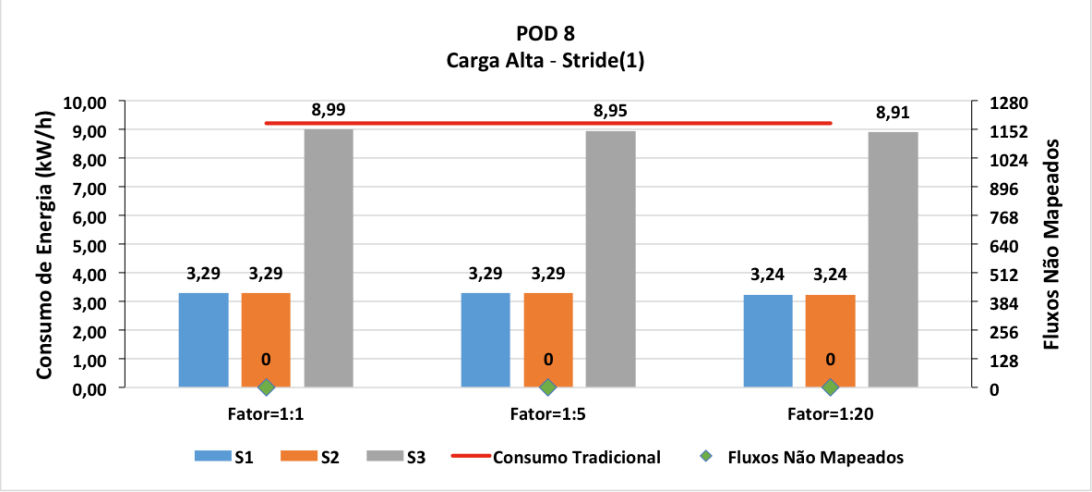
Figura 5.4 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Random* e k = 8.



(a) Carga baixa



(b) Carga média



(c) Carga alta

Figura 5.5 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Stride*(1) e k = 8.

da carga de rede, ficando com a taxa de consumo de energia em torno de 64% menor que o consumo tradicional. Com relação aos fatores de sobrecarga, o consumo se manteve similar em todos os fatores utilizados nos experimentos 1:1, 1:5 e 1:20.

5.5.2.3 Workload Stride(16)

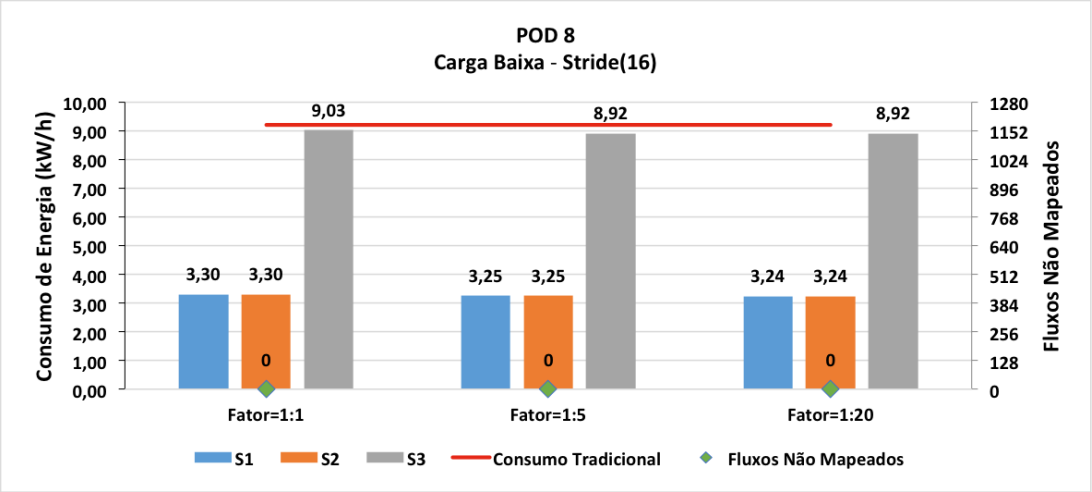
As Figuras 5.6a, 5.6b e 5.6c, apresentam os gráficos com o consumo de energia da rede para uma topologia fat-tree de tamanho $k = 4$, com cargas baixa média e alta, utilizando o padrão de tráfego *Stride*(16) com 10 fluxos por host. O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

Ao analisarmos os resultados obtidos com a configuração de rede [1000,100,100] e o uso do padrão de tráfego *Stride*(16), foi possível observar nas Figuras 5.6a, 5.6b e 5.6c, uma variação da taxa de redução do consumo de energia para cada carga de rede utilizando o fator de sobrecarga de 1:1. Essa variação ocorreu com as estratégias S1 e S2, que tiveram os valores de consumo de energia com cargas de rede baixa, média e alta de 64,21%, 36,12% e 21,80 respectivamente. Com o uso dos fatores 1:5 e 1:20, os valores não sofreram modificações, ficando com taxa de 64,21% menor em relação ao consumo tradicional com as estratégias de mapeamento de fluxo S1 e S2.

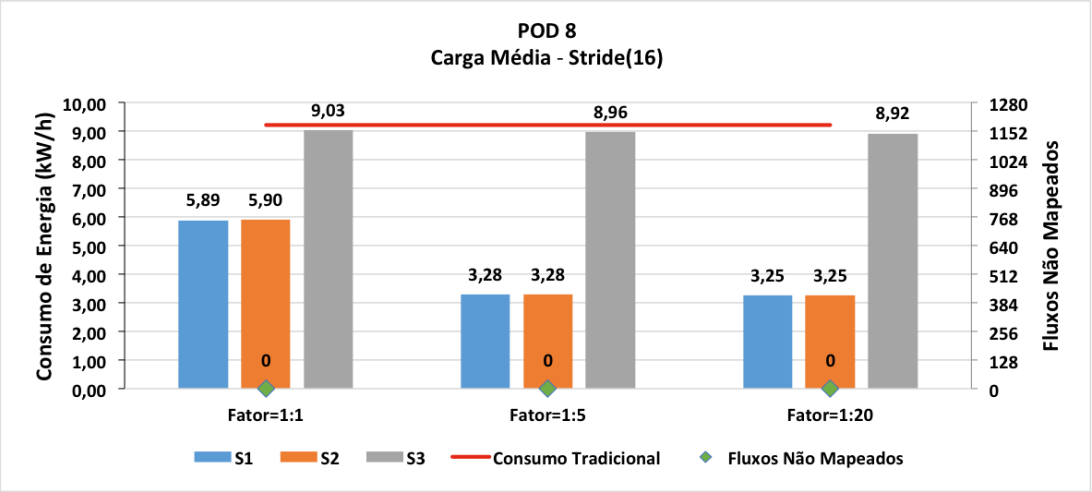
A estratégia S3 permaneceu com a mesma taxa de economia de energia para os três fatores de sobrecarga, mantendo a taxa de 3,25% menor que o consumo tradicional. Essa variação fica evidente nas Figuras 5.6b e 5.6c, pois a medida que a carga de rede entre os 128 hosts da topologia fat-tree com $k = 8$ aumenta para 50% e 80%, o fator 1:1 mantém um número maior de equipamentos e links ativos para atender ao fluxo gerado. Mesmo com a redução da taxa de economia com as cargas média e alta, todos os fluxos foram mapeados pelas estratégias, independente do fator utilizado.

5.5.3 POD=12 - Configuração de rede [1000, 100, 100]

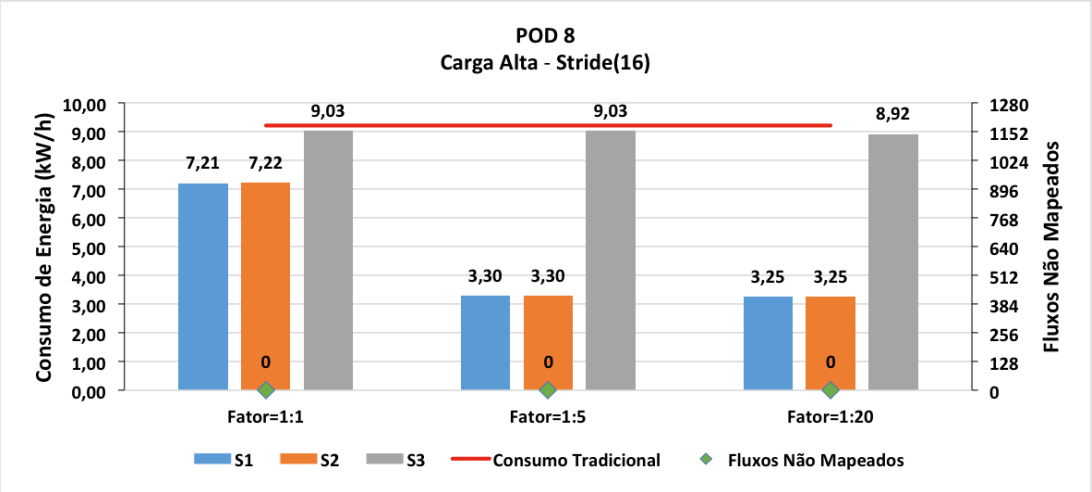
Os experimentos apresentados a seguir foram realizados com uma topologia fat-tree de tamanho $k = 12$, com a configuração de rede [1000, 100, 100]. Essa configuração representa respectivamente, os links que fazem a ligação da camada de núcleo com a camada de agregação, a ligação da camada de agregação com a camada de borda e a ligação da camada de borda com os hosts. A topologia fat-tree com $k = 12$ utilizada é composta por 36 *switches* de núcleo, 72 *switches* de agregação, 72 *switches* de borda e 432 hosts. A seguir são descritos os experimentos e exibidas as considerações e os resultados organizados por tipo de *workload: Random, Stride*(1) *e Stride*(36):



(a) Carga baixa



(b) Carga média



(c) Carga alta

Figura 5.6 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Stride*(16) e k = 8.

5.5.3.1 Workload Random

As Figuras 5.7a, 5.7b e 5.7c apresentam os gráficos com o consumo de energia da rede para uma topologia de Data Center fat-tree com tamanho k = 12, com uma carga de rede baixa (20%), média (50%) e alta (80%) respectivamente, com o padrão de tráfego *Random* e uma média de 10 fluxos por host. O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

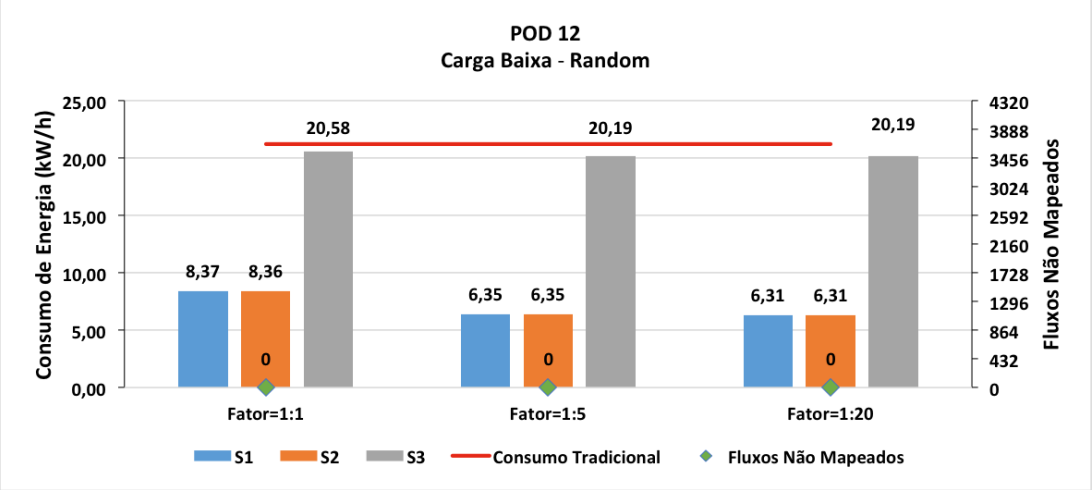
Analisando as Figuras 5.7a, 5.7b e 5.7c, foi possível verificar que o uso dos fatores de sobrecarga 1:5 e 1:20 com a configuração de rede [1000,100,100], colaboraram para a diminuição do número de fluxos não mapeados e também, para o aumento da taxa de economia de energia se comparados ao fator 1:1 com carga alta. O fator 1:20 obteve a taxa de economia de energia de 69,74% com o uso das estratégias S1 e S2 com carga alta. Em comparação com os fatores 1:1 e 1:5 que obtiveram taxas de economia de 25,54% e 60,67% menores em relação a topologia tradicional respectivamente, a economia de energia obtida pelo fator 1:20 foi melhor. Isso se deve em razão do elevados número de hosts (432) com a topologia k = 12 e da carga alta de rede (80%) geradas de forma randômica. Com relação aos fluxos não mapeados, cabe ressaltar que o uso do fator 1:1 em combinação com as estratégias S1, S2 e S3 tiveram 146, 274 e 146 fluxos não mapeados, o que equivale a 3,38%, 6,34% e 3,38% do total de fluxos respectivamente.

5.5.3.2 Workload Stride(1)

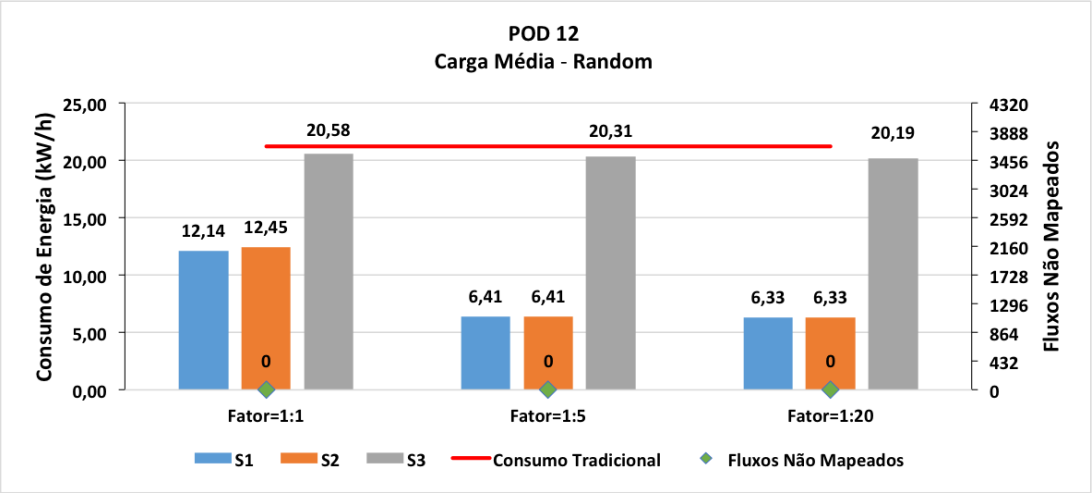
As Figuras 5.8a, 5.8b e 5.8c apresentam os gráficos com o consumo de energia da rede para a mesma topologia fat-tree de tamanho k = 12, com cargas baixa, média e alta, utilizando o padrão de tráfego *Stride*(1) com 10 fluxos por host. O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

Podemos observar nas Figuras 5.8a, 5.8b e 5.8c os resultados dos experimentos com o padrão de tráfego *Stride*(1) com carga de rede de 20%, 50% e 80%, que as estratégias S1 e S2 tiveram os mesmos resultados de economia de energia para todas as cargas de rede utilizadas. A taxa de consumo de energia foi 70% menor em relação ao consumo tradicional.

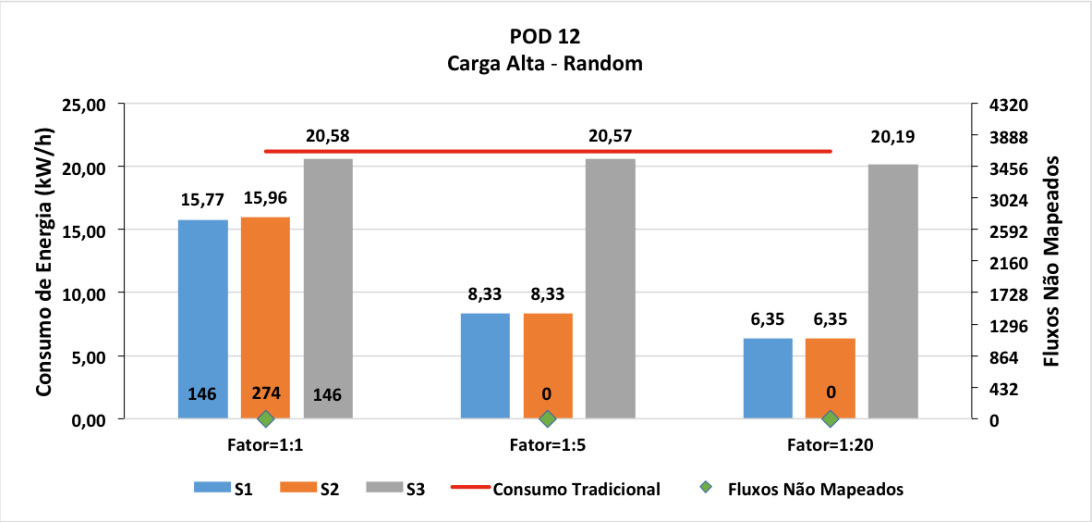
Conclusões similares podem ser obtidas com a análise da estratégia S3, que manteve o mesmo valor de redução do consumo de energia em relação a topologia tradicional de 2,84%. Com relação aos fatores de sobrecarga, o consumo se manteve similar em todos os fatores utilizados nos experimentos 1:1, 1:5 e 1:20



(a) Carga baixa

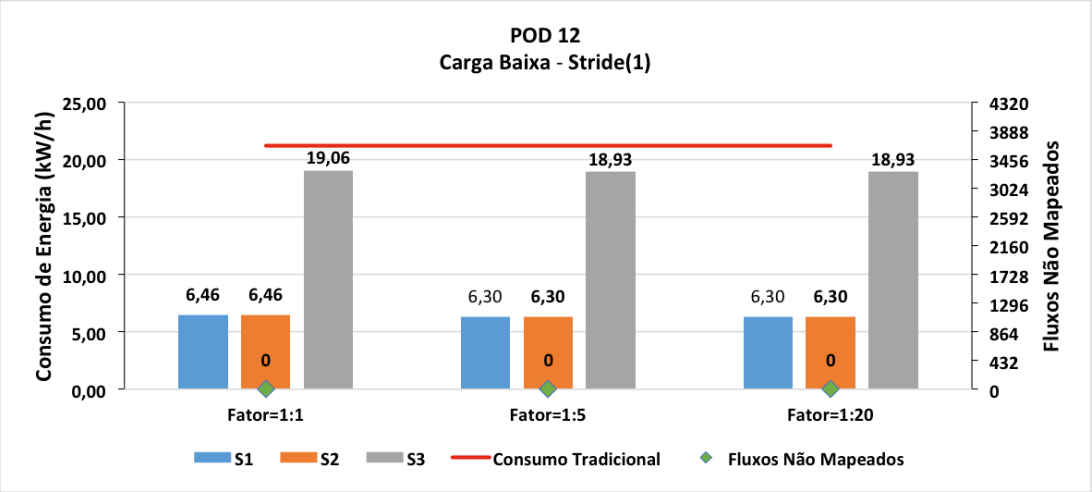


(b) Carga média

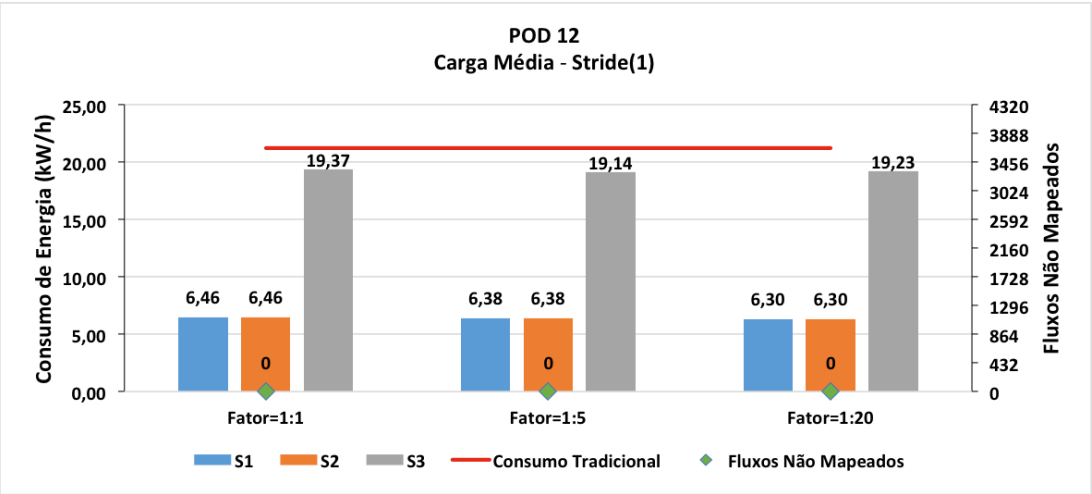


(c) Carga alta

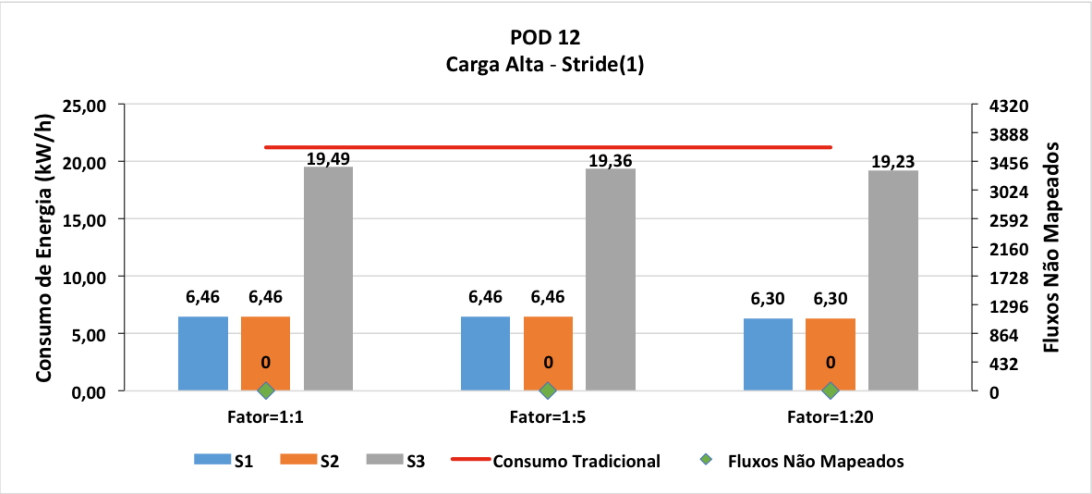
Figura 5.7 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Random* e k = 12.



(a) Carga baixa



(b) Carga média



(c) Carga alta

Figura 5.8 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Stride*(1) e k = 12.

5.5.3.3 Workload Stride(36)

As Figuras 5.9a, 5.9b e 5.9c, apresentam os gráficos com o consumo de energia da rede para uma topologia fat-tree de tamanho k = 12, com cargas baixa, média e alta, utilizando o padrão de tráfego *Stride*(36) com 10 fluxos por host.

O consumo tradicional da rede foi comparado com o consumo de cada uma das estratégias de mapeamento de fluxo, com a aplicação das técnicas de redução do consumo energético juntamente com os fatores de sobrecarga 1:1, 1:5 e 1:20.

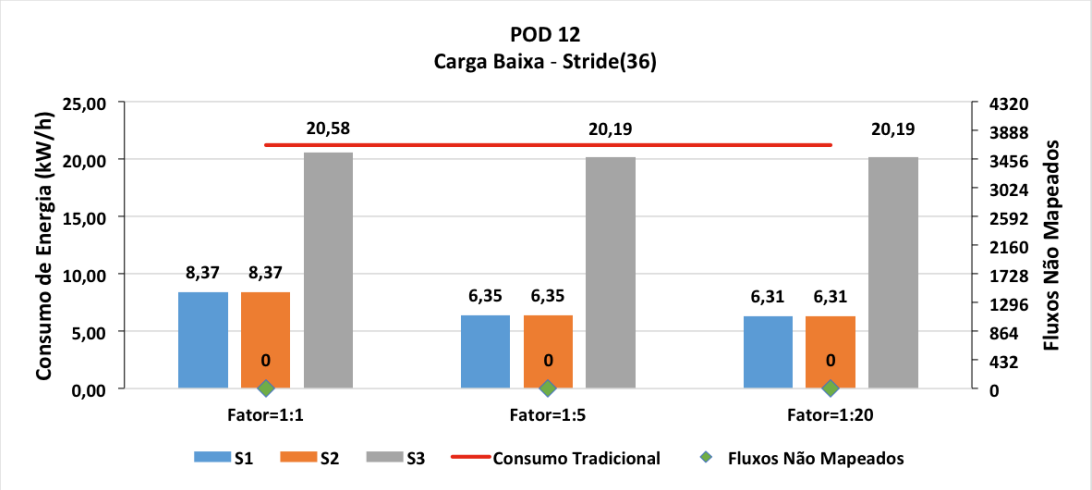
Ao analisarmos os resultados de consumo de energia obtidos com o uso do padrão de tráfego *Stride*(36) e a configuração de rede [1000,100,100], foi possível observar nas Figuras 5.9a, 5.9b e 5.9c, uma variação da taxa do consumo para cada uma das cargas de rede testadas (20%, 50% e 80%) utilizando o fator 1:1 de sobrecarga. Essa variação aconteceu com as estratégias S1 e S2, que tiveram os valores de consumo menores em relação a topologia tradicional de 60,48%, 42,73% e 28,23% para cargas de rede de 20%, 50% e 80% respectivamente. Com o uso dos fatores 1:5 e 1:20, os valores tiveram variação apenas no fator 1:5 com carga de 80%. Com 1:20, não sofreram modificações independente da carga de rede utilizada, ficando com uma taxa em torno de 70% menor em relação ao consumo tradicional com as estratégias de mapeamento de fluxo S1 e S2.

A estratégia S3, permaneceu com a mesma taxa de economia de energia para os três fatores de sobrecarga, mantendo a taxa de 4,67% menor que o consumo tradicional. A variação do consumos com o fator 1:1 fica evidente nas Figuras 5.9a, 5.9b e 5.9c, pois a medida que a carga de rede entre os 432 hosts da topologia fat-tree com k = 12 aumenta de 20% para 50% e de 50% para 80%, o fator de sobrecarga de 1:1 mantem um número maior de equipamentos e links ativos para atender ao fluxo gerado. Com relação ao mapeamento dos fluxos, ocorreu apenas uma perda de 13 fluxos com o fator 1:1 com carga de rede de 80%, os demais fatores tiveram todos os seus fluxos mapeados de forma correta.

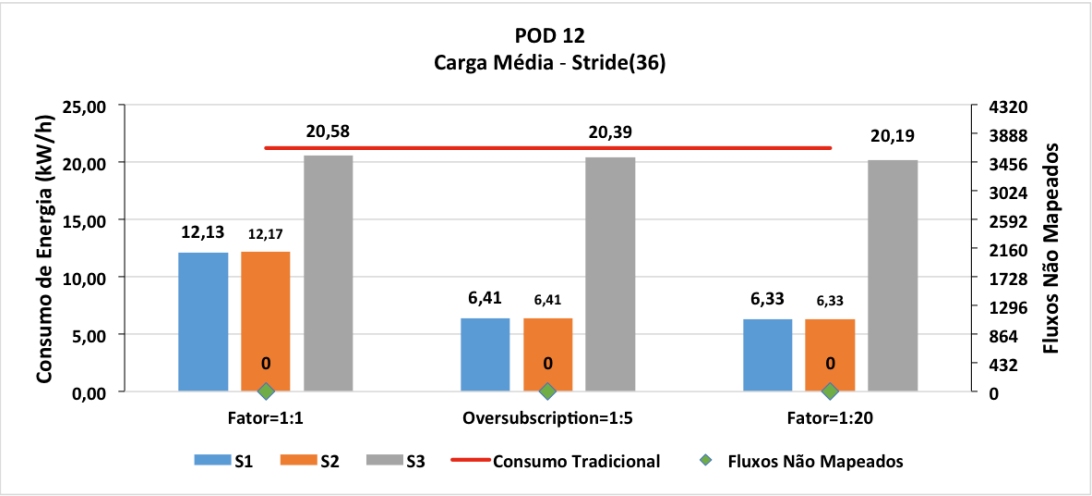
5.6 Análise geral

Este capítulo apresentou o ambiente de simulação, os parâmetros utilizados e os resultados dos experimentos deste trabalho. A partir dos resultados das avaliações, foi possível constatar que quando a utilização da rede é de 100% dos ativos (consumo tradicional), claramente não há economia de energia devido à falta de quaisquer oportunidades de desligamento.

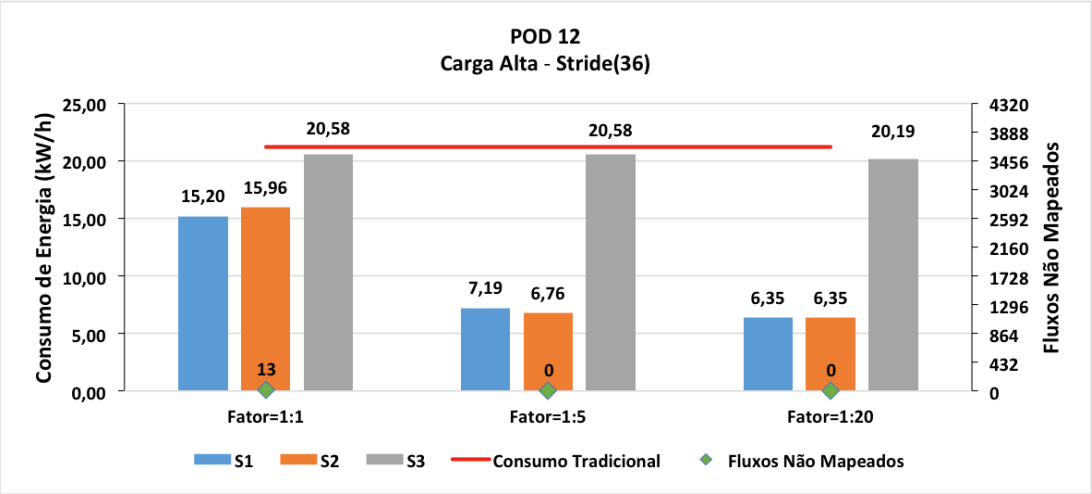
Foram comparados os impactos dos tamanhos da topologia fat-tree em relação a economia de energia para as estratégias propostas. Uma topologia fat-tree com um pod k consiste em $k^2/4$ *switches* de núcleo, $k^2/2$ *switches* de agregação, $k^2/2$ *switches* de borda e $k^3/4$ hosts. Para os padrões de tráfego utilizados, foi alterado o número de hosts



(a) Carga baixa



(b) Carga média



(c) Carga alta

Figura 5.9 – Consumo tradicional X consumo avaliado com S1, S2 e S3 com parâmetros de rede [1000, 100, 100], *Stride*(36) e k = 12.

através da variação do parâmetro k. Em seguida, foi atribuído aleatoriamente a origem e o destino dos fluxos para um conjunto de hosts. Ao comparar a economia de energia para as topologias fat-tree de tamanho $k = 4$, $k = 8$ e $k = 12$, foi possível verificar que consumo de energia aumenta em relação ao consumo tradicional. A redução do consumo de energia a partir do uso das estratégias S1 e S2 aumentou com o a variação do tamanho da topologia. Isto aconteceu tendo em vista que com aumento do valor de k na topologia fat-tree, as estratégias tiveram a possibilidade de escolher entre mais caminhos e mapear o fluxo de forma mais eficiente, devido a existência de mais caminhos disponíveis na rede do Data Center.

Devido ao padrão do tráfego em uma rede de Data Center variar muito, a utilização de diferentes fatores de sobrecarga pode ser uma oportunidade de garantir o correto mapeamento dos fluxos com um índice relevante de eficiência energética. Com base nos resultados obtidos, foi possível constatar que a utilização de diferentes fatores de sobrecarga, principalmente em ambientes de grande porte, agrega disponibilidade aos links. Isso se deve a sobrecarga de rede fornecer a capacidade de largura de banda suficiente para corresponder as garantias de serviço de redes com largura de banda comprometida, o que acabou acontecendo em diversos momentos com carga de rede alta e padrão de tráfego *Random* durante os experimentos.

Ao aplicarmos os fatores 1:5 e 1:20 a taxa de economia de energia na rede chegou a 70,02% com uma topologia fat-tree de tamanho $k = 12$ e 432 hosts e a 64,82% com uma topologia fat-tree de $k = 8$ e 128 hosts. A economia no consumo de energia em relação ao fator 1:1 para as mesmas topologias se manteve igual para $k = 12$, variando para $k = 8$. Isso se deve principalmente ao fato de que ao utilizar o fator de sobrecarga 1:1, foi necessário manter mais equipamentos e portas ligadas, influenciando assim no consumo final da topologia.

Finalmente no que diz respeito ao parâmetro de fluxos não mapeados, percebe-se que com a utilização do *workload Random* com carga de rede alta em todos os tamanhos de pod, ocorreram fluxos não mapeados, como pode ser verificado nas Figuras 5.1c, 5.4c e 5.7c.

6. CONCLUSÃO

Hoje em dia, os Data Centers consomem uma enorme quantidade de energia. Pesquisas têm dedicado seus esforços para melhorar a eficiência do consumo dos servidores. No entanto, não se tem dado atenção suficiente com a eficiência das redes de Data Center. Este trabalho se concentrou na eficiência energética para redes de Data Center, utilizando a combinação de diferentes estratégias de redução do consumo de energia como desligamento de *switches* e portas, diminuição da velocidades das portas com baixa utilização e na consolidação do tráfego com base no mapeamento eficiente dos fluxos.

No Capítulo 4, foi apresentada a motivação, a descrição do problema, os objetivos do trabalho e as estratégias utilizadas no mapeamento dos fluxos gerados no processo de comunicação na rede do Data Center.

No Capítulo 5 foram apresentados os experimentos e os resultados da simulação, onde foi constatado uma redução no consumo de economia na rede de 70,02% com uma topologia fattree de tamanho k = 12, através da utilização dos fatores 1:5 e 1:20 de sobrecarga em relação ao consumo tradicional, onde claramente não há economia de energia devido à falta de quaisquer oportunidades de desligamento. A utilização de diferentes fatores de sobrecarga nas redes de Data Center pode ser uma oportunidade de garantir o correto mapeamento dos fluxos com um índice relevante de eficiência energética.

Este trabalho complementa o estado da arte, pois em comparação com os demais trabalhos listados na Tabela 3.1, são utilizadas mais informações globais da topologia, como levar em consideração a capacidade do link para a seleção dos caminhos, determinando assim os *switches* e links ativos. Em outras palavras, a proposta deste trabalho utilizou mais informações para ajustar a topologia e, portanto, atingiu índices altos de economia de energia na rede do Data Center.

Finalmente, a eficiência energética no uso dos recursos computacionais pode implicar em uma sensível redução dos custos de operação de um Data Centers, pois no Brasil, o custo médio da energia elétrica, em kW/h, é de R$ 0,48[1], isso significa que a redução de 1 kW/h no consumo anual representaria uma economia de aproximadamente R$ 4.204,80 no ano.

6.1 Contribuições

As seguir estão listadas as principais contribuições deste trabalho:

- Estudo do estado da arte das técnicas de redução do consumo de energia em redes de Data Center.
- Desenvolvimento das estratégias de mapeamento de fluxos.

- Avaliação do consumo de energia em redes de Data Center com diferentes carga de rede em topologias fattree.
- Avaliação do impacto do uso de diferentes fatores de sobrecarga na rede do Data Center.

6.2 Trabalhos futuros

Como trabalho futuro, sugere-se a implementação do algoritmo em um controlador OpenFlow, a fim de executar tanto a alocação dos fluxos de rede e o estabelecimentos das conexões, quanto a manutenção dos links, explorando as funcionalidades de centralização dos controladores SDN.

A abordagem deste trabalho foi concentrada em um Data Center de forma individual. Como o crescimento da demanda por aplicações de computação em nuvem, um ponto de pesquisa futura é como criar Data Centers distribuídos e com eficiência energética. A redução do consumo de energia em arquiteturas de Data Center, que estão geograficamente distribuídas é mais complexa, já que vários fatores tem de ser levados em consideração, tais como: localização de servidor, balanceamento de carga nos sites e otimização da sobreposição da topologia de rede virtual.

Espera-se que o trabalho desenvolvido e resultados obtidos não representem o fim da pesquisa, mas sim um importante avanço para a identificação de novas oportunidades relacionadas a economia de energia em redes de Data Center.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] “Aneel - Agência Nacional de Energia Elétrica”. Capturado em: http://www.aneel.gov.br/area.cfm?idArea=49, Dezembro 2015.

[2] “Beacon”. Capturado em: http://www.beaconcontroller.net, Dezembro 2015.

[3] “Data center: Load balancing data center services”. Capturado em: https://learningnetwork.cisco.com/servlet/JiveServlet/downloadBody/3438-102-1-9467/cdccont_0900aecd800eb95a.pdf, Agosto 2015.

[4] “Floodlight”. Capturado em: http://floodlight.openflowhub.org, Dezembro 2015.

[5] “Maestro-platform - A scalable control platform written in java which supports openflow switches”. Capturado em: http://code.google.com/p/maestro-platform/, Dezembro 2015.

[6] “Network simulation introduction”. Capturado em: http://www.openxtra.co.uk/articles/network-simulation, Dezembro 2015.

[7] “Network simulator ns-2”. Capturado em: http://www.isi.edu/nsnam/ns/, Novembro 2015.

[8] “Opendaylight”. Capturado em: http://www.opendaylight.org, Dezembro 2015.

[9] “Openflow - Openflow switch specification”. Capturado em: https://www.opennetworking.org, Outubro 2015.

[10] “Pox: A python-based openflow controller”. Capturado em: http://www.noxrepo.org/pox/about-pox/, Julho 2015.

[11] “Techtarget - programmable network (pn)”. Capturado em: http://www.techtarget.com, Setembro 2015.

[12] Abts, D.; Marty, M. R.; Wells, P. M.; Klausler, P.; Liu, H. “Energy proportional datacenter networks”. ACM SIGARCH Computer Architecture News, 2010, pp. 338–347.

[13] Adnan, M. A.; Gupta, R. “Path consolidation for dynamic right-sizing of data center networks”. IEEE 6th International Conference on Cloud Computing (CLOUD), 2013, pp. 581–588.

[14] Al-Fares, M.; Loukissas, A.; Vahdat, A. “A scalable, commodity data center network architecture”. ACM SIGCOMM Computer Communication Review, 2008, pp. 63–74.

[15] Bari, M. F.; Boutaba, R.; Esteves, R.; Granville, L. Z.; Podlesny, M.; Rabbani, M. G.; Zhang, Q.; Zhani, M. F. “Data center network virtualization: A survey”, *IEEE Communications Surveys & Tutorials*, vol. 15–2, 2013, pp. 909–928.

[16] Benson, T.; Akella, A.; Maltz, D. A. “Network traffic characteristics of data centers in the wild”. 10th ACM SIGCOMM Conference on Internet Measurement, 2010, pp. 267–280.

[17] Benson, T.; Anand, A.; Akella, A.; Zhang, M. “Understanding data center traffic characteristics”, *ACM SIGCOMM Computer Communication Review*, vol. 40–1, 2010, pp. 92–99.

[18] Brown, R.; et al.. “Report to congress on server and data center energy efficiency: Public law 109-431”, *Lawrence Berkeley National Laboratory*, 2008.

[19] Budhiraja, M.; Saggar, S. “Green IT: Harvesting heat using tpv (thermophotovoltaic)”, *International Journal of Managment, IT and Engineering*, vol. 3–9, 2013, pp. 323–333.

[20] Calheiros, R. N.; Ranjan, R.; Beloglazov, A.; De Rose, C. A.; Buyya, R. “Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”, *Software: Practice and Experience*, vol. 41–1, 2011, pp. 23–50.

[21] Christofides, N.; Theory, G. “An algorithmic approach”. New York: Academic Press Inc, 1975, 415p.

[22] Coffman Jr, E. G.; Garey, M. R.; Johnson, D. S. “Approximation algorithms for bin packing: a survey”, 1996, pp. 46–93.

[23] Couto, R. S.; Campista, M. E. M.; Costa, L. H. M. “Uma avaliação da robustez intra data centers baseada na topologia da rede”, *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2012, pp 610–623.

[24] Dally, W. J.; Towles, B. P. “Principles and practices of interconnection networks”. San Francisco: Morgan Kaufmann Publishers Inc, 2004, 550p.

[25] Dijkstra, E. W. “A note on two problems in connexion with graphs”, *Numerische mathematik*, vol. 1–1, 1959, pp. 269–271.

[26] Farrington, N.; Andreyev, A. “Facebook’s data center network architecture”. IEEE Optical Interconnects Conf, 2013.

[27] Foster, N.; Freedman, M. J.; Harrison, R.; Rexford, J.; Meola, M. L.; Walker, D. “Frenetic: a high-level language for openflow networks”. Workshop on Programmable Routers for Extensible Services of Tomorrow, 2010, pp. 6.

[28] Goudarzi, H.; Pedram, M. “Energy-efficient virtual machine replication and placement in a cloud computing system”. IEEE 5th International Conference on Cloud Computing (CLOUD), 2012, pp. 750–757.

[29] Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. “Nox: towards an operating system for networks”, *SIGCOMM Comput. Commun. Rev.*, vol. 38–3, Jul 2008, pp. 105–110.

[30] Guedes, S.; Conceição, V.; Carvalho, N.; Rodrigues, L. “Plataforma de desenvolvimento e simulação de protocolos”, *VIII Conferência sobre Redes de Computadores (CRC’05)*, 2005.

[31] Hammadi, A.; Mhamdi, L. “A survey on architectures and energy efficiency in data center networks”, *Computer Communications*, vol. 40, 2014, pp. 1–21.

[32] Heller, B.; Seetharaman, S.; Mahadevan, P.; Yiakoumis, Y.; Sharma, P.; Banerjee, S.; McKeown, N. “Elastictree: Saving energy in data center networks”. 7th USENIX Conference on Networked Systems Design and Implementation, 2010, pp. 19–21.

[33] Henderson, T. R.; Lacage, M.; Riley, G. F.; Dowell, C.; Kopena, J. “Network simulations with the ns-3 simulator”, *SIGCOMM demonstration*, vol. 14, 2008.

[34] Huong, T.; Schlosser, D.; Nam, P.; Jarschel, M.; Thanh, N.; Pries, R. “Ecodane—reducing energy consumption in data center networks based on traffic engineering”. 11th Würzburg Workshop on IP: Joint ITG and Euro-NF Workshop Visions of Future Generation Networks (EuroView2011), 2011.

[35] Knight, S.; Jaboldinov, A.; Maennel, O.; Phillips, I.; Roughan, M. “Autonetkit: Simplifying large scale, open-source network experimentation”, *SIGCOMM Computer Communication Review*, vol. 42–4, Ago 2012, pp. 97–98.

[36] Koomey, J. “Growth in data center electricity use 2005 to 2010”, *Oakland, CA: Analytics Press. August*, vol. 1, 2011, pp. 2010.

[37] Koponen, T.; Casado, M.; Gude, N.; Stribling, J.; Poutievski, L.; Zhu, M.; Ramanathan, R.; Iwata, Y.; Inoue, H.; Hama, T.; et al.. “Onix: A distributed control platform for large-scale production networks”. 9th USENIX Conference on OSDI, 2010, pp. 1–6.

[38] Kurp, P. “Green computing”, *Communication of the ACM*, vol. 51–10, Out 2008, pp. 11–13.

[39] Lantz, B.; Heller, B.; McKeown, N. “A network in a laptop: Rapid prototyping for software-defined networks”. 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, pp. 19:1–19:6.

[40] Lee, C.; Rhee, J. K. K. “Traffic off-balancing algorithm: Toward energy proportional datacenter network”. 17th Opto-Electronics and Communications Conference (OECC), 2012, pp. 409–410.

[41] Lee, J.; Turner, Y.; Lee, M.; Popa, L.; Banerjee, S.; Kang, J.-M.; Sharma, P. “Application-driven bandwidth guarantees in datacenters”. 2014 ACM Conference on SIGCOMM, 2014, pp. 467–478.

[42] Leiserson, C. E. “Fat-trees: universal networks for hardware-efficient supercomputing”, *Computers, IEEE Transactions on*, vol. 100–10, 1985, pp. 892–901.

[43] Liu, N.; Dong, Z.; Rojas-Cessa, R. “Task and server assignment for reduction of energy consumption in datacenters”. IEEE 11th International Symposium on Network Computing and Applications (NCA), 2012, pp. 171–174.

[44] Mahadevan, P.; Banerjee, S.; Sharma, P. “Energy proportionality of an enterprise network”. First ACM SIGCOMM Workshop on Green Networking, 2010, pp. 53–60.

[45] Mahadevan, P.; Sharma, P.; Banerjee, S.; Ranganathan, P. “A power benchmarking framework for network devices”. 8th International IFIP-TC 6 Networking Conference, 2009, pp. 795–808.

[46] Mankoff, J.; Kravets, R.; Blevis, E. “Some computer science issues in creating a sustainable world”, *Computer*, vol. 41–8, 2008, pp. 102–105.

[47] McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. “Openflow: enabling innovation in campus networks”, *ACM SIGCOMM Computer Communication Review*, vol. 38–2, 2008, pp. 69–74.

[48] Nam, T. M.; Thanh, N. H.; Thu, N. Q.; Hieu, H. T.; Covaci, S. “Energy-aware routing based on power profile of devices in data center networks using sdn”. 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015, pp. 1–6.

[49] Romero de Tejada Muntaner, G. “Evaluation of openflow controllers”, Tese de Doutorado, KTH Royal Institute of Technology, Stockholm, Sweden, 2012, 90p.

[50] Saino, L.; Cocora, C.; Pavlou, G. “A toolchain for simplifying network simulation setup”. 6th International Conference on Simulation Tools and Techniques (ICST), 2013, pp. 82–91.

[51] Sanner, M. F.; et al.. “Python: a programming language for software integration and development”, *J Mol Graph Model*, vol. 17–1, 1999, pp. 57–61.

[52] Schult, D. A.; Swart, P. “Exploring network structure, dynamics, and function using networkx”. 7th Python in Science Conferences (SciPy 2008), 2008, pp. 11–16.

[53] Shimonishi, H.; Chiba, Y.; Takamiya, Y.; Sugyo, K. “Trema: An open source openflow controller platform”, *GENI Engineering Conference - (Poster GEC-11)*, 2011.

[54] Taheri, M. M.; Zamanifar, K. “2-phase optimization method for energy aware scheduling of virtual machines in cloud data centers”. International Conference for Internet Technology and Secured Transactions (ICITST), 2011, pp. 525–530.

[55] Varga, A.; et al.. “The omnet++ discrete event simulation system”. European simulation multiconference (ESM’2001), 2001, pp. 65.

[56] Verdi, F. L.; Rothenberg, C. E.; Pasquini, R.; Magalhães, M. “Novas arquiteturas de data center para cloud computing”, *Minicursos do XXVIII SBRC*, 2010, pp. 103–152.

[57] Villarreal, S. R.; Villarreal, M. E.; Westphall, C. B.; Westphall, C. M. “Legacy network infrastructure management model for green cloud validated through simulations”, *International Journal on Advances in Intelligent Systems*, vol. 7, 2014, pp. 124–135.

[58] Vu, T. H.; Nam, P. N.; Thanh, T.; Linh, N. D.; Thien, T. D.; Thanh, N. H.; et al.. “Power aware openflow switch extension for energy saving in data centers”. International Conference on Advanced Technologies for Communications (ATC), 2012, pp. 309–313.

[59] Wang, X.; Yao, Y.; Wang, X.; Lu, K.; Cao, Q. “Carpo: Correlation-aware power optimization in data center networks”. IEEE 31st Annual International Conference on Computer Communications (INFOCOM), 2012, pp. 1125–1133.

[60] Zhang, Y.; Ansari, N. “Hero: Hierarchical energy optimization for data center networks”. 2012 IEEE International Conference on Communications (ICC), 2012, pp. 2924–2928.